Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality

M. S. Franzin[1], **E. C. Freuder**[2], **F. Rossi**[1], **R. Wallace**[2]
1: Department of Pure and Applied Mathematics
University of Padova, Italy.
frossi@math.unipd.it, mfranzin@studenti.math.unipd.it
2: Cork Constraint Computation Centre
University College Cork, Cork, Ireland.
{r.wallace,e.freuder}@4c.ucc.ie

Abstract

We consider multi-agent systems with preferences, which are just standard multi-agent systems, except that each agent can set some preferences over its local data. This makes these systems more flexible and realistic, since it is possible to represent possibilities, costs, probabilities, preferences, and penalties. However, it also transforms the search for a feasible solution into the search for an optimal, or good enough, solution, since the use of preferences naturally implies the adoption of an optimization criterion, both for each agent and also for the system as a whole. Thus solution quality becomes an important aspect of such systems. Moreover, each agent may want to keep its information as private as possible. Thus privacy loss is another important aspect of such systems. Finally, an obvious crucial aspect is efficiency, since we would like to find a good solution as quickly as possible.

In this paper, we study the relations among these three aspects in the context of a multi-agent meeting scheduling system. We do this by first implementing a multi-agent system that incorporates preferences, and then by running experiments to capture the interesting and useful trade-offs among these three aspects.

Introduction

In multi-agent systems, usually each agent has its own knowledge and reasoning engine that can solve local problems. In many cases, this knowledge can be expressed by a set of constraints, restricting the combination of values of some variables (Tsang 1993). Such constraints can be hard (that is, satisfiable or not), but in most real-life scenario they rather specify preferences (Bistarelli, Montanari, & Rossi March 1997). In this case, problem solving means finding the most preferred solution, according to an optimization criterion. In a multi-agent system with preferences, each agent has its own optimization criterion, possibly different from that of the other agents.

Moreover, agents need to communicate, to solve problems that involve knowledge coming from several agents' sites. Thus they need to exchange information to obtain a global solution which satisfies all the agents. Also for such global problem solving tasks, there must be an optimization criterion, possibly different from that of the single agents, which guides the search to the best global solution.

It is often desirable that agents exchange as little information as possible, to keep their data private, while in the meantime reaching a good global solution in a short amount of time. Thus one would like to minimize privacy loss, to maximize the solution quality, and to be as fast as possible. However, it is predictable that optimizing all these three aspects (efficiency, privacy loss, solution quality) may sometimes be difficult. Thus we think it is interesting to study the relations among them.

In this paper we study these issues in the context of a multi-agent meeting scheduling system. In this system, each agent has its own calendar, with some meetings already scheduled. In addition, agents can have preferences for time slots and meeting locations, and usually there are constraints between time-slots which describe the distance to reach one location from another one.

In this paper, the optimization criterion for all agents and also for the global tasks is to maximize the minimum preferences, thus we use the fuzzy constraint formalism (Dubois, Fargier, & Prade 1993; Ruttkay 1994; Schiex 1992). However, in general, each agent can have its own optimization criterion, and the global one can be different as well.

To solve a global task, agents communicate in several proposal phases. In each phase, one agent proposes a solution to the other agents, which can either accept or reject the proposal. If there is some rejection, a new proposal phase is started. If they all accept, then we have a solution, whose preference value is the minimum among all the agents' preferences for that time-slot. At this point, a new negotiation process starts, to find a better solution, if needed. This continues until no better solution is found, which means that we already have an optimal solution.

To speed up the search for a solution and exchange little amount of information, each agent has its own constraints, describing its own knowledge, but it also accumulates a set of constraints which represents its view of the other agents. In fact, whenever agents communicate during the proposal phases, the information they exchange can be used to build an approximation of the constraint set of the other agents.

For example, when a proposal is rejected, the proposing agent can infer that the rejecting agent has already a meeting in that slot, or in a slot which is not reachable from that one because of the distance constraints. Also, whenever a

proposal is accepted, the proposing agent can infer that the accepting agent does not have a meeting in that slot, nor in any slot which would be incompatible with this one because of the distance constraints. Finally, each agent receiving a proposal can infer that the proposing agent does not have meeting in that slot, nor in any slot that would be incompatible because of the distnace constraints.

We developed our system starting from an existing system which did not have preferences but just hard constraints (Freuder, Minca, & Wallace 2001). For that system, the relation between privacy loss and efficiency has been deeply studied and shown with experimental results. We added preferences to that system and we observed the behavior of the new system under several conditions, to learn the relations among solution quality, efficiency, and privacy loss. This paper describes the new system and reports and discusses the experimental results.

Other approaches to multi-agent meeting scheduling systems have been proposed (Garrido & Sycara 1995; Luo, Leung, & Lee 2000; Tsuruta & Shintani 2000; Scott, Jenkin, & Senjen 1988; Sen, Haynes, & Arora 1997). Some of them don't have preferences, others are not completely distributed because they add a coordinator agent. More importantly, none of them studies the relations among efficiency, privacy loss, and solution quality, which is the main focus of this paper.

Hard and soft constraints

Hard constraints (Tsang 1993) are restrictions over the combination of values of certain variables, which state what is allowed and what is forbidden. They are said to be "hard" because there is no possibility for tolerance: either a tuple of values is allowed, or it is not.

The concept of soft constraints tries to make the hard constraint formalism more flexible and more widely applicable to real life. It tries to add enough machinery to be able to represent more flexible situations, like those where one constraint is more important than another, or a combination of values is preferred over another, or we have to deal with costs, possibilities, probabilities, or the like.

This is done in a rather simple and elegant way, by associating to each combination of variables' values (in a constraint), an element taken from an ordered set. These elements can play the role of preferences, costs, probabilities, and many other things. It is important that the set of such elements is ordered, since this order can help us compare two elements to understand which is the preferred one. However, the order can also be partial, showing the possibility that some elements are incomparable. This may happen, for example, when the elements represent lists of values, and each list item has its own total order, as in a multi-criteria constraint optimization problem.

In this more general formalism, hard constraints are just a very simple instance where the set of preference values has just two elements: allowed and forbidden, with allowed better than forbidden. Thus the extension from hard to soft constraints is very similar to passing from a two-valued to a multi-valued logic.

Once every tuple in every constraint has its preference value, we also need to understand how to generate the preference value of a complete assignment (of values to variables) from the preference values of the tuples in the constraints. To do this, we need a combination operator, say $\times$, which performs this work.

In a soft constraint setting, the goal is to find a complete assignment with the best preference value, where the meaning of "best" is given by the order over the preference values.

For example, in hard constraints the combination operator is AND, since all constraints need to be satisfied in order for a complete assignment to be a solution (and thus be associated to the allowed preference value).

Another, widely used, example of soft constraints is the class of fuzzy constraints: here, preference values are taken from the [0,1] interval, which is totally ordered, and where 1 is the best preference and 0 the worst one. The combination operator is min. This means that, in fuzzy constraint solving, the optimization goal is to find a complete assignment which has the maximum preference value, where the preference value of a solution is the minimum preference among all those of the tuples which are part of the solution.

It is possible to see that the set of preference values, the ordering over it, and the combination operator, form a mathematical structure which is very similar to a semiring (Bistarelli, Montanari, & Rossi March 1997) (actually, a c-semiring, which is just a semiring plus two additional properties).

In this paper, preferences will be added to multi-agent systems via the use of fuzzy constraints. However, any instance of the soft constraint formalism could be used to add preferences to such systems.

Preferences in meeting scheduling problems

In this paper we focus on a specific class of multi-agent systems, whose task is to reach agreement on a common meeting. Our version of the multi-agent meeting scheduling problem involves k agents. Each agent has its own calendar, which consists of appointments that are already fixed. Each appointment consists of a certain location (a city in our case) and a certain time (in our case, a single time slot).

The problem is to find a meeting time and place where all agents can meet together. Thus this meeting time has to be compatible with the schedules of the agents.

Each agent can also set a preference value for each city in each time slot. If each time slot is represented by a variable, whose values are the possible cities, according to the previous section, such preferences are associated to the values of the variables, and are taken from an ordered set P; the order tells which value of P represents a higher or a lower preference. In this paper, we consider values between 0 and 1, with higher values denoting higher preferences.

In addition, since each meeting has a preference value, these must be combined to obtain an overall preference. An aggregation operator, say $\times$, combines these preference values and obtains the overall preference of the meeting; the goal is to find a meeting time and place which has the highest overall preference. For this paper, the aggregation oper-

ator is min. Thus the goal is to maximize the minimum of the agents' preferences.

For this paper, we consider five possible cities (New York, Boston, Philadelphia, Los Angeles, ...), and meeting times are between 9AM to 6PM, on any day of the week. Each meeting is one hour long. The constraints describing the calendar of each agent basically set the already fixed meetings and the distance constraints among cities, which are shown in Figure 1.

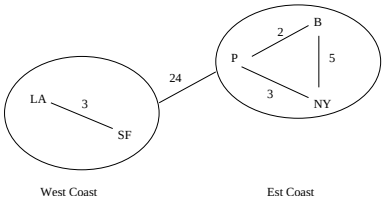

Figure 1: The distance constraints.

Notice that, while preferences can be set over single time-city slots, distance constraints are hard. That is, if two cities in two different time slots are not reachable from each other because the slots are too close (closer than the number of hours required to travel from one city to the other one), then the constraint is not satisfied by such two cities. It is instead satisfied if they are reachable. In other words, no preference is put over the compatible pairs of cities in this paper. However, it is very reasonable to think that in real life we would prefer to travel as little as possible, for example, and this could be easily expressed via the use of soft constraints rather than hard constraints also for the distance requirements.

Agent communication and knowledge inference

For this paper, we consider a very simple form of agent comunication. In fact, each agent can communicate with the others in only one of these ways:

- by proposing a time and a city, and a preference for such a proposal;
- by accepting a proposal and giving its preference value (higher than 0);
- by rejecting a proposal (that is, giving a 0 preference value).

Moreover, when an agent makes a proposal, this proposal is communicated to all the other agents. On the other hand, when an agent responds to a proposal, it responds only to the proposing agent.

More elaborate forms of communication can be considered. For example, in (Freuder, Minca, & Wallace 2001) they also allow agents to give one or all reasons for a rejection (in the form of already scheduled meetings).

Whenever two agents communicate in one of the above forms, they get some information about each other's schedule, because of the information that is actually communicated. Moreover, they can also infer some more knowledge about each other. In other words, they incrementally build an approximate view of the other agents' calendars. This knowledge can then be used to guide the selection of the future proposals, and thus hopefully speed up the whole solution process.

For example, when a proposal is rejected, the proposing agent can infer that the rejecting agent either has already a meeting in that slot already, or in a slot which is not reachable from that one because of the distance constraints. As an example, if agent A proposes to meet at 2pm on Monday in New York, and agent B rejects this proposal, then agent A can infer some knowledge about B: that either it has a meeting already at 2pm on Monday, or it has a meeting in a time-city that is incompatible with being 2pm in New York. However, the inference that can be made from a rejection is very vague, since nothing can be surely deleted from the set of possible future proposals.

On the other hand, whenever a proposal is accepted, the proposing agent can infer that the accepting agent does not have a meeting in that slot, nor in any slot which would be incompatible with this one because of the distance constraints. For the same proposal as above (2pm on Monday in NY), if agent B accepts the proposal, it means that surely it means that it must not have a meeting set for 2pm on Monday (in any city), and also that it does not have a meeting in a time-city that is incompatible with 2pm in New York. For example, B cannot have a meeting at 3pm on Monday in Los Angeles, because there is no time to get from NY to LA in one hour.

Finally, each agent receiving a proposal can infer that the proposing agent does not have meeting in that slot, nor in any slot that would be incompatible because of the distance constraints. Thus, in the example above, B can infer that A does not have a meeting already at 2pm on Monday in NY. So the inference is the same as for the acceptance case.

In general, we call "positive information" the information which is inferred from an agent A about another agent B, and which describes the time-city slots where A has discovered that B does not have meetings already, and thus are possibly good candidate slots for a common meeting. Instead, we call "negative information" the information which is inferred from A about B, and which describes the time-city slots where A has discovered that B has already a meeting, or it cannot reach a certain location because of other meetings, and thus cannot meet with A.

These two kinds of information are very similar to what is called "possible-has-meetings deleted" and "possible-can-meet deleted" in (Wallace, Freuder, & Minca 2002). In fact, positive information is related to the fact that we discover that some agent cannot have a meeting already in a certain time-city slot, while negative information is related to the discovery that an agent cannot possibly meet the others in a time slot.

Notice that the concept of positive information, with which we discover that an agent cannot have a meeting in certain locations at certain times, allows also to discover the agents' open slots: those time slots in which we are sure that an agent does not have any meeting already, in none of the cities. These are in fact easily related to acceptances of

meeting proposals.

Searching for a feasible or optimal solution: the negotiation scenario

Without preferences, the goal of a multi-agent meeting scheduling system is to find a feasible solution, that is, the first time-city slot which is acceptable for all the agents, given their schedule. With the addition of preferences over each time-city slot, we have turned the problem into the search for an optimal solution, which, as defined above, should mazimize the minimum preference over all agents.

For example, consider three agents, say A, B and C, and a proposal for 2pm on Monday in NY. Each agent has a preference value for this time-city slot. Agent A could give it preference 0.5, agent B preference 0.8, and agent C preference 0.3. Then, the overall preference for such a proposal is $0.3 = min(0.5, 0.8, 0.3)$. Consider now a different proposal, say 3pm on Tuesday in Los Angeles, and assume the following preference values: 0.5 for A, 0.5 for B, and 1 for C. Then the overall preference value is 0.5. Thus this solution is better than the previous one.

Notice that, in this preference scheme, a rejection (thus a preference value of 0) from one or more of the agents automatically gives an overall preference value of 0, thus making the proposal one of the worst ones.

To find an optimal solution, in a centralized system one would employ a branch-and-bound or local search -based search engine, which would collect all the information and search the solution space more or less intelligently to find one optimal solution. In our multi-agent system, instead, we keep the agents' information private, and thus we rely on the communication among the agents to find an optimal solution. More precisely:

- Agents communicate in several proposal phases.
- In each phase, a proposal is made by one of the agents (chosen following a certain strategy). Such a proposal is taken from the best time-city slots in the proposer's calendar, and a check is made to make sure that this proposal has not been made previously, and also that it is consistent with the knowledge that the proposing agent has collected about the other agents. More precisely, both negative and positive information about the other agents are used to focus the search on a subset of the possible proposals. We recall that the negative information tell us that a certain agent cannot meet with the others in certain time-city slots. Thus it would be useless to propose such slots. On the other hand, the positive information tell us that a certain agent does not have a meeting in certain time-city slots. Thus such slots are possible candidates for a common meeting, and therefore they can be proposed. The first viable proposal which passes all these tests is communicated to the other agents.
- The other agents, which receive the proposal, reply (to the proposer alone) with a rejection (preference = 0) or an acceptance (and a preference value $\neq 0$). In the meantime, the negative and positive information for each agent (about the other ones) are updated according to the proposal and the answers to it.
- If there is some rejection, a new proposal phase is started. Otherwise, if they all accept, then we have a solution, whose preference value is the minimum among all the agents' preferences for that time-slot.
- At this point, a new negotiation process starts, with the same constraints as before, except that all the preference values which are smaller than or equal to the value of the last solution found are set to 0. This implies that the new negotiation process will necessarily find a better solution, or else it will find a solution with preference 0.
- When the negotiation process ends with a solution with preference 0, and it will certainly happen after a finite number of negotiation processes, we are sure that the last found solution is an optimal one. Thus our algorithm always find an optimal solution in a finite number of steps. Of course, the complexity of this algorithm is, in the worst case, exponential in the size of the problem (number of slots and agents).

We also introduce the concept of a threshold, to model the situation in which one looks for a feasible solution above a certain threshold. In this case, proposals are always chosen among those above the threshold, and answers are rejections whenever the agent receiving the proposal has a preference smaller than the threshold. Thus the threshold is actively used to reduce the number of proposals and thus speeds up the whole process.

We also consider the use of thresholds to find optimal solutions: in this case, the threshold is instrumental to reduce the number of proposals to get to an optimal solution above the threshold, or to state that there is no solution above it. In both cases, when using a threshold, several negotiation processes may be needed to find the solution.

Our system

Our system has been developed in Java, starting from the code of an existing system which solves the same kind of problems but without preferences (Freuder, Minca, & Wallace 2001). The system is available at the URL http://www.studenti.math.unipd.it/~mfranzin.

The system shows a meeting scheduling problem with 3 agents, and allows the user to choose the number of initial meetings the agents have already in their weekly schedule. This number is the same for all the agents.

The interface is shown in Figure 2. In this case there are 15 initial meetings: the number is set via the slider in the lower left corner, and the meetings are chosen randomly among all possible slots, in a way that is consistent.

The upper right panel shows the initial schedule for each of the three agents (agent A in this case, see upper left window). The 15 meetings are represented by higligthed time slots in the calendar. Moreover, another slot is darkened and it represents a common solution for all the agents. In fact, we only generate problems where there is at least one guaranteed solution. Preferences are set randomly by the system over all time-city slots, and can be shown by clicking with the mouse on the calendar time slots. Here we have shown the preferences for the Sunday at 10am time-slot.

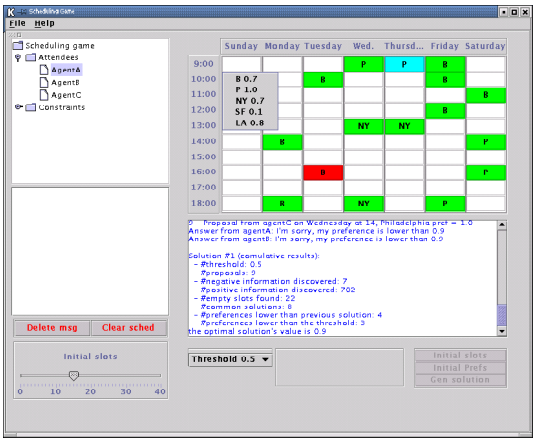

Figure 2: Java interface for our system.

The user can also set a threshold, by clicking on the lower middle button, that will be used to find a feasible or optimal solution above the chosen value. If no value is set, the default value is 0. In Figure 2, the thrshold is set to 0.5.

The lower right window shows the interaction and the information collected during the run of the system. More precisely, we record here number of proposals, positive and negative information, open slots discovered, proposals and corresponding acceptance or rejection, preference value of each solution found, and preference value of the optimal solution.

Experimental choices

Our experiments show the behaviour of the system while the number of initial agents varies from 5 to 40 in steps of 5. We show what happens both while searching for the first feasible solution, and also while searching for an optimal solution.

In each experiment, we show the figures related to one test run, which consists of different things depending on whether we are looking for

- a feasible solution: here a run is a series of proposals, until a proposal which is accepted by all agents;
- an optimal solution: here a run consists of several sequences of proposals, until no better solution can be found;
- a feasible solution above a certain threshold: here a run is a sequence of proposal which lead to the first solution above the threshold;
- an optimal solution above a certain threshold: here a run is several sequences of proposal, which lead to an optimal solution above the threshold.

At each step, a proposal is made by one of the agents, chosen following a round-robin strategy. Such a proposal is taken from the best time-city slots in the proposer's calendar, and a check is made to make sure that this proposal has not been made previously, and also that it is consistent with the knowledge that the proposing agent has collected about the other agents (as explained in the negotiation section). The first viable proposal which passes all these tests is communicated to the other two agents, each of which replies (to the proposer alone) with a rejection (preference = 0) or an acceptance (and a preference value $\neq 0$).

In these experiments, all the figures on the y axis are averaged over 100 experimental runs with the same parameters. Efficiency is measured in terms of number of proposals during a test run. Privacy loss is measured via the positive information, the negative information, the open slots discovered during a run. These privacy loss measures were divided by 6, which is the number of communication links in a system of three agents ($n \times (n-1)$ in a system with n agents).

Experimental results

Figure 3 shows the privacy loss measures (open slots, negative information, and positive information) computed during the search for a feasible solution.

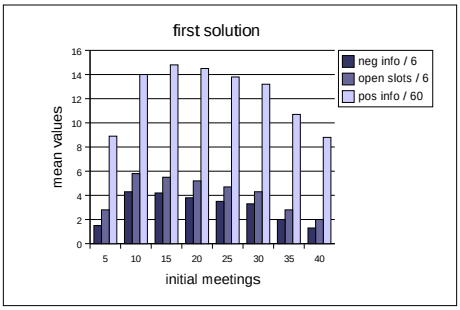

Figure 3: Privacy loss to find a feasible solution.

It is easy to see a curvilinear relation of these measures over the number of initial meetings. More precisely, with a small number of initial meetings, the problem has more feasible solutions and thus little privacy loss is required to find a solution. On the other hand, with a large number of initial meetings, there are many conflicting situations, which reduce the number of common solutions and thus also the possibilities for information exchange, which are mainly related to proposal acceptance. So again little privacy loss is needed to find a feasible solution. This behaviour is very similar to what was computed for the system in (Freuder, Minca, & Wallace 2001), and the main reason is that preferences are not involved when looking for any feasible solution.

When instead we look for a feasible solution above a certain threshold, the peak in the privacy loss measures, which is around 15 initial meetings without threshold, gets shifted to the left (see Figures 4 and 5). The reason is that, even with few initial meetings, the threshold generates more rejections and in general more interaction to find a feasible solution. On the other hand, with a large number of initial meetings, we get less privacy loss, since less proposals (and thus less interaction) can be performed because of unacceptable timeslots (that is, with a preference below the threshold).

When we look for an optimal solution, the relation between number of initial meetings and privacy loss changes

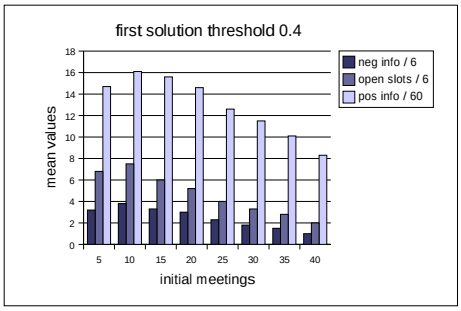

Figure 4: Privacy loss to find a feasible solution with threshold 0.4.

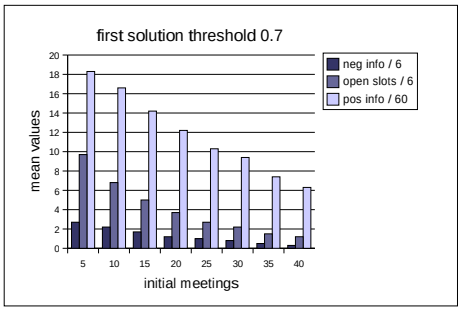

Figure 5: Privacy loss to find a feasible solution with threshold 0.7.

substantially (see Figure 6): with smaller numbers of initial meetings we need to exchange a larger amount of private information to find an optimal solution. In fact, the presence of many solutions when there are few initial meetings does not help. Figure 6 shows also another information: the number of rejections which depend on the fact that the previous solution found has a preference value higher than, or equal to, the preference of the proposed slot for a receiving agent. This number, together with the negative information bar, shows the number of rejections.

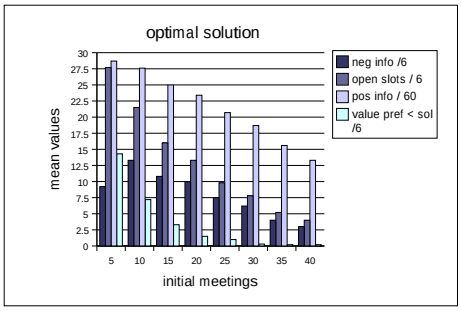

Figure 6: Privacy loss to find an optimal solution.

Figure 7 shows the number of proposals and the preference values of the first and best solutions found. As the number of meetings grows, the difference between the preference value of the first and the optimal solution gets smaller. The same holds also for the difference between the number of proposals needed to find such solutions. However, it can be noted that this difference remains substantial, while the preference values are almost the same with a large number of meetings. This means that, if we have a large enough number of initial meetings (say 25 or more), we can get a very good approximation of an optimal solution by taking any feasible solution, and saving much time (that is, number of proposals). This is probably due to the small number of common solutions when the agents have many initial meetings.

It is also interesting to notice that the efficiency of the system (in terms of number of proposals) when looking for a first feasible solution, is practically the same as in the system described in (Freuder, Minca, & Wallace 2001), where preferences are not used. Thus the generalization to a preference-based system does not slow down the search for a feasible solution.

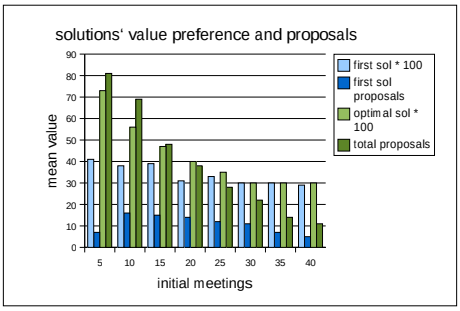

Figure 7: Number of proposals and preference values for first and optimal solutions.

The analysis done over Figure 7 can be done also with the setting of a threshold. In this case, the threshold helps in finding either a feasible solution or an optimal solution in a faster way. Figures 8 and 9 show the difference in preference value and in number of proposals for the case of threshold 0.4 and 0.7. We can see that, as the threshold gets higher, it is more and more convenient to search for a feasible solution even if we care for an optimal one, since the difference in preference value is very small while the number of proposals is much smaller.

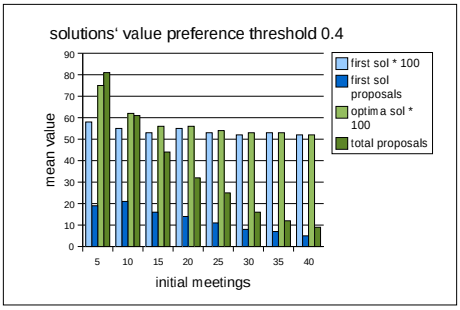

Figure 8: Number of proposals and preference values for first and optimal solutions with threshold 0.4.

However, one needs to be careful with this, because the

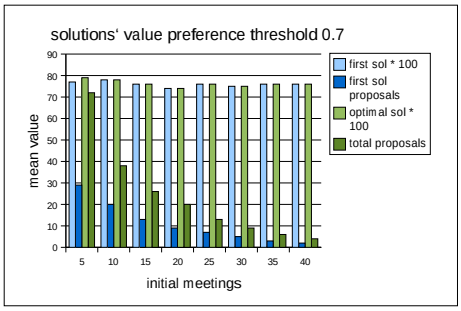

Figure 9: Number of proposals and preference values for first and optimal solutions with threshold 0.7.

number of unsolvable problems grows with the threshold, as shown in Figure 10, which refers to the problem of finding an optimal solution. Thus it is risky to set a high threshold. Notice also how the number of proposals and the privacy loss measures decrease as the threshold increases. For this figure, we have set the number of initial meetings to 25. Figure 11 shows a similar graph for the case of searching for a feasible solution.

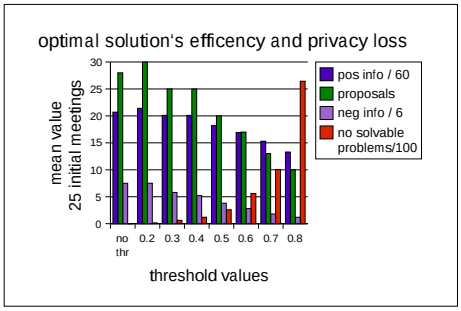

Figure 10: Number of proposals, positive and negative information, and number of unsolvable problems, to find an optimal solution, with 25 initial meetings.

Conclusions and further work

We have implemented a multi-agent scheduling meeting system with preferences, and we have run experiments to study the relations among privacy loss, efficiency, and solution quality. The main lesson we have learnt from such experiments concerns the fact that both the number of initial meetings and the threshold influence the level of such measures. In particular, for example, when agents have already a large number of initial meetings, searching for a feasible solution will give us a very good approximation of an optimal solution in a shorter time (w.r.t. looking for an optimal solution). Moreover, it is important to notice that the search for a feasible solution is not slowed down by the addition of the preferences.

Many extensions of the current system are possible. For example, in the current system all agents have the same optimization criterion (max-min), which coincides also with the overall one. In many real-life scenario, on the other hand,

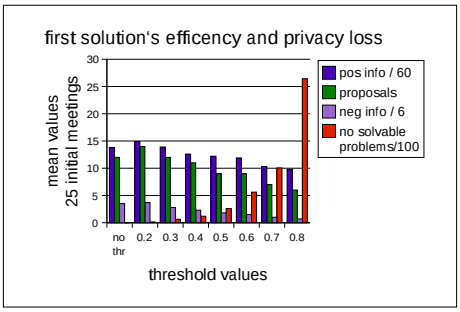

Figure 11: Number of proposals, positive and negative information, and number of unsolvable problems, to find a feasible solution, with 25 initial meetings.

it is possible to have agents with different criteria and also to combine their work via another criterion. We could also allow for a variable number of agents. Moreover, also the distance constraints (rather than just the slots) could have preferences or importance levels. This could allow for example to express optimization criteria like the desire to minimize the overall travel time for an agent.

Also, it would be interesting to address issues like liars (agents who respond to a proposal with a lie) or proposal rejections which include reasons for the rejections (like other meetings) or counter-proposals.

Acknowledgments

We would like to thank Marius Minca, who developed the multi-agent meeting scheduling system described in (Freuder, Minca, & Wallace 2001), for providing the code from which we started to implement our system, and for many useful discussions about the behaviour of his system. Eugene Freuder is supported by a Principal Investigator award from Science Foundation Ireland.

References

Bistarelli, S.; Montanari, U.; and Rossi, F. March 1997. Semiring-based Constraint Solving and Optimization. *Journal of the ACM* 44(2):201–236.

Dubois, D.; Fargier, H.; and Prade, H. 1993. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In *Proc. IEEE International Conference on Fuzzy Systems*, 1131–1136. IEEE.

Freuder, E. C.; Minca, M.; and Wallace, R. J. 2001. Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents. In *IJCAI-01 Workshop on Distributed Constraint Reasoning*.

Garrido, L., and Sycara, K. 1995. Multi-agent meeting scheduling: Preliminary experimental results. In Lesser, V., ed., *Proceedings of the First International Conference on Multi–Agent Systems*. MIT Press.

Luo, X.; Leung, H.; and Lee, J. H. 2000. Theory and properties of a selfish protocol for multi-agent meeting scheduling using fuzzy constraints. In *Proc. ECAI 2000*.

Ruttkay, Z. 1994. Fuzzy constraint satisfaction. In *Proc. 3rd IEEE International Conference on Fuzzy Systems*, 1263–1268.

Schiex, T. 1992. Possibilistic constraint satisfaction problems, or "how to handle soft constraints?". In *Proc. 8th Conf. of Uncertainty in AI*, 269–275.

Scott, A.; Jenkin, K.; and Senjen, R. 1988. Design of an agent-based, multi-user scheduling implementation. In *Proc. 4th Australian DAI Workshop, Brisbane*. Springer LNAI 1544.

Sen, S.; Haynes, T.; and Arora, N. 1997. Satisfying user preferences while negotiating meetings. *Internat. J. Human-Computer Stud.* 47:407–427.

Tsang, E. P. K. 1993. *Foundations of Constraint Satisfaction*. Academic Press.

Tsuruta, T., and Shintani, T. 2000. Scheduling meetings using distributed valued constraint satisfaction algorithm. In *Proc. ECAI 2000*.

Wallace, R. J.; Freuder, E. C.; and Minca, M. 2002. Consistency reasoning and privacy/efficiency relations in multi-agent systems. In *submitted for publication*.