Opportunities for Learning in Multi-Agent Meeting Scheduling*

Elisabeth Crawford and **Manuela Veloso**
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh PA, 15232
ehc+,mmv+@cs.cmu.edu

Abstract

In this paper we explore opportunities for learning in Multi-Agent Meeting Scheduling. We view this multi-agent task as fully distributed with several challenging characteristics: (i) agents have ownership of their own calendars; (ii) agents exchange information among each other with the goal of finding an open meeting time; (iii) agents can negotiate multiple meetings concurrently. We have implemented a negotiation strategy in which the agents communicate all their available time slots. This "open negotiator" is designed to reflect the open calendar approach of Microsoft Outlook in a distributed setup. We show where this negotiation strategy fails to lead to efficient scheduling and discuss how agents could use learned information to improve social welfare and scheduling performance.

Introduction

The Multi-Agent Meeting Scheduling problem (MAMS) presents a number of significant challenges, including the online modeling of user preferences and the satisfaction of these preferences through efficient scheduling. For the purposes of this paper we assume that each user in an organisation's computer system has a personal scheduling agent that knows their preferences and the contents of their calendar. This agent has the ability to communicate with the agents of other users to negotiate times for new meetings and move existing meetings. The agents negotiate with each other by exchanging messages about meetings e.g messages proposing meeting times. A meeting under negotiation is confirmed and entered into the calendar of each user when the agents all agree on a particular time.

To communicate effectively the agents need to adhere to some protocol. Compliance with a protocol, ensures that the agents can understand the meaning of all the messages they receive, and can prevent communication deadlock.

*This research is sponsored by the Department of the Interior (DOI) - National Business Center (NBC) and the Defense Advanced Research Projects Agency (DARPA) under contract no. NBCHC030029. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the DOI, NBC, DARPA or the U.S. government.

A significant feature of the MAMS problem is its distributed nature. At any particular point in time, only a user's own agent can claim they have accurate knowledge of the user's preferences and calendar. Even for example, if one agent revealed to another its user's entire calendar, by the time the message is received by the second agent the information may be out of date. This is because an agent may be negotiating multiple meetings simultaneously. A major challenge in designing effective agents is handling this inherent uncertainty so that negotiation is efficient.

An alternative formulation of the meeting scheduling problem is used by Microsoft Outlook. Outlook largely ignores the negotiation step and issues of uncertainty about other users' calendars. The scheduling features in Outlook, rely largely on an open calendar system, where users are required to make their calendars publicly viewable within the organisation. When a person wants to schedule a meeting, they indicate the proposed attendees, and the resources required. Using the publicly available information Outlook proposes to the meeting initiator the first available time slot. If the initiating user accepts, a scheduling message is sent to the Outlook clients of the attendees. At some stage after receiving the message, each of the attendees hopefully responds to the meeting request. The initiating user can view the responses and then decide how to proceed. There are a number of problems with this approach. Consider for instance, the situation where a user wants to attend a meeting but refuses the time (Outlook also allows users to make counter-proposals). This may happen when although the proposed time was marked free in the user's calendar when the proposal was sent, the user has received an earlier request for this time by another meeting. By the time the user processes the new proposal the slot is no longer free. A user may also refuse a time because it is inconvenient. Since users have not been given the opportunity to express detailed preferences about when they would like the meeting to occur it is quite possible that an unfavorable time will be chosen.

Furthermore, a major limitation of Outlook is that it will not consider moving existing meetings on behalf of the user. In many cases, it may only be possible to schedule very large meetings when many users move smaller meetings. By dealing with these issues of uncertainty, rescheduling and preference, negotiation agents can potentially schedule more

meetings for better times.

In the next section of this paper we briefly outline previous work on the MAMS problem. In section 3 we describe a protocol for negotiation between agents and in section 4 we detail an implementation of this protocol where agents are willing to share their calendars. In section 5 we show results from the simulation of this approach and explain where the opportunities for multi-agent learning lie.

Previous Work

Previous work on MAMS can be grouped into negotiation based approaches and market based approaches. The negotiation based approaches in general assume a degree of compliance with protocols and procedures (Sen & Durfee 1998; Jennings & Jackson 1995; Gonzlez & Santos ; Garrido & Sycara 1995) . Agents may have some capacity to act with self-interest but the issue of rogue or malicious agents is not addressed. On the other hand, market based approaches assume agents act with self-interest and attempt to implement social welfare maximising mechanisms (Ephrati, Zlotkin, & Rosenschein 1994). We will briefly look at the different approaches in turn.

Sen and Durfee (1998) conducted a probabilistic and simulation based analysis of negotiation strategies. The basic framework they considered was:

1. Host announces meeting
2. Host offers some times
3. Agents send host some availability information
4. Repeat 2 and 3 until intersection found.

Similar protocols have been looked at by other researchers, for example, (Jennings & Jackson 1995) and (Gonzlez & Santos), while (Garrido & Sycara 1995) look at a more complex protocol.

Sen and Durfee (Sen & Durfee 1998) consider a variety of approaches to negotiation including:

- Host announces meeting and offers her best time.
- Host announces meeting and offers a number of times.
- Invitee just responds with a yes or no to times received.
- Invitee proposes alternatives.
- Times under consideration are reserved (ie blocked from being offered in concurrent negotiations).
- Times under consideration are not reserved.

Sen and Durfee's (1998) analysis is centered around determining under what conditions the different approaches are most efficient. The notion of efficiency used is how many iterations of the protocol are required and how many *important* meetings are scheduled. It is worth noting here, that many other notions of efficiency are valid in this domain. For instance, the percentage of meetings successfully scheduled, and how much the agents like the schedules are both useful metrics.

Sen and Durfee (1998) found that the efficiency of the various approaches, depended on aspects of the system state, such as how busy the agents were and how many meetings were being scheduled at once. The purpose of Sen and Durfee's study was to develop heuristics for making good negotiation decisions. Such heuristics form a very useful first step toward efficient distributed scheduling. However, to handle the large variety of situations and impacting factors it is necessary to combine heuristics with learned information. In section 5 of this paper we will discuss some ways in which learning can be used to improve scheduling efficiency.

Sen and Durfee view the problem of MAMS as a distributed (and collaborative) search process in the presence of soft and hard constraints. In contrast Ephrati, Zlotkin and Rosenschein (Ephrati, Zlotkin, & Rosenschein 1994), assume agents to be selfish, and strategic in their actions. The authors propose a market-based approach that attempts to extract each agents' true preferences about meeting times. They create a centralized market of *convenience points* which the agents use to place bids for meeting times. The system imposes a Clarke Tax (Clarke 1971) on the agents in an attempt to ensure they always tell the truth (ie the that the system is incentive compatible). Consideration is given to a number of different variations of the mechanism according to when points are handed out and when in the process preferences are expressed, e.g when a new meeting comes in or at the start of a time period.

The appeal of a mechanism design approach is the possibility of maximising social welfare by extracting the true preferences of the agents and selecting the best time. Unfortunately, Ephrati,Zlotkin and Rosenschein's (Ephrati, Zlotkin, & Rosenschein 1994) mechanism fails to guarantee that agents are best off revealing their true preferences (Crawford & Veloso 2004). The complexity of MAMS makes the design of an effective mechanism a very difficult task.

A common criticism of both negotiation and market-based approaches is that they often complicate what is already a hard problem (efficient scheduling), by worrying about issues like self-interested agents and maximising social welfare. After all, open calendar systems like Outlook are deployed in many organisations. These systems feature a lack of privacy for users and very limited ability to express preferences about times, but the possibility remains that open calendar style systems can make scheduling more tractable.

In section 4 we describe a negotiation strategy for MAMS intended to reflect this open calendar approach. We show that such an approach in many instances fails to lead to efficient scheduling. In section 5 we look at how learning more complex strategies has the potential to both improve social welfare and efficiency.

Protocol

In this section we outline a general protocol for meeting scheduling. The protocol enables multiple agents to negotiate meeting times, come to agreements, and move existing meetings in a distributed environment where many negotiations may be running concurrently.

Outline of the Protocol

- To *initiate a meeting* an agent sends a message to each attendant, with some times marked as POSSIBLE.
- When an attendee agent receives an ordinary message about a meeting it marks some times as POSSIBLE and sends to the initiator.
- To *request that a meeting be fixed* for a certain time, the initiator marks that time as PENDING and sends to the attendees.
- When an attendee agent receives a message with a PENDING time slot, it accepts by marking the time as PENDING in the reply and rejects by marking the time in any other way.
- To *confirm a meeting*, the initiator first checks that all the attendees have marked the time as PENDING. The initiator then marks this time as CONFIRMED and sends to the attendees. The initiator adds the meeting to the calendar.
- When a message is received with a CONFIRMED time slot, the attendee agent checks that it previously marked the slot as PENDING. If so, the attendee updates the status to CONFIRMED and adds the meeting to the calendar.
- To *recall a request* for a time to be marked as PENDING (e.g because one of the attendee's agents refused the PENDING request), the initiator marks the time as IMPOSSIBLE and sends to the attendees.
- To *cancel a meeting*, the initiator marks the previously CONFIRMED time slot as IMPOSSIBLE and sends to the attendees. The meeting is removed from the user's calendar and assigned for rescheduling.
- When an attendee agent receives a message where the previously CONFIRMED time is marked impossible, it removes the meeting from the calendar.
- To *request a meeting be canceled*, an attendee agent marks the CONFIRMED time as IMPOSSIBLE and sends to the meeting initiator.

We also have the following rule - a time may only ever be marked as PENDING by an agent for one meeting at once.

This protocol allows a large variety of negotiation strategies. For instance, agents are free to offer any number of times that they like, they may offer times for more than one meeting at once, or instead reserve offered times, and they can cancel any meeting they choose. To make the protocol robust against agents failing to reply, timeouts can very simply be added. For example, if an initiator does not hear from a particular agent in regards to a PENDING request, they can resend the request after a certain period of time and eventually just timeout assuming the agent is down.

In the next section we present a negotiation strategy that implements this protocol.

Open-Negotiator

The Open-Negotiator is designed to reflect the open calendar style approach seen in Microsoft Outlook, but in a fully automated and distributed setting. Unlike Outlook the Open-Negotiator approach has automatic support for moving meetings, but this is done reluctantly. The intuition behind this approach is that providing a lot of information about availability should lead to more efficient scheduling. And further, that moving meetings should be avoided, since it is time consuming and disruptive to the user's calendar. We will see in the next section, however, that this approach has significant drawbacks in some situations.

The Open-Negotiator approach can be summarized as follows:

- When initiating a meeting the agent marks *all* its available times as POSSIBLE.
- When an attendee agent receives a message about a meeting for the first time, it marks *all* the available times as POSSIBLE.
- When the initiator receives a message about a meeting it computes the intersection between the POSSIBLE slots in this message and the POSSIBLE slots in the offers collected so far. If the intersection is empty, the initiator marks a slot corresponding to an available time in their calendar as POSSIBLE if there is such a slot that has not already been offered. If there is no such slot, the initiator selects the best *unavailable* slot (slot containing a meeting) and marks it as POSSIBLE.
- When an attendee receives a message about a meeting that is not a special message or the first message, it marks an available slot as POSSIBLE if there exists such an unoffered slot, otherwise it marks an *unavailable* slot as POSSIBLE.
- When the initiator of the meeting finds an intersection in the POSSIBLE slots of attendees and all attendees have sent at least one offer, the initiator marks its favourite time in the intersection as PENDING.
- When an attendee receives a message with a PENDING time slot, if this slot is not PENDING for another meeting, and all available slots have been offered, the attendee accepts and marks the slot as PENDING. Otherwise the attendee rejects and marks one more time as POSSIBLE.
- When an initiator has newly CONFIRMED a meeting for some time, if there is another meeting already scheduled for this time, the initiator marks the time slot as IMPOSSIBLE for the original meeting and sends this to the initiator - unless the agent is the initiator of both meetings, in which case it proceeds with the meeting canceling procedure, then re-initiates the meeting.
- When an attendee marks a time for some meeting as confirmed, if there is another meeting already scheduled for this time, the agent marks the time slot as IMPOSSIBLE and sends a message to the initiator of this meeting. If the initiator is this agent, it simply proceeds with the meeting canceling procedure and then re-initiates.

A problem with this approach is that it is only the initiator of the meeting that gets to express any preference. If there is more than one time in the intersection of offers then the initiator chooses the time they favour the most. Since the attendees are offering all their available times at once -

they are not expressing a preference. By offering times bit by an attendee can express their preference by giving their favoured times first. Despite this problem, at this point what we are trying to gauge is the efficiency of the simple open-calendar style approach and where learning can lead to improvements.

Experiments

We have designed three different experiments using the Open-Negotiator. All the experiments are run over a calendar containing 50 time slots - 10 slots per each of the days of the working week. In each experiment we randomly generated preferences for 20 different agents. The preferences are modeled very simply. There is a morning, midday and afternoon profile according to what part of the day the agent prefers. The part of the day an agent prefers is selected at random. The preferences for that part of the day are randomly selected values from the upper end of the preference range. For instance, if the chosen preference was morning, the values for morning times are selected from the upper end of the preference range, values for midday times from the middle of the range, and values for the afternoon times from the lower end of the range. In this way, although we only have three main types of preferences, it is nonetheless not very likely that two agents have the exact same preference profile.

The first experiment, looks at the effect of the density of the agents starting schedules on the time to negotiate new meetings. We randomly generated a set of 20 new meetings that will be the subject of negotiation in each trial. The set contains one meeting with all 20 agents attending, one 4 agent meeting, three 3 agent meetings and the rest of the meetings have just 2 participants.

We looked at a variety of schedule densities. We randomly generated meeting sets with sizes ranging from 10 meetings up to 220. Each set contains at least one meeting with all participants, some large meetings and many two, three and four person meetings. The number of meetings of each size is determined by the same ratios for each meeting set size. Given the initial meeting set, we randomly schedule the meetings in each agents empty calendar to arrive at the starting calendar configuration. We ran 10 trials for each initial meeting set size. From these starting calendars the agents must then schedule the 20 new meetings. Thus, we can examine the effect calendar density has on the time to schedule new meetings.

The second experiment, looks at how scheduling time, relates to the number of meetings that are up for scheduling at once. We are interested to see how the efficiency of the protocol is effected by the number of meetings under negotiation at the one time. In this experiment we start with a set of 100 meetings scheduled at random in the agent's calendars. We then schedule different sized sets of meetings from this starting density, ranging in number from 5 up to 160.

Finally, the third experiment examines the effect of meeting size on time to schedule. A fairly full initial density of 200 meetings is used. We then test how long it takes to schedule a 2 person meeting, a 3 person meeting, and so forth all the way up to a meeting with all 20 agents.

For each setting of the three experiments above we perform a number of trials. This is particularly necessary since we only fix the starting densities and meetings, not the initial calendars. This means that in two different trials the same meetings may be scheduled for different times. This is useful because it gives us a notion of the variance between different calendar configurations, which, in itself, suggests ways in which learning may help efficiency.

Results and Opportunities for Learning

Effect of Schedule Density

Figure 1 shows the time it took to schedule the set of new meetings, from different initial schedule densities. The x-axis represents the number of meetings already scheduled in the system when the simulation starts, and the y-axis, the time taken to the schedule the twenty new meetings. Figure 2 shows the effect schedule density has on the total number of messages sent by agents during the simulation.

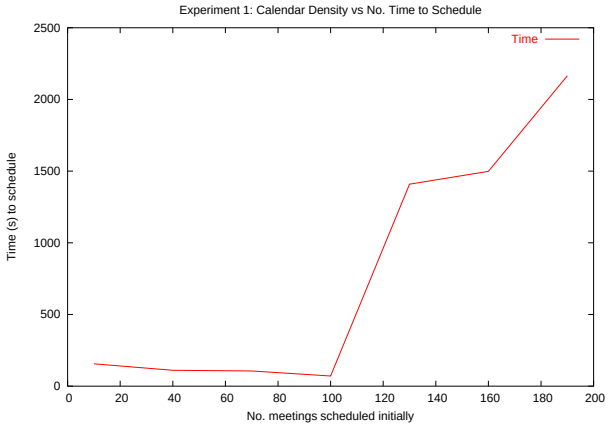

Figure 1: Experiment 1 - Time to schedule 20 meetings from different initial calendar densities

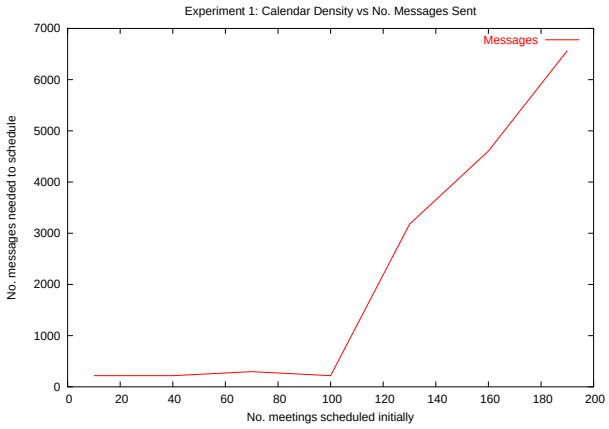

Figure 2: Experiment 1 - No messages sent in scheduling 20 meetings from different initial calendar densities

When comparing Figures 1 and 2 we can see that while

the number of messages needed to schedule the 20 new meetings is fairly constant for densities between 10 and 100, the *time to schedule actually falls as the density increases*. The reason for this, is that for these low densities, all the 20 meetings can be scheduled without moving any of the original meetings. Thus, the protocol is able to in each instance, with roughly the same number of messages, determine common free slots for each meeting. There is some variation in the number of messages due to PENDING requests being refused because another meeting has already been made PENDING for the time requested. But this does not have a large effect.

Although for initial densities between 10 and 100 meetings the number of messages needed to be sent is the same - the size of these messages varies greatly. When there are only 10 meetings in the starting schedule, each of the agents is sending almost their entire schedule. They each have to iterate over all the time slots marking them as POSSIBLE. The initiator then has to calculate the entire intersection of responses. This intersection which is likely to be large, is then sorted by the initiator and finally a time is chosen. Clearly, this process takes longer the more time slots are open. As such, the time taken falls until the density reaches a point where the new meetings cannot be scheduled without moving some of the original meetings.

Both Figures show a sharp increase at the 120 density mark. This is because, for these densities, to schedule the larger meetings some existing meetings have to be moved. The bulk of the time, and the messages, are spent on scheduling the 20 person meeting. Due to the fact that the negotiation strategy only offered one new *unavailable* time in any one message, many messages needed to be sent before the initiator found a time in the intersection of all offers. Once this time was found, moving the existing meetings proved quick, since the meetings canceled typically had few participants.

For the small initial meeting set sizes the variation between trials was low. However, the variation increased with the size of the initial meeting set and became quite significant for sizes greater than 100. This indicates how greatly the system state (in terms of the calendars of the agents) can effect scheduling efficiency.

Opportunities for Learning

There are clearly a number of ways we could use learning to improve the negotiation strategy. Consider for instance, the range of calendar densities where the time to schedule actually falls as the density increases. We would like for agents to take advantage of the tractability of the problem in these instances to better satisfy their preferences. By learning *indicators* for when scheduling meetings is likely to be straightforward, an agent could chose to act more strategically at times without impacting too greatly on efficiency. Instead of simply offering all their available times at once, an agent may to choose to offer subsets of these times in order of preference. Given that agents learn appropriate numbers of times to offer, we could see an increase in social welfare, while maintaining efficiency. Learning *indicators* for the state of the system as a whole, involves each agent reasoning about the calendars of the other agents. However since many agents would interact with each other relatively infrequently other features e.g organisational hierarchies and user input may be needed.

Learning indicators of when scheduling is likely to be hard would also be very useful. According to how difficult the agent believed it was going to be to find a time slot, it could decide to offer different numbers of unavailable times per message - as opposed to just one. This could dramatically increase efficiency provided the agents choose the unavailable times to offer wisely. In particular, it would be very useful for the agents to learn what sorts of meetings are easy to reschedule. This is not just in terms of whether the meeting is large or small, but also according to what the agent has learned about the attendees of the meeting.

Effect of Scheduling Many Meetings at Once

Figure 3 demonstrates the effect scheduling different numbers of meetings, has on the time taken. The x-axis shows the number of new meetings scheduled by the system, while the y-axis shows the number of seconds it took to do the scheduling. The curve gets steeper as the number of meetings grows, indicating the increasing complexity. Figure 4 on the other hand shows a fairly uniform increase in the number of messages required to do the scheduling. What this indicates is that the increase in steepness of the time curve is largely caused, not by there being many meetings scheduled at once - but simply the sheer number of meetings in the calendar. Recall that to begin with there are 100 meetings in the system, thus when the agents are trying to schedule 120 meetings, there are going to be 220 meetings amongst only 20 agents competing for time slots. The steady increase in the number of messages sent, indicates that the negotiation strategy is coping with more meetings being scheduled simultaneously. The danger, is that as more meetings are being scheduled concurrently, more PENDING requests will have to be recalled, due to different meetings trying to reserve the same time. However the results reveal this to be not a significant issue.

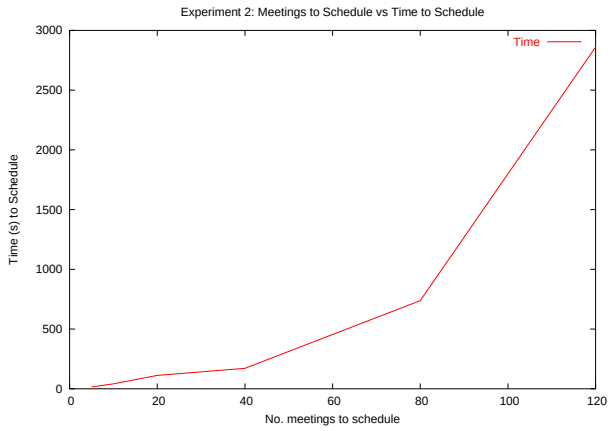

Figure 3: Experiment 2: Time taken to schedule different numbers of meetings from a starting density of 100

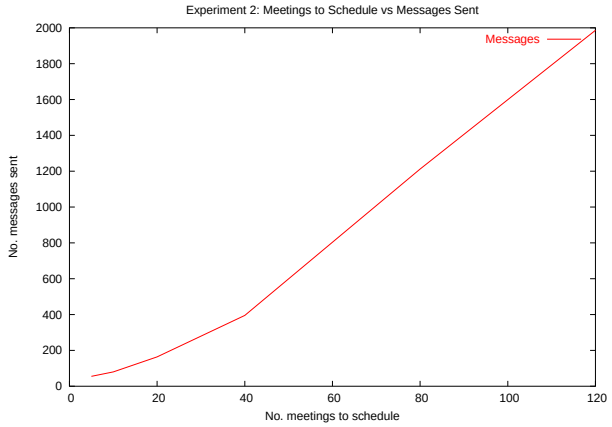

Figure 4: Experiment 2: No messages sent when scheduling different numbers of meetings from starting density of 100

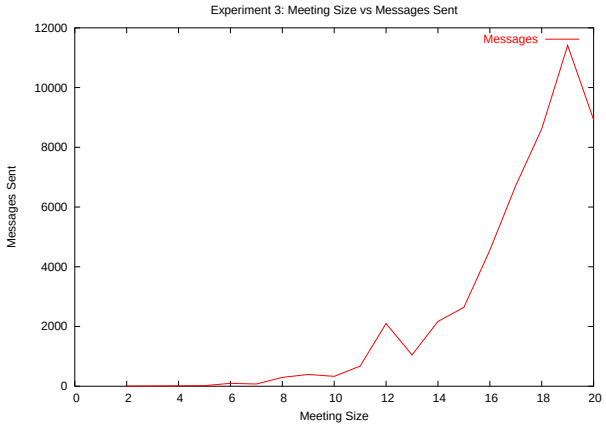

Figure 6: Experiment 3: No messages sent when scheduling different sized meetings from starting density of 100

Effect of Meeting Size

Figures 5 and 6 show the effect meeting size has on the efficiency of scheduling. The x axis represents the size of the meeting in terms of the number of agents attending, while the y axis is respectively the time to schedule the meeting and the number of messages sent to schedule the meeting. While we observe that there is a general up-wards trend, in the time and number of messages required, we also observe a great deal of variation. We saw both variation in the different trials for the same meeting, as well as variations between meeting sizes. For instance, on average, meetings of size 13 were scheduled faster than meetings of size 12. While these variations could be smoothed out by doing a very large number of trials, it is interesting to note that the exact configuration of the initial calendar has a huge effect on the efficiency. Note, for example, that it took on average less messages to schedule the meeting of size 20, than size 19. This is largely due to the differences in the configurations of the initial calendars.

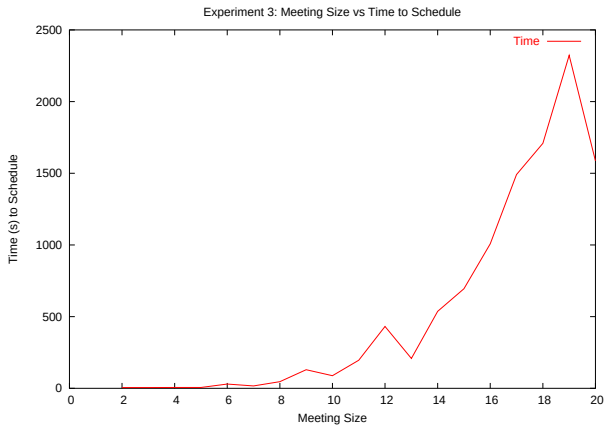

Figure 5: Experiment 3: Time taken to schedule different sized meetings, starting from a density of 200

Opportunities for Learning

The large impact the configuration of the initial calendar can have indicates an interesting opportunity for learning to increase efficiency. While it is relatively simple to learn what sized meetings on average are hard to schedule, learning how to build and use models of the system state is hard. As mentioned before, a major challenge is the lack of examples any one agent has to draw conclusions from, further to this, there is the issue of what features to choose to look at and how to combine information learned from heterogeneous sources.

Conclusions

In this paper we have described a flexible protocol for MAMS. Using this protocol we have studied the usefulness of a negotiation strategy that is both open about availability, and reluctant to move already scheduled meetings. The strategy was designed to reflect the approach taken by Microsoft Outlook, which, despite its drawbacks, is in use in many organisations.

We found that the Open-Negotiator approach worked very effectively when there was a common free time slot. However, this is the situation when the problem is most tractable in general, and the open approach takes no advantage of this. By learning to recognise when scheduling is likely to be tractable, and by learning appropriate numbers of slots to offer in negotiation messages, agents can increase the likelihood of getting the meeting scheduled at a time they prefer, while still maintaining efficiency.

The open negotiation strategy performed poorly when there was no common free time slot. The reluctant approach to offering unavailable times, lead to slow scheduling of meetings with many participants. However, too eager an approach to moving meetings could also be very inefficient, possibly leading to many meetings being moved in a chain, to accommodate just one new meeting. Clearly, agents need to be able to recognise which meetings are the best candidates for moving, and how cautious they should be when

offering unavailable times. We would like agents to be able to use learning to help make these decisions.

MAMS is a complex problem that provides many challenges for Multi-Agent Learning. In the future, we would like to give agents the ability to make inferences about the state of the system and other agents calendars from few negotiation interactions and heterogeneous sources, such as hierarchies, groups, and other associations between people (learned or given).

References

Clarke, H. E. 1971. Multi-part pricing of public goods. *Public Choice* 11:17–33.

Crawford, E., and Veloso, M. 2004. Mechanism design for multi-agent meeting scheduling, including time preferences, availability and value of presence.

Ephrati, E.; Zlotkin, G.; and Rosenschein, J. S. 1994. A non–manipulable meeting scheduling system. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*.

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 (ICMAS'95)*. The MIT Press: Cambridge, MA, USA.

Gonzlez, A., and Santos, J. A negotiation protocol for meeting scheduling based on a multiagent system.

Jennings, N. R., and Jackson, A. J. 1995. Agent based meeting scheduling: A design and implementation. *IEE Electronics Letters* 31(5):350–352.

Sen, S., and Durfee, E. H. 1998. A formal study of distributed meeting scheduling. *Group Decision and Negotiation* 7:265–289.