Learning to Select Negotiation Strategies in Multi-Agent Meeting Scheduling

Elisabeth Crawford and Manuela Veloso *

Computer Science Department,
Carnegie Mellon University,
Pittsburgh PA 15213, USA
{ehc,mmv}@cs.cmu.edu

Abstract. In this paper, we look at the Multi-Agent Meeting Scheduling problem where distributed agents negotiate meeting times on behalf of their users. While many negotiation approaches have been proposed for scheduling meetings, it is not well understood how agents can negotiate strategically in order to maximize their users' utility. To negotiate strategically, agents need to learn to pick good strategies for negotiating with other agents. We show how the *playbook* approach, introduced by [1] for team plan selection in small-size robot soccer, can be used to select strategies. Selecting strategies in this way gives some theoretical guarantees about regret. We also show experimental results demonstrating the effectiveness of the approach.

1 Introduction

Personalized software agents for meeting scheduling have the potential to reduce the daily cognitive load on computer users. Scheduling meetings can be a time consuming process requiring many email messages to be exchanged, and often existing meetings need to be moved to make room for new ones. Potentially, software agents can remove this burden entirely by communicating with each other to schedule meetings. Since user's have ownership of their own calendars, and private preferences about meeting scheduling, it makes sense to approach this problem in a distributed manner. Automated negotiation has been proposed as a method for multiple agents to reach agreement on meeting times. Negotiation approaches have many advantages over the open calendar approach taken by Microsoft Outlook (see [2] for a discussion).

Typically negotiation protocols feature a meeting initiator that proposes meeting times and collects the proposals of other participants. Consider, for instance, the following simplified protocol:

– while there is no intersection in proposals

* Thanks to the reviewers and Michael Bowling for helpful comments and suggestions. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

- the initiator proposes some times to the other agents
- each agent proposes some times to the initiator

In this context, a negotiation strategy is a set of rules for deciding what times to propose at each point in the process. The space of possible negotiation strategies is extremely large. Even if we restrict the space in some way, e.g. to strategies that offer a fixed number, x, of new times per negotiation round, there are still a huge number of options. In particular, there is a different strategy for each possible value of x, and then there are all the ways of combining these values of x with rules for deciding what particular times to offer. In developing software agents for meeting scheduling, we are faced with the problem of: (i) deciding which negotiation strategies agents should consider, and (ii) designing methods that agents can use to choose between these strategies when negotiating a particular meeting.

In order to most effectively satisfy user preferences, we would like our agents to adapt their behavior to each of the agents they negotiate with. There is a wide range of important ways in which agents can differ. For instance, agents can represent users of different importance and busyness, use very different negotiation strategies, and can have users with very different preferences. Clearly a strategy that works well for negotiating with one agent may work very poorly with another. Poor strategy choice can lead to meetings being scheduled at times the user does not like, or to the negotiation process taking a very long time. In general, we would like an agent to trade-off satisfying its user's preferences, with minimizing the length of the negotiations, in a way that maximizes its user's utility.

One method for deciding what strategy to use when negotiating with a particular agent is to use a model based approach that tries to construct a model of the agent and then based on this model select a strategy. There are a number of reasons why this approach would be difficult to use in practice. Firstly, obtaining an accurate enough model of another agent is a very difficult learning problem, since the only interaction agents have is through the exchange of times when they negotiate meetings. From this information, it is hard to make accurate conclusions about what times an agent prefers, how busy the agent is, what negotiation strategy it is employing etc. Secondly, to build a model of another agent, many training examples are required. It would be preferable if an agent was able to learn to negotiate, while actually negotiating.

In this paper, we show how an agent can learn online which strategies to use by observing its own rewards, as opposed to trying to model the other agents. Our approach is based on the idea of *plays* introduced by Bowling, Browning and Veloso [1]. Bowling *et al.* focus on the domain of robot soccer (small-size league) where they equip a team with a series of multi-agent plans called a *playbook*. The team plan to use at a given point in time is selected according to a no-regret learning algorithm. We show how we can apply these ideas to the problem of learning how to negotiate with different agents. Our experimental results demonstrate that this approach allows a learning agent to converge to sensible strategies for negotiation with different fixed strategy agents. We also show that an agent learning online using this approach can perform well in comparison to the best (in hindsight) fixed strategy.

2 Related Work

A variety of methods for reaching agreements on meeting times have been proposed in the last ten years, including negotiation based approaches, e.g. [3, 4], Distributed Constraint Reasoning (DCR) approaches [5], and market based approaches [6]. In this section, we describe work on the first two methods, looking in particular at how user preferences are dealt with.

Sen and Durfee [4] 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 an intersection is found.

Similar protocols have been looked at by other researchers, for example, [3], and [7], while [8] looked at a more complex protocol. These negotiation approaches have handled user preferences for meeting times in quite different ways. Shintani *et al.* [8] propose a persuasion based approach. The persuasion mechanism involves compromising agents adjusting their preferences so that their most preferred times are the persuading agent's most preferred times. This method relies strongly on the agents complying with the protocol.

Garrido and Sycara [7] and Jennings and Jackson [3] take the approach of allowing agents to not only propose meeting times, but also to quantify their preferences for proposals. The agent that is collecting the proposals, then makes decisions based on the reported utilities of all the meeting participants. This style of approach involves a lot of trust, since for the procedure to work well all the agents must report their preferences truthfully.

While the approaches outlined are all concerned with user preferences they differ from the work described here in that we are interested in *how an agent can negotiate strategically in order to satisfy its user's preferences*.

Distributed Constraint Reasoning (DCR) approaches have also been applied to multi-agent meeting scheduling. For example Modi and Veloso [5] model the meeting scheduling problem according to the DCR paradigm and evaluate strategies for making *bumping decisions*. The way in which agent's decide when to *bump* (i.e., move an existing meeting to accommodate a new meeting) can have implications for the efficiency of the meeting scheduling process. Intuitively, if the agents want the scheduling process to finish quickly, they should try to bump meetings that will be easy to reschedule. Similarly to the negotiation approaches, the work on DCR has not focused on how agents can act strategically, rather the agents have been assumed to be cooperative.

3 Plays for Meeting Negotiation

In the context of small-size robot soccer (where an overhead camera and an off-board computer allow for coordinated team planning) Bowling *et al.* [1] introduce the notion of a play as a team plan. Each play must assign a role to each of the robots, e.g. one

robot is instructed to shoot, another to guard the team's own goal, and so forth. Each play also has an applicability condition that determines in which scenarios it applies, and a termination condition that is used to decide when the play finished and a new one needs to be selected. An offensive play, for example, might be applicable whenever the ball is in the opponent's half of the field, and terminate either when a goal is scored, or the applicability condition is violated.

The *playbook*, captures all the plays that are available to the team. Bowling *et al.* provide a simple language that a human expert can use to add new plays. During the course of a game, plays are weighted according to their level of success or failure, and the play to use at each decision point is selected based on these weights. The weights on plays are adapted in such away that regret (difference between how well the team did and how well it could have done had it used the best, in hindsight, fixed play) about play selection goes to zero in the limit.

The meeting negotiation problem has a number of important features in common with small-size robot soccer. In both domains, the space of available strategies is huge. It is not possible for agents to adapt online if they must consider the entire space. Furthermore, the environment in both domains is dynamic, the models of the 'opponents' are unknown, and online learning is required for good performance. In this section, we will discuss how we adapt the plays formulation to the problem of learning how to negotiate with different agents.

We can map the plays terminology, from robot soccer, to the meeting scheduling problem. The plays correspond to complete negotiation strategies, the opponent corresponds to the agent the learning agent is negotiating with, and the playbook is simply the set of negotiation strategies available to the learning agent. Unlike in robot soccer, in the meeting scheduling problem, we are playing with multiple 'opponent' agents at the same time. As such, the learning agent must adapt strategy selection for each of the different agents it negotiates with simultaneously.

We let a negotiation strategy consist of 4 elements: (i) an applicability condition, (ii) a rule for deciding at each negotiation round what times (if any) to offer independent of the exact proposals received, (iii) a rule for deciding which times (if any) to offer based on the proposals received, and (iv) a rule for deciding when to give up[1]. Fig. 1. shows an example strategy, Offer-k-b, that offers k new available times each round, and after b rounds starts taking into account the proposals it has received. If necessary, Offer-k-b will offer times that would require an already scheduled meeting to be bumped. Depending on the values of k and b this strategy can be very selfish and cause the negotiation to take a long time. As such, if the 'opponent' agent is very important, the strategy is only applicable if the value of k is large and the value of b is small.

Each time a new meeting needs to be scheduled, if the learning agent is an attendee, it selects which strategy to use according to the adapted playbook for the initiator agent. If the learning agent is initiating the meeting, it selects a possibly different negotiation strategy for communicating with each attendee according to the adapted playbook for that attendee. The learning agent considers the execution of a strategy to be complete when (i) the meeting it was selected to schedule has been added to the agent's calendar and (ii) any meetings that the learning agent is involved in that have been bumped

[1] The distinction between (ii) and (iii) is unnecessary. It is made to assist the exposition.

1. **APPLICABILITY**: if the other agent is very-important and ($k < 20$ and $b > 5$) return false; else return true.
2. **INDEPENDENT OFFER**: in any negotiation round offer my k most preferred, available, un-offered times.
3. **DEPENDENT OFFER**: if negotiation round $> b$. Apply the simple compromiser substrategy which works as follows:
   - If I am an attendee of the meeting, search for any times proposed by the initiator that I have free but have not offered. If one or more such times exist offer my most preferred time. Else, offer the time proposed by the initiator that contains the meeting with the fewest participants.
   - If I am the initiator rank all the times proposed by other agents according to the number of agents that have proposed that time. Out of all the times with the highest number of proposals if any of these times are available, offer my most preferred such time, otherwise offer the unavailable time containing the meeting with the fewest participants.
4. **ABANDON**: if negotiation round > 50 return true.

Fig. 1. Offer-k-b negotiator

have been rescheduled for new times. A strategy is also considered to have been completely executed, if the learning agent has given up on scheduling the new meeting, or on rescheduling a bumped meeting. Each time a strategy terminates, the playbook weights are updated according to the success of the strategy.

4 Adapting Weights and Selecting Strategies for Negotiating with Different Agents

For each 'opponent' agent, the learning agent must learn which strategies to select. The learning algorithm has the following key components, (i) a rule for updating the weights on strategies in the playbook and (ii) a rule for selecting the strategy to apply based on these weights. Bowling *et al.* [1] used results in the literature on *experts* problems (also commonly referred to as *k-armed bandits* problems) to derive the rules required. We are able to use these rules for adapting weights on negotiation strategies. In this section, we briefly describe the approach and its basis in the experts literature. For a more complete treatment we refer to the reader to [1].

In the experts problem, an agent choses actions or options repeatedly based on the instructions it receives from a set of *experts*. Each time the agent needs to make a choice it selects which expert to listen to. In the traditional formulation, once the action or option has been selected, the agent receives a pay-off from that action. In addition, the pay-offs it would have received had it followed the advice of each of the other experts are revealed. The performance of the agent is measured by the notion of regret. Let the reward received from following the advice of expert i at choice point p be r_i^p. The regret

of the agent after k choices have been made is given by the following formula:

$$regret_k = \max_{over\ experts\ i} \sum_{p=0}^{k} r_i^p - \sum_{p=0}^{k} r_{x_p}^p$$

where x_p denotes the expert the agent chose at choice point p. Regret is simply the award achievable by always asking the best expert minus the reward actually achieved. A desirable property of an experts algorithm is that average regret goes to zero as the number of choices approaches infinity. There exist algorithms for various formulations of the problem that achieve no-regret in the limit e.g. [9, 10]

Bowling *et al.* [1] show how algorithms for selecting experts with no regret can be used to select plays. In the context of plays (and in the context of selecting strategies for negotiation), we need to use a different formulation of regret that takes into account the fact that not all plays (or strategies) are applicable at each choice point. This can be done by using the notion of *Sleeping Experts* developed by Freund et. al [11]. We say an expert is awake when it is applicable at a particular choice point, and asleep otherwise. Following the notation used in [1], we let $a_i^p = 1$ if expert i is awake at choice point p, and $a_i^p = 0$ otherwise. Then if $\Delta(n)$ is the set of probability distributions over all n experts, we get the following formula for sleeping regret (SR) after k choices:

$$SR_k = \left(\max_{x \in \Delta(n)} \sum_{p=1}^{k} \sum_{i=1}^{n} a_i^p \left(\frac{x(i)}{\sum_{j=1}^{n} x(j) a_j^p} \right) r_i^p \right) - \sum_{p=0}^{k} r_{x_p}^p$$

The first half of the formula simply quantifies the reward the agent could have received if the best possible distribution over awake experts had been selected at each choice point.

In the context of plays, and negotiation strategies, there is one final difficulty. Unlike in the traditional experts problem, agents only find out the reward of the action they actually take. In order to account for this, Bowling *et al.* [1] combine elements of the Exp3 algorithm proposed by Auer et al [10] (which handles the problem of unknown rewards) with the sleeping regret approach of [11]. We describe their approach here, and use it to adapt playbook weights for each 'opponent' agent, and select the strategy to use according to these weights.

Let $R_i^k = \sum_{p=0}^{k} \hat{r}_i^p$. Where $\hat{r}_i^p = 0$ if i not selected at point p and $\frac{r_i^p}{Pr(x_p=i)}$ otherwise. We call the weight for strategy i at decision point p, w_i^p, and we let $w_i^p = e^{R_i^p}$. The value $e^{r_i^p}$ is denoted as m_i^p, and we refer to this value as the multiplier and use it to adjust the weights according to the reward received from carrying out the negotiation strategy (or play). The probability that the strategy chosen at point p, denoted x_p, is strategy i is given by the following equation:

$$Pr(x_p = i) = \frac{a_i^p w_i^p}{\sum_j a_j^p w_j^p}$$

Once strategy x_p has been executed, and the reward $r_{x^p}^p$ received, we update the weights as follows:

$$w_i^t = \hat{w}_i^p . N_i^p$$

where $\hat{w}_i^p = w_i^{p-1}$ for i not selected, but for i selected:

$$\hat{w}_i^p = w_i^{p-1}(m_i^p)^{\frac{1}{Pr(x_p=i)}}$$

The N_i^p term is used to ensure that sleeping does not affect a strategy's probability of being chosen. $N_i^p = 1$ if $a_i^p = 0$ and otherwise:

$$N_i^p = \frac{\sum_j a_j^p w_j^{p-1}}{\sum_j a_j^p \hat{w}_j^p}$$

To apply the approach to negotiation we need to decide how we are going to set the multipliers. The multipliers specify the degree to which the success or failure of a strategy affects the weight. We base the multipliers on a model of user utility. We let the utility a user derives from a negotiation strategy take into account three elements:

1. the user's preference for the time-of-day (tod) the new meeting is scheduled for – $val(tod)$.
2. the increase (or decrease) in utility from moving other meetings, i.e., for all meetings that were moved, the agent's utility is increased by $\sum_{moved} val(tod_{new}) - \sum_{moved} val(tod_{old})$.
3. the number of negotiation rounds r required to schedule the new meeting and move any old meetings.

The user's utility function is parametrized by two constants α and β which specify the relative importance of time-of-day valuations and negotiation cost. Formally a user's utility for the outcome of a negotiation strategy is modeled as:

$$U(i) = \alpha(val(tod) + \sum_{moved} val(tod_{new}) - \sum_{moved} val(tod_{old})) - \beta r$$

We use the user's utility function and highest time-of-day value to estimate the maximum possible utility a negotiation strategy can achieve. We then set the multiplier according to how the reward actually achieved relates to this maximum. The multiplier is set according to the first row of Table 1 that applies. Also note that if the negotiation strategy fails to schedule the new meeting, or to reschedule any bumped meetings, a failure has occurred. Currently we use a multiplier of 0.25 for this case.

The bounds on regret obtained by using the plays approach are strongest if the 'opponent' agent is using a fixed strategy and we assume that changes to the environment (i.e., the calendars) are not affecting the rewards. If the other agent is also learning, then in the terminology of [10], we are dealing with a *non-oblivious* adversary. As such, since the playbook approach builds on Exp3, the theoretical bounds on regret are weaker.

5 Evaluation

In this section, we describe how we have evaluated the effectiveness of using a plays approach to selecting negotiation strategies.

$U(i) > 0.75 * maxU$	1.75
$U(i) > 0.5 * maxU$	1.5
$U(i) > 0.25 * maxU$	1.25
$U(i) > 0$	1
$U(i) > 0 - 0.25 * maxU$	0.75
$U(i) > 0 - 0.5 * maxU$	0.5
$U(i) > 0 - 0.75 * maxU$	0.25

Table 1. The multiplier is given by the first row for which the left hand entry evaluates to true

5.1 Communication Protocol

We have created a simulation environment consisting of a set of agents equipped with a common protocol for communicating about meetings. The protocol has three basic stages: a negotiation phase, in which agents exchange proposals, a pending stage, in which a time proposed by all the agents is agreed upon, and a confirmation stage, after which the meeting is entered into the agents' calendars. Support is also provided for *bumping* (canceling and rescheduling) meetings. There are a number of different types of messages that the agents exchange:

- meeting time proposals
- requests to bump meetings
- cancellation notices for meetings
- pending requests for times – when a meeting initiator finds an intersection in proposals, it sends a pending request for one of the times in the intersection to each of the participants.
- pending responses – when an attendee receives a pending request it responds with either:
  - a pending acceptance and marks the meeting as pending, or
  - a pending rejection (if the time is pending for another meeting, we require that the agent rejects the request).
- confirmation notices – sent out by the initiator when all attendees reply to a pending request with a pending acceptance.

5.2 Negotiation Strategies

We have implemented a number of negotiation strategies that comply with the protocol outlined. We use two of these strategies in our experiments in this paper. The first strategy – Offer-k-b was previously described (see Fig. 1.). This strategy is parametrized, and hence it covers a large number of distinct strategies. The second strategy we use is called Availability-Declarer (Fig. 2.). This strategy can be very useful in practice, particularly in situations where the agents are very busy. The key feature of this strategy is that it offers all the available times in the first week straight away. In subsequent negotiation rounds it does the same for later weeks.

1. **APPLICABILITY**: if importance(other-agent) $>=$ moderately-important return true.
2. **INDEPENDENT OFFER**: in the first round offer all available times for the current week, in second round offer all available times for the following week and so on until all available times up until the last possible time for the meeting have been offered.
3. **DEPENDENT OFFER**: if negotiation round > 5, apply the simple compromiser sub-strategy described in Fig. 1.
4. **ABANDON**: if negotiation round > 50 return true.

Fig. 2. Availability Declaring Negotiator

5.3 Preferences

We use a simple model of time-of-day preferences. Each agent has a preference ordering over morning times, middle of the day times and afternoon times. For example, if an agent prefers the morning, then the middle of the day, and then the afternoon, times in the morning are assigned a value of 3, times in the middle of the day a value of 2 and times in the afternoon, a value of 1.

5.4 Experiments and Results

We have empirically evaluated the effectiveness of using a plays approach to select negotiation strategies. The experiments we describe consist of one learning agent, which we are evaluating, and three fixed strategy agents of varying preferences and busyness. We also look at the effect of adding another learning agent into the mix. The learning agents have three strategies in their playbooks – Availability-Declarer, Offer-10-5 and Offer-3-5. In the experiments discussed, these strategies are always applicable.

Convergence In each experiment, the agents schedule approximately 80 new two person meetings (we restrict our attention to two-person meetings to simplify the discussion). The learning agent is an attendee (not an initiator) of each of these 80 meetings. We show how the learning agent's playbook weights converge to sensible strategies for each of the fixed strategy agents.

In our first experiment the learning agent's time preference is morning, then midday and then afternoon. The α and β values of the learning agent's utility function are 4 and 0.1 respectively. The agent's calendar is approximately 25% full when the experiment is started. Unlike the meetings we schedule in the testing phase, the initial meetings in the calendar can involve any number of the agents.

Fig. 3. shows how the learning agent's playbook weights adapt for Agent2. Agent2 starts out with a similar number of initial meetings to the learning agent, uses the Availability-Declarer strategy, and has the same time preferences as the learning agent. Fig. 3. shows how the playbook weights quickly converge to towards the Availability-Declarer strategy. While the other two strategies are also likely to work well in this instance, the Availability Declarer strategy offers the possibility resolving the negotiation faster. Since the learning agent and Agent2 have the same preferences, there is no strategic advantage to the learning agent only releasing its availability slowly.

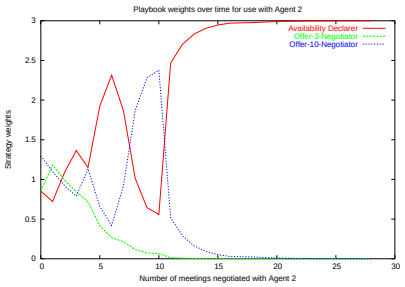

Fig. 3. Weights adaptation for Agent2

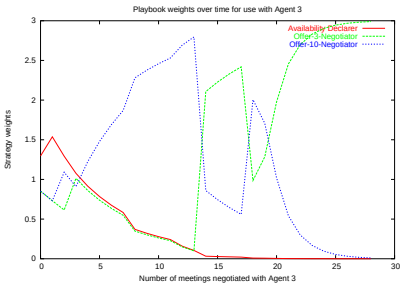

Fig. 4. Weight adaptation for Agent3

Fig. 4. shows the weight adaptation for Agent3. Agent3 uses the Availability-Declarer strategy and starts out with a similar calendar density to the learning agent, but with opposite preferences. Agent3 most prefers afternoons, then the middle of the day, and then the morning. Fig. 4. shows that the learning agent quickly establishes that the Availability-Declarer strategy is less useful for negotiating with Agent3 than the Offer-10-5 and Offer-3-5 strategies. After about 25 meetings have been scheduled the weights converge on the the Offer-3-5 strategy. Note that the Availability-Declarer strategy is a poor choice for use with Agent3. When both agents negotiate with this strategy, the initiator (always Agent3 in these experiments) is likely to quickly find a large intersection of available times. The initiator can choose its most preferred time in this intersection and since Agent3's and the learning agent's preferences clash, the time chosen will likely be bad for the learning agent. The learning agent has a clear strategic incentive to declare its available times more slowly and in order of preference. Since the learning agent's utility function rates achieving good times-of-day much higher than minimizing the number of negotiation rounds, it converges on the Offer-3-5 strategy rather than the Offer-10-5. This is despite the learning agent's calendar being quite full (93%), and hence mutally available slots fairly rare, by the time the experiment concludes.

Fig. 5. shows the weight adaptation for Agent4. Agent4 has similar preferences (midday, morning, then afternoon) to the learning agent. Agent4 uses the Offer-10-5 negotiator and starts with a dense calendar (about 80% full). Fig. 5. shows that the learning Agent quickly determines that the Offer-3-5 strategy is not very effective when dealing with a very busy agent that has similar preferences. After approximately 15 meetings have been scheduled, the learning agent converges on the Availability-Declarer strategy.

We ran the same experiment described above but with a different utility function for the learning agent and different initial calendars. The utility function had α as 4, and β as 1. This change caused the weights to converge on Availability-Declarer for each of the agents, since the negative effect of negotiation length was greatly increased.

Performance No regret algorithms bound the average difference between the performance of the learning algorithm, and the best fixed strategy in the limit. However, since a learning agent does not schedule an infinite number of meetings with each other agent, it is important to examine how well the learning algorithm performs in practice.

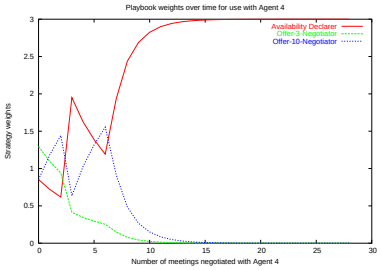

Fig. 5. Weight adaptation for Agent4

We used the 4 agents previously described (the play learning agent and the three fixed agents) and ran 10 trials. In each trial the agents' calendars were randomly initialized with 160 meetings. 200 new meetings were scheduled in each trial, but the calendars were cleared to their initial state after every 20 meetings were scheduled. This reflects the common scenario where people have a set of meetings that occur weekly and new meetings that arise over time. Fig. 6. shows the learning algorithm achieving higher utility than playing a random strategy or using any fixed strategy.

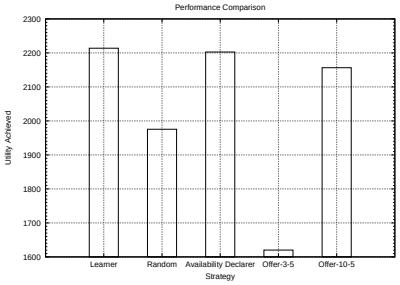

Fig. 6. Performance against only fixed agents.

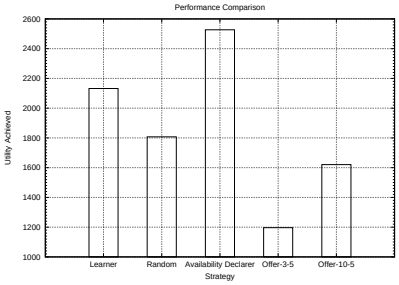

Fig. 7. Performance when another learning agent is added.

The learning algorithm used gives the strongest regret guarantees when the other agents are fixed. Fig. 7. shows that the learning algorithm also performs reasonably well when we add a learning agent (that uses the same algorithm), to the three fixed agents. These results are typical of a variety of experimental configurations.

Discussion Using a small number of alternative strategies and agents, we were able to show that the learning converged in a sensible way. We were also able to show that when the other agents used fixed strategies, the learning algorithm performed better than using the best fixed strategy. It is important to remember that the theoretical results only bound the regret between the pay-off from using the learning algorithm, versus the best fixed strategy that is *in the playbook*. This means that the strategies that appear in

the playbook must be carefully selected by a human expert. It is also worth noting that a large playbook would make learning impractical. As such, it might be worthwhile, in practice, to use a hierarchy of playbooks. For instance, initially a very diverse playbook might be used to decide which was the best class of strategies. A playbook containing different strategies from that class could then be used to tune parameters. This seems a promising avenue for future research.

Another avenue for future research is improving the performance results for the case where more than one agent is learning. In the future, we would like to experiment with a variety of no-regret style algorithms that are more specifically designed for this case.

6 Conclusions

We introduced the idea of using a *playbook* approach for learning to select the best strategies for negotiating with different agents. The space of negotiation strategies is huge, and thus it is not possible for an agent to learn how to negotiate in the complete space. The plays-based approach cuts the strategy space down to a set of strategies that are effective in different situations, allowing an agent to learn which of these strategies work best with different fixed-strategy agents. This approach provides some theoretical bounds on the regret the learning agent can experience. We have demonstrated experimentally that using a plays-based approach leads to good performance.

References

1. Bowling, M., Browning, B., Veloso, M.: Plays as effective multiagent plans enabling opponent-adaptive play selection. In: Proceedings of International Conference on Automated Planning and Scheduling (ICAPS'04). (2004)
2. Crawford, E., Veloso, M.: Opportunities for learning in multi-agent meeting scheduling. In: Proceedings of the AAAI Symposium on Artificial Multiagent Learning. (2004)
3. Jennings, N.R., Jackson, A.J.: Agent based meeting scheduling: A design and implementation. IEE Electronics Letters **31** (1995) 350–352
4. Sen, S., Durfee, E.: A formal study of distributed meeting scheduling. Group Decision and Negotiation **7** (1998) 265–289
5. Modi, P.J., Veloso, M.: Bumping strategies for the private incremental multiagent agreement problem. In: AAAI Spring Symposium on Persistant Agents. (2005)
6. Ephrati, E., Zlotkin, G., Rosenschein, J.: A non–manipulable meeting scheduling system. In: Proc. International Workshop on Distributed Artificial Intelligence, Seatle, WA (1994)
7. Garrido, L., Sycara, K.: Multi-agent meeting scheduling: Preliminary experimental results. In: Proceedings of the First International Conference on Multi-Agent Systems. (1995)
8. Shintani, T., Ito, T., Sycara, K.: Multiple negotiations among agents for a distributed meeting scheduler. In: Proceedings of the Fourth International Conference on MultiAgent Systems. (2000) 435 – 436
9. Littlestone, N., Warmuth, M.: The weighted majority algorithm. In: IEEE Symposium on Foundations of Computer Science. (1989) 256–261
10. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.: Gambling in a rigged casino: the adversarial multi-armed bandit problem. In: Proceedings of the 36th Annual FOCS. (1995)
11. Freund, Y., Schapire, R., Singer, Y., Warmuth, M.: Using and combining predictors that specialize. In: STOC. (1997)