Learning to Select Negotiation Strategies with Strategic Experts

Elisabeth Crawford[*] and Manuela Veloso
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA, USA, 15213
{ehc,mmv}@cs.cmu.edu

ABSTRACT

In many multiagent systems, agents need to negotiate amongst themselves in order to reach agreements. In such systems, agents (or the users they represent) often have private preferences about the negotiation. For instance, agents may have strong preferences for some outcomes over others, and preferences about how the negotiation is conducted. In a multiagent system, without a centralized control, it is possible for agents to use a wide range of negotiation strategies. The effectiveness of an agent's choice of strategy is heavily dependent on (i) the preferences of the other participants in the negotiation, and (ii) the strategies used by these participants. As such, in order to negotiate effectively, agents need to learn how to choose appropriate negotiation strategies. In this paper, we describe an approach, based on the Strategic Experts Algorithm [7], to learning how to select negotiation strategies in a multiagent setting. We demonstrate the usefulness of this approach in the domain of multiagent meeting scheduling.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Agents, Multiagent Systems

Keywords

Multiagent Learning, Automated Negotiation, Meeting Scheduling

[*]The first author is a student.

1. INTRODUCTION

Negotiation is widely used in multiagent systems for reaching agreements among agents. For instance, negotiation is used to agree upon prices in e-commerce systems, e.g. [13], for resource allocation, e.g. [17], and for scheduling meetings, e.g. [12]. While negotiation among multiple agents assumes that agents want to reach agreement, it is the case that agents also have private preferences that they wish to maximize.

In order to maximize their private preferences, agents need to adapt the way they negotiate. Different strategies are likely to work well in different situations. For example, when an agent negotiates with an honest agent, the best strategy might be to also be honest, but when the other agent is not being honest, it is likely to be in the agent's interests to act strategically (so as not to be taken advantage of).

Learning agents aim at improving their negotiation strategies with experience. There are two main factors that make adapting negotiation difficult. Firstly, the space of possible negotiation strategies may be very large. Secondly, choosing the right strategy for a specific situation, given an agent's limited knowledge, is not trivial.

One way to solve the first problem is to restrict the agent's attention to a diverse set of useful strategies (e.g. [18, 5]). These strategies can be learned or be chosen by human experts. Given a set of diverse strategies, an agent must then learn to choose good strategies for different situations. Unfortunately, there are a huge variety of factors in multiagent systems that can affect the success of negotiation strategies. These factors include things about the system state, the strategies of other agents, and the preferences/payoffs of other agents.

One approach to address the second problem, i.e., to learn to select the best strategy, would be to attempt to model relevant aspects of the multiagent system. An agent could try and identify different aspects, including the strategy used by each other agent, and the payoff function/preferences of each other agent. However, given that agents may only rarely interact with each other, learning a model of another agent is clearly very difficult. Furthermore, in complex systems, even if an agent *does* have a lot of accurate information about the current situation, it may be very hard to work out what strategy would match the situation best. As such, we propose that agents learn what strategies to select by *observing their own rewards*, as opposed to trying to model other agents and the state of the system.

Agents that can learn online, from the rewards they receive, what strategies work well, either generally, or for different, easily identifiable contexts, can potentially solve the strategy selection problem. In the context of multiagent meeting scheduling, where agents wish to learn what strategies work well with what agents, we demonstrated that algorithms that seek to minimize regret [3] can be very useful [5] for strategy selection. The approach was designed under the stationary assumption that the strategies of the other agents do not also adapt. When we applied these regret-minimizing algorithms to the case where multiple agents are learning, the performance degraded as expected.

In this paper, we study the problem of learning negotiation strategies in the presence of learning agents. We propose an approach to learning to select negotiation strategies that uses the Strategic Experts Algorithm (SEA) [7], or its generalization, Exploration-Exploitation Experts (EEE) [6]. These experts algorithms are designed to work in reactive environments, where the actions taken by an agent can affect its future payoffs. Although bounds on the performance of SEA and EEE in the limit have been proven [6], it is not guaranteed that our approach will work well, since in real-world multiagent negotiation problems the number of negotiation episodes is limited.

Thus, we demonstrate the usefulness of our approach on a real-world negotiation problem. The multiagent meeting scheduling problem is an important real-world problem since organizing meetings is a time-consuming task. As such, there have been a number of efforts to develop multiagent meeting scheduling systems, e.g., [14, 10], and more recently [12, 2]. Since users have ownership of their own calendars, and private preferences about meeting scheduling, we approach the meeting scheduling 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 [4] for a discussion). However, very little work has been done on how agents can negotiate strategically to improve the utility they provide their users.

We show that our approach can be used to select good negotiation strategies in the multiagent meeting scheduling problem. It works particularly well in the case where the other agents in the system are also adapting. Interestingly, we also found that in most cases the agents using our approach tended to settle on generous negotiation strategies. Thus, even though all the agents were attempting to optimize their own user's utility, this did not generally result in them employing uncooperative negotiation strategies.

2. LEARNING TO SELECT NEGOTIATION STRATEGIES IN A STATIONARY ENVIRONMENT

2.1 The Negotiation Problem

For each negotiation episode, the agent a must select a strategy s from S_a, the set of all strategies available to it. The agent uses this strategy to carry out the negotiation. At the end of the negotiation a determines its immediate change in utility, which we will refer to as $a's$ reward. We say that $a's$ reward is determined by the environment and it may depend on many factors such as, the agent or agents a negotiated with, and $a's$ past behavior.

How a should select its strategy, for a particular negotiation, depends on the rewards a expects to receive from the strategies available to it. a can base its expectations on the rewards it received using its strategies in similar negotiation contexts. We assume that the set of all negotiation contexts, from $a's$ perspective, C^a, has been divided into subsets C_i^a such that for all $c, c' \in C_i^a$, c is similar to c'. For instance, in a multiagent system where agents conduct pairwise negotiations, we might have a subset of C^a for each agent a negotiates with.

However, learning to select good negotiation strategies is an online problem. An agent cannot simply select the strategy that has performed best in the past because (i) the agent may not have experience of each of the negotiation strategies in the current context, (ii) a strategy may perform poorly (or conversely well), a few times, due to chance or an uncommon feature of the context C_i^a, and (iii) the best strategy to use may change over time. To perform well an agent needs to *explore* (try strategies at random) as well as exploit (select strategies according to past experience).

In this paper we describe two approaches to handling the exploration-exploitation trade-off. The first, an approach based on work by Bowling, Browning and Veloso [3] in the domain of robot we describe in this section.

2.2 Plays Approach

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.* [3] introduce the notion of a play as a team plan. Each play has an applicability condition and a termination condition, and assigns a role to each of the robots.

All the plays available to the team are placed in the *playbook* by a human expert. During the course of a game, plays are weighted according to the reward they produce. 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.

In [5] we showed how we can adapt the playbook approach to the specific problem of negotiating meeting times in a multiagent system. Negotiation problems in general, have a number of important features in common with small-size robot soccer. In both domains, the space of available strategies can be use. It is not possible for agents to adapt online if they must consider the entire space. Furthermore, in both domains, the models of the 'opponents' are unknown, and online learning is required for good performance.

We can map the terminology, from robot soccer, to the problem of selecting negotiation strategies as follows. The plays correspond to complete negotiation strategies, the opponent corresponds to the agent the agent a is negotiating with (or more generally to the current context C_i^a), and the playbook is simply the set of negotiation strategies available to a. Unlike in robot soccer, in multiagent negotiation problems, we are playing with multiple 'opponent' agents at the same time. As such, the learning agent must adapt strategy selection differently for the different contexts C_i^a simultaneously.

2.2.1 Learning to Weight Playbook Strategies

For each context C_i^a agent a must learn which strategy to select. The playbook 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.* [3] used results in the literature on *experts* problems to derive the rules required. These rules can also be used for adapting weights on negotiation strategies. We will now briefly describe the plays approach and its basis in the experts literature.

In the experts problem, an agent choses actions 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. There exist algorithms for various formulations of the expert problem that guarantee that average regret goes to zero as the number of times the agent choses goes to infinity e.g. [11, 1]. We will say that such algorithms have no-regret.

Bowling *et al.* [3] combine elements of the Exp3 algorithm proposed by Auer et al [1] (which handles the problem of unknown rewards) with the sleeping regret approach of [8]. We describe their approach here, in the context of negotiation strategies.

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. The probability that the strategy chosen at point p, denoted x_p, is strategy i is given by the following equation:

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

Where a_i^p is a binary value indicating whether or not i can be applied at point 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 not being applicable does not affect a strategy's probability of being chosen. $N_i^p = 1$ if $a_i^p = 0$ and otherwise:

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

In this paper will not consider negotiation strategies that are only sometimes applicable, however it is a nice feature of the plays approach that it can handle this case.

For the negotiation problem an agent a using this approach must adapt the playbook weights separately for each distinct negotiation context. In other words, a uses the algorithm to learn a set of weights for each negotiation context.

2.2.2 Discussion

The plays approach is designed to minimize regret, in particular, to ensure that in the limit, average regret goes to zero. However, regret (by definition) is calculated as the difference between how well the learning algorithm does versus how well it could have done had it always selected the best fixed strategy against the sequence of actions that were chosen by the opponent (or environment). The plays approach, and regret minimization algorithms generally, are not designed to handle the case when agents' actions/strategies impact the environment, and the choices of other agents.

We found, that an agent using a plays based approach to selecting negotiation strategies performed as well as the best strategy from its playbook when negotiating with fixed strategy agents [5]. Furthermore, an agent using the approach was able to converge on good strategies after only approximately 25 meetings were scheduled. However, when we added even one other player to the system, that was also learning, we noticed that performance was significantly worse than the best fixed strategy. Given that the algorithm is not designed to handle this case, this drop in performance is not surprising. In meeting negotiation, as in other negotiation settings, it is not safe to assume that the other agents use fixed strategies. We need a method that can learn to select good strategies when the other agents are also adapting.

3. A STRATEGIC EXPERTS APPROACH FOR LEARNING IN A REACTIVE ENVIRONMENT

In this paper, we propose an approach to learning to select negotiation strategies that handles the case where other agents are also learning. de Farias and Megiddo recently introduced the Strategic Experts Algorithm (SEA) [7], as well as a generalized version, Exploration-Exploitation Experts (EEE) [6]. These algorithms learn to take the advice of the best expert when playing a repeated stage game. Unlike previous experts algorithms, SEA and EEE, can take into account how the agent's actions affect the actions of the environment or an opponent player. The key idea behind the algorithms, is that when actions affect the environment and other agents, each expert (in our case a strategy) needs to be used multiple times in succession to gauge its effect.

Our approach to learning to select negotiation strategies involves running the SEA or EEE algorithm for each context, C_i^a. This allows an agent a to learn online what strategies are good in each context. We will now describe these algorithms in terms of our negotiation problem.

The key steps in the SEA and EEE algorithms are the exploration and exploitation steps. Before we describe these steps, we need to define two terms. A *stage*, is simply one run of the stage game (recall that SEA and EEE are designed for use in repeated stage games). In our case this translates to one negotiation episode. A *phase*, is a sequence of stages, for which the same expert (in our case strategy) is used. Given these terms we can define the exploration and exploitation steps.

- *Exploration*: At the beginning of a new phase, a needs to select a strategy. With some probability a *explores*, by choosing a strategy $s \in S_a$ uniformly at random. a then uses this strategy in every stage of the new phase.

- *Exploitation*: When a is not exploring, it instead *exploits* its learned knowledge. When exploiting, a identifies the current context C_i^a and picks the strategy with the highest past average reward for this context (ties are broken randomly). As in the exploration step, a then uses this strategy in every stage of the new phase.

Figure 1 shows the details of the algorithms in terms of our negotiation problem. For a number of different settings of EEE, de Farias and Megiddo show bounds, of various kinds, on the performance difference between the best fixed strategy and the learning strategy in the limit.

4. MULTIAGENT MEETING SCHEDULING: NEGOTIATING MEETING TIMES

Multiagent meeting scheduling is an important, real-world domain where negotiation can be used to reach agreements. In fact, in most multiagent meeting scheduling systems that have been proposed, meeting times are agreed upon via negotiation between meeting participants. Typical negotiation protocols feature a meeting initiator that proposes meeting times and collects the proposals of other participants. The following simplified protocol is typical:

- while there is no intersection in proposals
  - the initiator proposes some times to the other agents
  - each agent proposes some times to the initiator

A negotiation strategy, for meeting scheduling, is thus a set of rules for deciding what times to propose at each point in the process. Users often have strong preferences about meeting scheduling. In particular, users have preferences about when their meetings are scheduled, as well as about the scheduling process itself e.g., about how long it takes. Despite this, we are not aware of any other work on strategic negotiation in this domain.

A number of negotiation approaches for multiagent meeting scheduling have been proposed. Sen and Durfee [15] 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, [10], and [9], while [16] looked at a more complex protocol. These negotiation approaches have handled user preferences for meeting times in quite different ways. Shintani *et al.* [16] 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.

Garrido and Sycara [9] and Jennings and Jackson [10] allow agents to not only propose meeting times, but also to quantify their preferences for proposals. The agent that is collecting the proposals, then chooses the meeting time based on the reported utilities of all the meeting participants.

The approaches outlined, can work very well if we assume that all the agents comply with the protocols and behave cooperatively. However if we relax this assumption, an agent that always complies e.g., by reporting its true preferences for times, may be taken advantage of by agents that lie and exaggerate their preferences. By negotiating strategically agents can work to satisfy their users' preferences. Note that these preferences are not necessarily entirely selfish, in fact it is likely that many users are willing to accept times they do not like if it is necessary for the meeting to be scheduled and so forth.

The approach to negotiation we propose in this paper, fits very well with the meeting negotiation problem. Firstly, there are many possible strategies for negotiating meeting times. It is simply not possible for an agent to learn in the whole space of strategies in reasonable amount of time. Our approach solves this problem because it cuts down the number of strategies an agent must consider. Secondly, the environment is not stationary - meetings are added and moved and other agents can also adapt their strategies. Thus it is important to use an approach that can handle a reactive environment.

In order to apply our approach to the meeting negotiation problem, we need to divide the set of possible negotiation contexts, from an agent a's perspective, into sensible subsets. We note that agents can differ in many ways e.g. agents can use very different strategies and have users with a variety of preferences, and can represent users of different importance and busyness. Clearly a strategy that works well for negotiating with one agent may work very poorly with another. So we at least need subsets C_i^a for each other agent i that a negotiates with. If a is attending a meeting and agent i is the initiator, a would identify the context as C_i^a. If agent a is initiating a meeting that more than one agent attends, then a would identify a different negotiation context for each of the attendees. We demonstrate in the next section that agents can use our approach with these contexts to achieve good performance.

5. EXPERIMENTS

In this section we demonstrate that our proposed approach to negotiation works well in the multiagent meeting scheduling problem. We first describe the experimental setup, including the negotiation strategies considered, the communication protocol, and the model of agent preferences.

- $R_s^{C_i^a}$ is the average reward achieved by strategy s in context C_i^a.
- $N_s^{C_i^a}$ is the number of phases in which s has been followed in context C_i^a.
- $S_s^{C_i^a}$ is the number of times a has negotiated using s in context C_i^a.
- $i^{C_i^a}$ is the number of phases for C_i^a so far.

1. *Initialization*: $\forall C_i^a, s R_s^{C_i^a} = N_s^{C_i^a} = S_s^{C_i^a} = 0$ and $i^{C_i^a} = 1$
2. *Explore/Exploit Decision*: Explore with probability $p_i^{C_i^a}$, if using the SEA restriction $p_i^{C_i^a} = 1/i^{C_i^a}$. With probability $1 - p_i^{C_i^a}$ exploit. Let s be the strategy chosen.
3. *Use and Update*: Use s for the next n negotiations in context C_i^a. If using the SEA variant, $n = N_s^{C_i^a}$. Then set $N_s^{C_i^a} = N_s^{C_i^a} + 1$, $S_s^{C_i^a} = S_s^{C_i^a} + n$. If we let $\hat{R}$ be the average payoff received over the n stages, then the update for $R_s^{C_i^a}$ is given by the following formula:
$$R_s^{C_i^a} = R_s^{C_i^a} + \frac{n}{S_s^{C_i^a}}(\hat{R} - R_s^{C_i^a})$$
4. $i^{C_i^a} = i^{C_i^a} + 1$, go to 2.

Figure 1: EEE Algorithm, adapted from de Farias and Megiddo, 2005

5.1 Negotiation Strategies

We have included two negotiation strategies in our experiments. However one strategy is parametrized and can actually represent multiple, diverse, strategies. An agent, a using the Offer-k-b strategy offers k new times that are free in its calendar every negotiation round. After b negotiation rounds a compromises by offering one of the times that have been proposed to it. The details of this negotiation strategy are given in figure 2. Different parameters for k and b result in very different negotiation strategies. If k is very low and b very large the agent offers its less preferred times very slowly. If k is large or b small however the strategy that results is very generous.

The other strategy we consider, AvailabilityDeclarer, is a very cooperative strategy. In practice this strategy proves to be very useful, particularly when agents are very busy. An agent a using the AvailabilityDeclarer strategy offers all of its available times in the first week straight away. In subsequent negotiation rounds, it does the same for later weeks. The details of this strategy are shown in figure 3.

5.2 Communication Protocol

In order for agents to negotiate meeting times they need a common communication protocol. Our 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:

- 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.

5.3 Preferences

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:

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

In our approach, every time a meeting is scheduled, a stage ends. We use the utility function described to evaluate the reward the agent receives in the stage. This reward is used

1. In any negotiation round offer a's k most preferred, available, un-offered times.
2. If negotiation round $> b$. Apply the simple compromiser sub-strategy which works as follows:
    - If a is an attendee of the meeting, a searches for any times proposed by the initiator that a is free for, but has not offered. If one or more such times exist offer a offers its most preferred such time. Else, a offers the time proposed by the initiator that contains the meeting with the fewest participants.
    - If a is the initiator, a ranks 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, a offers its most preferred such time, otherwise a offers the unavailable time containing the meeting with the fewest participants.
3. If negotiation round is greater than some limit x the scheduling is abandoned.

Figure 2: Offer-k-b negotiator

1. In the first round a offers all its available times for the current week, in second round offers all its 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.
2. If negotiation round > 5, a applies the simple compromiser sub-strategy described in Fig. 2.
3. If negotiation round > 50 a abandons the scheduling.

Figure 3: Availability Declaring Negotiator

by the algorithm to track the effectiveness of the negotiation strategies in each context.

5.4 Evaluation Procedure

In the experiments we will describe all the meetings have two participants. This makes it easier to reason about why the learning converges upon particular strategies. The initial calendars of the agents are populated with randomly generated meetings. The agents are then given a set of meetings to schedule. In each experiment one agent is designated as the agent to be evaluated. This agent uses our learning approach to negotiate the set of meetings with the other agents. Before the agent schedules these meetings however, we record the initial state of all the agents' calendars. This allows us to reset all the calendars and then have the agent we are evaluating schedule the meetings again with non-learning strategies. In particular, we have the designated agent schedule the meetings using a random approach. Each time a meeting needs to be negotiated, the random approach simply selects from amongst the agent's strategy pool S_a uniformly at random. We also have the agent schedule the meetings using each fixed strategy from S_a. This allows us to compare the performance of our learning approach against the random approach and against each of the fixed strategies.

5.5 Results

Figure 4 shows the performance of our approach to learning to select negotiation strategies. We evaluated an agent learning with our algorithm against 4 other agents also using the algorithm. The agents had the following strategies available to them: Offer-1-20, Offer-3-20, Offer-5-20, Offer-10-10, Offer-15-10, AvailabilityDeclarer. We used a version of the learning algorithm with a fixed phase length of 5. The agents had a variety of preferences, favoring different times of the day and having different α and β values. As the figure shows, the learning approach performed much better than random strategy selection. It did not perform quite as well as the best fixed strategy, AvailabilityDeclarer, but it did outperform all the other fixed strategies. It was interesting to note, that despite the differing preferences of the agents they nonetheless converged towards using the more cooperative strategies. In fact, AvailabilityDeclarer was often the highest ranked strategy by the agents for every context (i.e., for negotiating with each other agent). Furthermore, the learning happened relative quickly, with agents generally discovering one of the good strategies after scheduling less than 25 meetings with a particular agent. We found however, that when we used phases of length $n = N_s^{C_i^a}$ (i.e, phases that increased in length over time) that the algorithm did not get enough chance to explore, even if 100 meetings were scheduled with each agent. This lead to agents sometimes not identifying good strategies. Using a fixed phase length however gave the algorithm a good chance of exploring all the strategies early on.

We also found that our learning algorithm performed well against fixed strategy agents. Figure 5 shows how an agent using our approach to select between the strategies: Offer-1-20, Offer-10-10 and Offer-20-10, performed against 4 fixed strategy agents. This performance is not as good as we were able to achieve using a plays-based approach in [5]. But could be improved by tuning the algorithm parameters

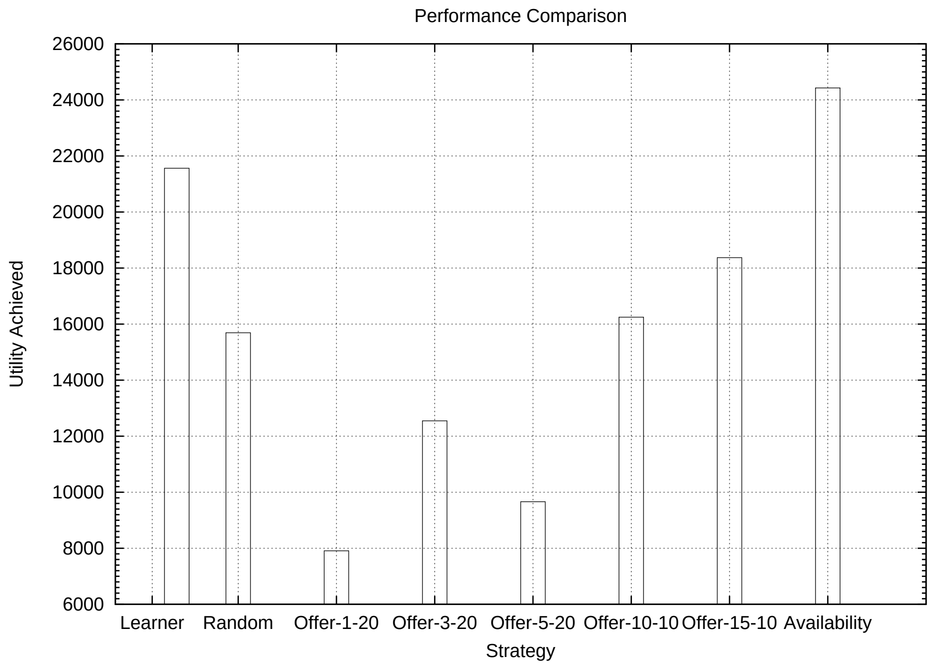

Figure 4: Performance comparison when all agents are learning, results averaged over 10 trials

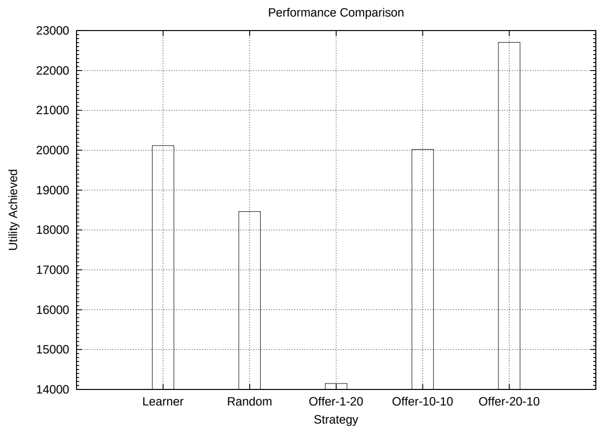

Figure 5: Performance comparison when other agents use: Availability Declarer, Offer-1-5, Offer-10-10 and Offer-20-10, results averaged over 5 trials

for the case where the environment is stationary.

We found that the learning algorithm was appropriately sensitive to changes in the user's utility function. In the cases where the agent's utility function had high α values and low β values, we found that the agent would sometimes converge to selecting one of the less cooperative strategies. In particular, this tended to occur when the other agent was using a fixed strategy and had opposite preferences. This did not occur in all cases however, since sometimes it is unnecessary e.g.. in the case where the other agent is mostly only available at times the learning agent prefers.

6. CONCLUSION

We introduced an approach to learning to select strategies for negotiation in a multiagent system, based on the SEA and EEE algorithms in [7, 6]. Our approach effectively handles the case where choice of strategy impacts the environment and the strategies of other agents. We showed that our approach can be used to select good strategies in the multiagent meeting scheduling problem. Furthermore we found that by using this approach, in most cases the agents learnt to cooperate, despite the fact that each agent was aiming to maximize its user's preferences.

7. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual FOCS*, 1995.

[2] P. Berry, M. Gervasio, T. Uribe, K. Myers, and K. Nitz. A personalized calendar assistant. In *In the AAAI Spring Symposium Series, March*, 2004.

[3] M. Bowling, B. Browning, and M. Veloso. Plays as effective multiagent plans enabling opponent-adaptive play selection. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS'04)*, 2004.

[4] E. Crawford and M. Veloso. Opportunities for learning in multi-agent meeting scheduling. In *Proceedings of the AAAI Symposium on Artificial Multiagent Learning*, 2004.

[5] E. Crawford and M. Veloso. Learning to select negotiation strategies in multi-agent meeting scheduling. In *Proceedings of 12th Portugese Conference on Artificial Intelligence, Springer, LNCS*, 2005.

[6] D. de Farias and N. Megiddo. Exploration-exploitation tradeoffs for experts

algorithms in reactive environments. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 409–416. MIT Press, Cambridge, MA, 2005.

[7] D. P. de Farias and N. Megiddo. How to combine expert (and novice) advice when actions impact the environment? In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[8] Y. Freund, R. Schapire, Y. Singer, and M. Warmuth. Using and combining predictors that specialize. In *STOC*, 1997.

[9] L. Garrido and K. Sycara. Multi-agent meeting scheduling: Preliminary experimental results. In *Proceedings of the First International Conference on Multi-Agent Systems*, 1995.

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

[11] N. Littlestone and M. Warmuth. The weighted majority algorithm. In *IEEE Symposium on Foundations of Computer Science*, pages 256–261, 1989.

[12] P. J. Modi, M. Veloso, S. F. Smith, and J. Oh. Cmradar: A personal assistant agent for calendar management. In *6th International Workshop on Agent-Oriented Information Systems*, 2004.

[13] V. Narayanan and N. Jennings. An adaptive bilateral negotiation model for e-commerce settings. In *7th International IEEE Conference on E-Commerce Technology*, 2005.

[14] S. Sandip and D. Edmund. On the design of an adaptive meeting scheduler. In *Proc. of the Tenth IEEE Conference on AI Applications*, pages 40–46, 1994.

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

[16] T. Shintani, T. Ito, and K. Sycara. Multiple negotiations among agents for a distributed meeting scheduler. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 435 – 436, July 2000.

[17] L.-K. Soh and C. Tsatsoulis. Reflective negotiating agents for real-time multisensor target tracking. In *Int. J. Conf. On Artificial Intelligence*, 2001.

[18] S. Zhang, S. Ye, F. Makedon, and J. Ford. A hybrid negotiation strategy mechanism in an automated negotiation system. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 256–257, New York, NY, USA, 2004. ACM Press.