Modeling Privately Owned Information in the Incremental Multiagent Agreement Problem

126

Abstract

In this paper we define the Extended Incremental Multiagent Agreement Problem with Preferences (EIMAPP). In EIMAPP, variables arise over time. For each variable some set of distributed agents must agree on which option (from a given set) to assign to the variable. Each of the agents may have a different preference about which option to use. EIMAPP is designed to reflect many real world multiagent agreement problems, including multi-robot task allocation and multiagent meeting scheduling. We show how EIMAPP can be solved distributively through multiagent negotiation. In order to negotiate most effectively, agents must reason about each other. Our approach builds on the fact that agents necessarily *reveal* some information about their own preferences and constraints as they negotiate agreements. We take this limited, and noisy information, and use it to build probabilistic models of other agents. We show how agents can use these models to negotiate more efficiently.

Introduction

In Agreement problems multiple parties must reach agreement on some issue. Agreement problems arise regularly in people's daily lives, and are generally solved via negotiation. Some examples include: (i) the exchange of availability information to schedule meetings; (ii) negotiation about role assignments when people work as a group; and (iii) group decision making about how to solve problems. Some of these negotiations, such as meeting scheduling, are largely carried out electronically and are thus suitable for automation. Personal assistants agents, with the capability to negotiate, could schedule meetings on behalf of their users and conduct other regular negotiations, such as task assignments.

Agreement problems also arise in multi-robot domains. Currently, when robots need to work together, it is often the case that the robots have all been built by the same company, or are owned by the same group of people. As such, they are sometimes controlled centrally, or assumed to operate as a team. In the future, as heterogeneous robots become widely deployed in the workplace and the home, robots will need to negotiate with each other to reach agreements in much the same way that people do. In order to carry out some tasks, robots with different owners, preferences, and capabilities, may need to work together. To do this they need to agree upon methods and role assignments. Given that robots have their own preferences and capabilities, this process of reaching agreement will require negotiation.

In this paper, we formalize agreement problems by defining the Extended Incremental Multiagent Agreement Problem with Preferences (EIMAPP). EIMAPP is a generalization of IMAP defined in (Modi & Veloso 2005). In EIMAPPs, variables, v, arise over time. For each variable, the set of decision makers, $\mathcal{D}_v$, must agree upon an option, o, from the set of all options, $\mathcal{O}$, to assign to v. Each of the agents has a privately owned preference function that specifies the agents payoff for assigning v to each of the different options.

EIMAPPs have some interesting properties. While each agent has its own preference function, agents only receive payoff when they reach agreements. As such, there is at least *some* built-in incentive to compromise. In certain domains (e.g., time critical domains), efficiency may be of particular importance to the agents. In these domains there is further incentive to compromise.

In this paper, we show how EIMAPP can be solved via distributed negotiation. We model negotiation in EIMAPP as a series of rounds. In each round, each of the agents in $\mathcal{D}_v$ offers a set of times $O_{v_t} \subseteq \mathcal{O}$. This offer is added to the options the agent has offered previously. When there is an intersection in the offers of all the agents, the negotiation ends and the agreement is confirmed. Within this model, agents are free to negotiate according to their own interests.

We show an agent can learn useful models of other agents by observing negotiation histories. In order to do this, the agent divides the options into a set of disjoint logical categorizes, $\mathcal{C}$. Using domain knowledge, the agent can hypothesize a conditional probability distribution, $Pr(\alpha \text{ proposes } o \mid o \in c)$ over $c \in \mathcal{C}$. This distribution gives the probability that another agent proposes an option, given that the option falls into the logical category c. For some other agent α, starting from some initial belief over the disjoint categories for each option, the agent can make a Bayesian update to its belief whenever α makes an offer. We show that this simple procedure can be very effective, and use it to learn models that allow for more efficient negotiation (in terms of number of negotiation rounds to reach agreement and the number of options offered).

Problem Definition

Definition 1: *The Extended Incremental Multiagent Agreement Problem (EIMAP)* consists of the following elements:

- A set of agents $\mathcal{A}$.
- A set of values/options $\mathcal{O}$.
- A set of variables $\mathcal{V}$. The set of variables is not provided up-front. Rather, new variables arrive over time. A variable that arrives at time t is denoted v_t.
- For each variable $v \in \mathcal{V}$, $\mathcal{D}_v \subseteq \mathcal{A}$ is the set of agents that decide v. To decide a variable, v, each of the agents in $\mathcal{D}_v$ must agree to assign a particular option $o \in \mathcal{O}$ to v.

Restricted option usage

In our definition of EIMAP we made no restriction on the variables an option could be assigned to. Two useful restrictions for modeling real-world problems are as follows:

- **Assign-once options** - in this restriction each option o can be assigned to at most one variable. Assign-once options can be used to represent collective use-once resources.
- **Assign-once per-agent options** - in this restriction a variable v can be assigned an option o only if $\forall \alpha \in \mathcal{D}_v$ there does not exist a $v' \in \mathcal{V}$ such that $\alpha \in \mathcal{D}_{v'}$ and v' is assigned to o. Assign-once per-agent options can be used to model problems like meeting scheduling, where each agent can have at most one meeting (variable) assigned to each time-slot (option) in its calendar. With this restriction, EIMAP corresponds to the Incremental Multiagent Agreement Problem (IMAP) defined by Modi et al. (Modi & Veloso 2005) in order to abstract the multiagent meeting scheduling problem.

Reassigning Variables in EIMAP

In some EIMAP domains it is possible to break the assignment of an option to a variable v. This has been referred to in the literature (Modi & Veloso 2005) as *bumping* v. Bumping has particular significance when we have assign-once and assign-once per-agent options. For instance, in the case of assign-once per-agent options it may be necessary to reassign one or more existing variables in order to make the assignment of a new more constrained variable possible.

Preferences in EIMAP

IMAP as defined by Modi et al. (Modi & Veloso 2005), does not include a built-in notion of agent preferences. Nonetheless in many IMAP and EIMAP domains, the agents have preferences about which options are assigned to variables. For instance, in a multi-robot task domain, some options for completing a task might be more expensive to a particular agent than others. In some cases, the preferences of the agents actually correspond to the private preferences of computer users. In the multiagent meeting scheduling domain, users have preferences about the times when they have meetings. For instance, some users might prefer morning meetings, while others prefer afternoon meetings.

Definition 2: *EIMAP with Preferences (EIMAPP)* is an instance of EIMAP where each agent $\alpha \in \mathcal{A}$ has a function $pref_\alpha$ that assigns to each $o \in \mathcal{O}$ a number which indicates α's preference for assigning option o to a variable v. The function $pref_\alpha$ is not necessarily simply a function of o, but, depending on the domain, might also be a function of v, or even $\mathcal{D}_v$.

Privately Owned Information and Reaching Agreements

In EIMAP domains there are typically a number of privacy considerations. Certain types of information are privately owned. In order to distributively reach agreements agents reveal some of their privately owned information, but it is important they retain control of what information is communicated.

Types of privately owned information

- **Variable Ownership**
  Only the agents $\mathcal{D}_v$ must know about the existence of v in order for it to be assigned an option. In some domains it can be undesirable for all the agents to know about all the variables. For instance, in the meeting scheduling domain, many users would not like all the other users in the system to know they are organizing an appointment with their doctor.
- **Assignment Ownership**
  Following from above, only the agents $\mathcal{D}_v$ need to know which option, o, is assigned to v. In the case of assign-once options, this means that any one agent may not have complete knowledge about which options are available for assignments. In the case of assign-once per-agent options, it means that an agent only has very incomplete information about which options each of the other agents has available. An agent α only knows that an option o is used by agent β if $\exists\, v$ s.t $\{\alpha, \beta\} \subseteq \mathcal{D}_v$ and v is assigned to o.
- **Preferences Ownership**
  In the case where agents have preference functions these functions are privately owned by the agents. Particularly when these preferences represent the preferences of users, it is important to allow for privacy considerations.

Reaching agreements through information sharing

Without some central control, in order to agree on an option to assign to a variable v, the agents, $\mathcal{D}_v$, must communicate with each other about options. For instance, one agent might propose that v be assigned option o. *This reveals some information.* For example, if we are in a assign-once per-agent setting, an agent is likely to propose o only if it has o available for assignment. Furthermore, self-interested agents are more likely to propose options they prefer than options they dislike.

While some information sharing is necessary, it is important that we take privacy considerations into account when designing the procedure agents use to reach agreements. Procedures which involve agents *having* to reveal all their assignments and preferences might be efficient, but are undesirable from the privacy perspective. Rather, we consider frameworks where agents have some control over what information they reveal.

Distributed Constraint Reasoning Approach

Modi et al. (Modi & Veloso 2005) use a Distributed Constraint Reasoning (DCR) style of approach to modeling multiagent agreement problems. In particular, they introduce the *Iterative Agreement Protocol* for reaching agreements in IMAP with assign-once per-agent options. In this protocol one agent is designated as the initiator of the variable v. The protocol operates in rounds as follows:

1. initiator proposes one option to all the agents in $\mathcal{D}_v$
2. each agent in $\mathcal{D}_v$, sends an accept/reject response to the initiator
3. if all the agents have accepted, then there is an agreement and the variable is assigned
4. else go to 1. and repeat

A key feature of the protocol is that an agent α accepts a proposal *whenever* it won't violate the use-once per-agent constraint, i.e, whenever α currently has the option available for assignment. If the option is assigned to another variable, then α must decide whether to bump the current assignment. Modi et al., introduce a number of strategies an agent can use to decide when to bump one variable in favour of another.

The approach proposed by Modi et al. (Modi & Veloso 2005), is most useful when agents do not have preferences. Modi et al., suggest that preferences about options could be expressed as additional constraints. For instance, a constraint that option o should never be used. This works very well for preferences that are easily modeled by "hard" constraints. However, the approach does not provide a ready solution for handling preferences in general i.e., "soft constraints".

Privately Owned Information in the DCR approach In Modi et al.'s Iterative Agreement Protocol, described above, the initiating agent has some control over what information it reveals, since it chooses which options to propose. The other agents however have much less control. They must explicitly accept or reject each proposal. In the protocol a rejection means that the agent has already assigned the option to another variable. An acceptance reveals less information, since the agent might have the option free or simply be willing to bump the variable currently assigned to the option. The key advantage of this approach, is that since agents are required to accept a proposal whenever the assign-once per-agent constraint is not violated, the process should settle on a option fairly quickly, thereby keeping the number of options the agents must accept/reject comparatively low. The disadvantage is the lack of control the agents have over what information they reveal.

Negotiation Approach

In this paper, we are particularly interested in domains where preferences are dominant. As such, *we model the agreement process as a negotiation between self-interested agents*.

Agent pay-offs for assigning variables For each v the agents $\mathcal{D}_v$ assign to o, they each receive a payoff according to their individual preference function, $pref_\alpha(o, v)$. Since the agents receive payoff for reaching agreements, it is in their interests to coordinate.

Negotiation process We structure the negotiation process as a series of rounds.

- while no intersection in proposed options:
  - each agent proposes a set of options
- confirm the assignment of some option in the intersection to the variable, using a set confirmation protocol

This process can be conducted either through multi-cast or an initiator agent that collects all the proposals. The approach allows for a wide variety of agent behaviour. Agents are free to try and optimize their payoff. They can accept any options offered to them, accept only options they prefer, accept options they don't prefer but only after a certain number of rounds, etc. This flexibility is important since it allows agent behaviour to reflect the wide range of behaviour that users may desire.

Negotiation Cost In most domains it is important for the agreement process to be efficient. As such, we make efficiency in the self-interest of the agents, by penalizing the payoff they receive according to the number of rounds required for negotiation.

Let the *negotiation cost* to agent α be $c_\alpha(r)$, where r is the number of rounds. In order to try and optimize their payoffs agents must weigh the relative importance (according to their own pay-off function) of getting an option they prefer with negotiating efficiently. The negotiation cost is set according the domain, and/or the preferences of any user/s the agent represents.

Privately Owned Information in the Negotiation Approach The negotiation approach does not require the agents to explicitly reveal any of their privately owned information. While agents will negotiate according to their preferences, they are never required to explicitly state their preference for an option or give their function $c_\alpha(r)$. Furthermore, they are never required to directly accept/reject a proposal when some condition (related to their privately owned information) holds. The flexibility of the negotiation approaches means that the agents have a large amount of control over what information they reveal.

Modeling Other Agents

In the previous section we discussed an approach to reaching agreements that involved negotiation amongst self-interested agents. In negotiating variable assignments agents necessarily reveal some information about themselves. Other agents can potentially use this information to improve their pay-offs. However, this is not necessarily a bad thing. The most obvious way for agents to use learned information is to increase their pay-offs by improving the efficiency of negotiations - and this benefits everyone. For example, consider the case where agent α has a set of strongly preferred options $\mathcal{O}_{pref_\alpha}$, and similarly agent β has a set of options $\mathcal{O}_{pref_\beta}$. If there is an $o \in \mathcal{O}_{pref_\alpha}$ that α knows is also in $\mathcal{O}_{pref_\beta}$, then α should offer o first, since β is likely

to be happy to agree to this option by proposing it in the next round. This situation works out well for both agents.

In this section, we show how agents can learn models of other agents from the very limited information the negotiation process provides. In the next section we show that such models can improve the efficiency of negotiation in a real-world domain.

Relevant Agent Characteristics

An agent negotiates by selecting which options to propose and when to propose them. When an agent α is deciding which options to propose to another agent β in the next round, what α really wants to know is *how likely β is to agree to the options* (if α proposes them) in the following round, and in subsequent rounds. Consider the example in Figure 1.

α needs to assign variable v with agent β.

- α has two options available for assignment, o_1 and o_2, such that $pref_\alpha(o_1) = 9$ and $pref_\alpha(o_2) = 6$
- α's cost per negotiation round $c_\alpha(r) = 2r$.
- o_1 is a very bad option for β and β's negotiation strategy stipulates that β will only agree to very bad options after more than 3 negotiation rounds have passed.
- o_2 on the other hand is a good option for β and β will agree on it right away.

If α has this information about β, α can compute the value of offering o_1 versus o_2.

- If α offers o_2 in the first round, β will agree in round 2 by also offering o_2. So α will get a total payoff of $6 - 2 * 2 = 2$.
- If α offers o_1 only, β will not agree until the 4th round. So α will get a total payoff of $9 - 4 * 2 = 1$.

So α should offer o_2.

Figure 1: Negotiation Example

The example in Figure 1 illustrates *the factors that influence how likely an agent is to agree to an option*. These factors include:

1. **option availability** - when in a restricted option setting
2. **the agent's preferences** - preferences for options as well as the cost of negotiation length to the agent
3. **the agent's negotiation strategy** - which is somewhat determined by the preferences

A self-interested agent would like to learn all this information, but negotiation reveals very little.

Training Data

The *training data* an agent can collect is very limited. If the agents are negotiating by multi-cast then for each variable negotiated, an agent can record the complete negotiation history. Let a *negotiation history* for a particular variable v, h_v, have the following form.

For each: $\alpha \in \mathcal{D}_v$,

$$h_v(\alpha) = [\text{offer}_\alpha(v, t_1), \text{offer}_\alpha(v, t_2), \ldots, \text{offer}_\alpha(v, t_{\text{final}})]$$

, where $\text{offer}_\alpha(v, t_i)$ is the set of options proposed by agent α for variable v in round t_i of the negotiation.

From a small set of negotiation histories it is not possible to separate completely the concepts of option availability, preferences and strategy, in order to understand an agents behaviour with certainty Consider the following situation. Agents α and β have been negotiating for a couple of rounds. Agent α offers o_k and in the next round agent β offers o_k. Agent β's offer could have been made for many reasons. Here are just a few: (i) agent β prefers o_k and has it available, (ii) agent β does not prefer o_k but it is available, and β is compromising according to his strategy because a certain number of rounds have passed, (iii) β is agreeing to o_k because apart from the options he has already offered, he has no other options available.

In real-world domains it may be the case that two agents will negotiate with each other quite a small number of times. For instance, in the meeting scheduling domain, 2 agents would have to meet approximately twice a week in order for them to have 100 negotiations in a year. As such, it is important for to explore methods that perform well with limited training data.

Probabilistic Agent Model

Given the difficulty of learning precise models of what drives another agent's behaviour, we instead propose using a *simple probabilistic model*.

In order to model the preferences and option constraints of an agent, α, we use a set of belief vectors B_α. Let $b_\alpha(o) \in B_\alpha$ be a *belief vector* over a set of simple disjoint categorizations C for each option o (one belief vector for each option). In our experiments we focus on use-once per agent options and hence use the following categorizations:

- (i) available and preferred;
- (ii) available but not preferred;
- (iii) not available.

Each belief vector $b_\alpha(o)$ represents the probability we give to o belonging to each of the disjoint categories given the observed data.

Updating the Agent Model

In order to update the model, we need to make some assumptions about how the options an agent proposes relate to the option categories that we have defined in the belief vector. One way to do this is to hypothesize a *conditional probability distribution* which, for each category, gives the probability that the agent proposes an option, given that the option fits the category. If we let $c \in C$ be a category then this is the conditional probability distribution given by $Pr(\alpha \text{ proposes } o \mid o \in c)$.

Using the probability distribution, when the agent proposes an option o and this is observed by the learning agent,

the learning agent can use Bayes Rule to update its current belief vector for that option, $b^t_\alpha(o)$, as follows:

$$b^{t+1}_\alpha(o)[c] = \frac{Pr(b^t_\alpha(o)[c])Pr(\alpha \text{ proposes } o | o \in c)}{\sum_{i \in C} Pr(b^t_\alpha(o)[i])Pr(\alpha \text{ proposes } o | o \in i)}$$

Dealing with options that are never offered We can set an initial belief vector $b^0_\alpha(o)$ for each option o. If we only update the belief vectors of options that we observe α proposing, then for each option o which α never proposes we will always have belief vector $b^0_\alpha(o)$. Depending on the semantics of our disjoint categorization we can handle this in a number of ways. If we are using the categorization previously described, it may make sense to to initialize the belief vectors such that "not available" has the highest probability, followed by "available but no preferred" and then "available and preferred". In doing so we are building in the assumption that options which are never offered are more likely to be "unavailable" than "available" and if they are "available" are more likely to be not "preferred". Alternatively we could augment the update process. For instance we could hypothesize various conditional probabilities e.g., the probability that option o is in category c given that more than r rounds of negotiation have occurred and o has not been offered. The approach taken largely depends on the degree of domain knowledge available.

Using additional domain information In some cases there may be additional domain information available that can help the learner update the model. This is particularly the case where there are *semantic relationships between the different options*. For instance, at some universities, a particular class is generally held Monday/Wednesday or Tuesday/Thursday, at the same time on both days. As such, if the learner believes that agent α is unavailable at 10am on Tuesdays, it may want to increase its belief that α is unavailable at 10am on Thursdays. Another example is, if the learner knows agent α likes to meet before mid-day on Monday, Tuesday and Wednesday, but not in the afternoons on those days, the agent could use domain knowledge to reason that the agent prefers morning meetings to afternoon meetings and those adjust its beliefs for times on the other days.

Reasoning with the Learned Model

Exactly how the agent uses the models it has learned depends a lot on properties of the domain. Given the agents' aim to satisfy their preferences and reach agreements quickly, it is likely the disjoint categorizations will be chosen in a way that distinguishes between available and unavailable, and preferred and not preferred. In other words, the category an option falls into has implications for how likely the agent is to agree to assign that option to a variable. Assuming the disjoint categories are "preferred and available", "not preferred, but available" and "unavailable" as before, we illustrate how an agent can reason using its beliefs.

An agent can use its beliefs B_α about the categories options fall into for a particular agent, α, in order to reason about which options α is most likely to agree to. In particular, if the following inequalities hold:

1) $b_\alpha(o_1)$[preferred & available]
   $> b_\alpha(o_1)$[¬preferred but available]
2) $b_\alpha(o_1)$[preferred & available] $> b_\alpha(o_1)$[not available]
3) $b_\alpha(o_2)$[¬preferred & available]
   $> b_\alpha(o_2)$[preferred & available]
4) $b_\alpha(o_2)$[not preferred & available] $> b_\alpha(o_2)$[not available]

then the agent can reason that α is more likely to agree to o_1 than o_2. Exactly how the agent uses this sort of ranking information is dependant on the agent's utility function and strategy for negotiation. One possible way is to help the agent to decide which options to offer when it is indifferent. The learned model could also be used to evaluate whether or not it is worth the agent 'holding out' for a preferred time. For instance, if the agent believes α is unavailable for all of its preferred times, then if the agent cares about negotiation cost, it should probably compromise straight away.

Experimental Domain: Multiagent Meeting Scheduling

Multiagent Meeting Scheduling is a real-world EIMAPP domain. We can represent it in the EIMAPP terminology as follows:

- The set of agents, $\mathcal{A}$, contains one agent to represent each computer user that schedules meetings.
- The set of options $\mathcal{O}$ is a set of time slots.
- The variables are the meetings that arise over time.
- For each variable, v, the agents $\mathcal{D}_v$ are the meeting participants.
- Options are assign-once per-agent.
- Preferences over options (times) and negotiation costs are set according to each agent's user's preferences and these preferences are privately owned by each agent/user.

Negotiation

In our experiments we use a parametrized negotiation strategy, Offer-k. Agents using the strategy offer a set number, k, of available times (they have not previously offered) every negotiation round. Negotiation is done through an initiator agent. An agent will agree to an offer it receives (by offering the same time in the next round) if the time is available and the agent's preference for the time exceeds some acceptance threshold. The agent will also agree to an offer according to the paremater settings of the strategy. The strategy has the following parameters that determine when an agent will agree to an offer:

- number of rounds after which the agent will compromise i.e., agree to a time that falls below some preference threshold;
- number of different time proposals (made by the agent) after which the agent will compromise;

- number of rounds after which the agent will agree to bump a meeting;
- number of different time proposals after which the agent will agree to bump a meeting.

These parameters can be set according to the user's preferences.

Modeling Other Agents

In order to model other agents we use the simple disjoint categorization of options described in the previous section, namely we model times as:

- (i) available and preferred;
- (ii) available but not preferred;
- (iii) not available.

The model is not too fine grained. This is important, because any two agents will schedule meetings with eachother relatively infrequently meaning that we need to be able to learn the model from a very limited amount of training data.

Evaluating Agent Performace

There are two main factors that determine an agent's performance. The quality of the schedules it negotiates for it's user and the cost it incurs in the negotiation.

The quality of the schedule is measured according to user's preference function over times. To derive a value for a schedule we simply calculate the user's preference for having a meeting at each of occupied times in the schedule. Negotiation cost can be measured in terms of the number of rounds. Negotiating for a small number of rounds might be important to a user for a number of reasons. Firstly, the fewer times offered, the less information the agent is revealing about user's schedule. Secondly, a human, not a computer agent, may be receiving the agent's negotiation messages. Thirdly, it might be important to the user to compromise after a certain amount of time, and so forth.

Exactly how schedule quality and negotiation costs are weighed against eachother (note that it is easy to negotiation quickly by simply accepting any time) is determined by the user's utility function.

Improving Performance Using Model Learning

As the agent negotiates with other agents, it is possible for the agent to learn models of the other agents online and use these models to negotiate more efficiently without sacrificing schedule quality. In particular, if an agent has an accurate learned model of the form we have described, it can, where possible, offer times that it likes *and* that other agents also like and/or are available for.

Everytime the agent receives an offer it updates its beliefs as described in the previous section. When the agent makes its own time proposals it uses the learned models to help it select the times. At each round, there is some set of times the agent could propose according to the history and the parameters of negotiation strategy. In the case where the agent does not have a learned model these times are ranked according to the offers that have been received. Suppose our agent is the initiator of a four person meeting. If time t_1 has been proposed by two of the agents then it would have a higher rank than a time proposed by only one of the agents. In the case where there are learned models we have further information we can use to produce a ranking of times. In our experiments we added three points to a time's score if one agent had proposed it, two if it was categorized by the model as preferred and available, and one if it was categorized as available but not preferred. We then sorted the times according to their scores, and selected the top k times to offer.

Experimental Results

In this section, we show some results from our simulations in the Multiagent Meeting Scheduling domain. We have simulated meetings being scheduled amongst a group of agents over a series of weeks. In each trial some times in each agent's calendar are selected at random to be blocked to represent unavailability caused by commitments outside the multiagent system. In each trial, a set of initial meetings (these meetings range in size, from two person meetings, to meetings involving all agents), are scheduled at randomly selected time slots. These meetings are constant across trials, but the slots in which they are scheduled change. We refer to an agent's blocked time slots, and initial meetings as the agent's base calendar. In each trial the agents schedule a set of new meetings. Each time a certain number of meetings from the new set, e.g., 10, is scheduled, all the agents' calendars are reset to their initial states (the base calendars) before the agents continue. This resetting simulates the regular meetings that most people have, and the new meetings that are scheduled each week.

We have used the same hypothesized conditional probability distribution (needed to do the belief updates) in all our experiments - $P(\alpha \text{ proposes } t | t \text{ is preferred and available}) = 0.9$, $P(\alpha \text{ proposes } t | t \in \text{ is preferred and not available}) = 0.5$ and $P(\alpha \text{ proposes } t | t \in \text{ is unavailable}) = 0.1$.

Learning the Model

Figures 2 and 3 demonstrate that using our approach agents can learn models with a high degree of accuracy. In these experiments, 5 agents scheduled 150 new meetings together and we evaluated the average accuracy of one agent's models. We ran 100 trials and the standard deviation was negligible. Figures 2 and 3 show the average proportion of times the learned model labeled correctly. A time is considered to be labeled correctly if the category with highest belief in the learned model corresponds to agent's true preferences and base calendar. For example, if a time is available in β's base calendar and is preferred by β, then it is correctly labelled by the learned model if the highest belief is on *preferred and available*. Note that even if we have a perfectly accurate model it will not always reflect the current status of β's calendar since base calendar meetings can be moved, and new meetings are scheduled as the simulation runs.

The Figures 2 and 3 also show the average proportion of times that are *close to correct*. We define the belief to be close to correct when it is incorrect, but has not mistaken a

preferred and available time for an unavailable time, or vice versa. Figure 2 shows that when we start with a uniform belief over the categorizations we get an average accuracy of almost 80%. We can improve this results simply by biasing the inital belief towards the categorizations that imply we are less likely to recieve offers (unavailable, available but not preferred). Figure 3 shows that with this bias we get an average accuracy of 86%.

Misclassifications occur for a number of reasons. Firstly, suppose two agents are negotiating. If there is some time that both agents are available for, but neither prefers, it is unlikely to be offered (unless the agents are very busy). As such each agent might believe the other is unavailable for the time. However in a case like this, since neither agent wants to meet at this time, it is not particularly important these two categories are confused. Secondly, suppose two agents are negotiating and one agent is very busy. There may be some times that the busy agent regularly offers that it is available for but does not prefer (because the busy agent has no preferred and available times left for instance). It is likely that the other agent will believe these are preferred and available times for the busy agent. This is a misclassification, but since the busy agent is willing to schedule meetings at these times, again, the misclassification is not particularly harmful.

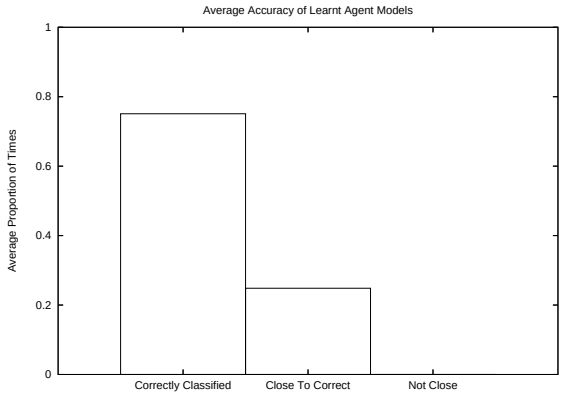

Figure 2: The graph shows the average accuracy of the learned models starting from an unbiased belief.

Improving Performance

Figure 4 shows the significant reduction achieved in the average number of negotiation rounds through learning. In this experiment, we ran 100 trials and the agents both offered 1 timer per negotiation round. There were 5 agents in the system, and the initial meetings involved all 5 agents, but all the new meetings were only between two of the agents - the agent we were evaluating and one other. These two agents had a small overlap in preferences at midday. One agent liked morning times, the other afternoon times, but both liked midday. Despite the fact that the agents we were evaluating learned online, there was a significant performance improvement from learning after only 20 meetings were scheduled. After 80 meetings were scheduled the average number

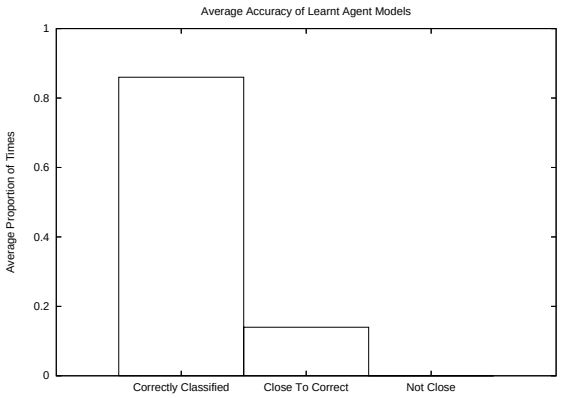

Figure 3: The graph shows the average accuracy of the learned models starting from an biased belief.

of negotiation rounds when both agents were learning was *almost half* that of the average number for the no learning case. Figure 4 shows that performance was slightly better for the case where both the agents learned, than the case where just one agent learned. The decrease in scheduling cost achieved through learning comes at no cost to schedule quality, so this reduction in cost represent and overall improvement in performance.

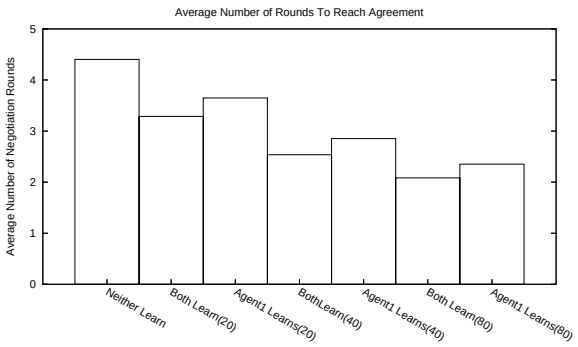

Figure 4: The graph shows the average number of rounds it took to agree upon meeting times between 2 agents after 20, 40 and 80 meetings had been scheduled. The agents had a small overlap in their preference functions.

We found that the improvements were less pronounced for random preferences. Figure 5 shows a graph from an experiment where the agents had randomly assigned preferences, with a 20% probability of a time being preferred. Nonetheless, the reduction in the average round count is almost 25%. In general, we found that the performance improvement was most pronounced when there were a lot of potential options each agent could offer, but only a few options that were agreeable to all agents.

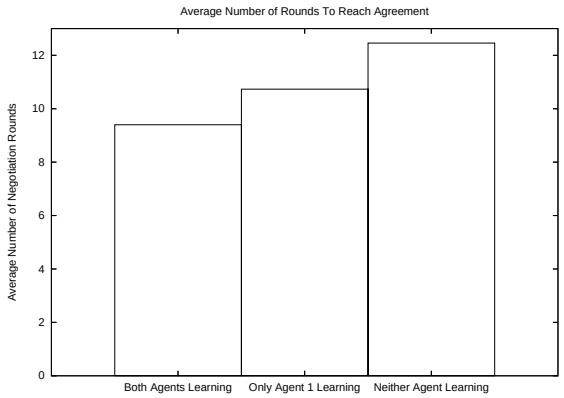

Figure 5: The graph shows the average number of rounds it took to agree upon meeting times between 2 agents after 20 meetings had been scheduled. The agents' preferences were randomly assigned.

Related Work

Bayesian methods have previously been applied to the problem of preference learning in negotiation. Bui et al., (Bui, Kieronska, & Venkatesh 1996) apply Bayesian learning to the problem of learning others' preferences in a negotiation setting and use meeting scheduling as their test domain. The setting they explore is quite different to the one we focus on this paper however. The agents are assumed to be cooperative, and negotiate by truthfully revealing their preferences for different negotiation outcomes. This gives the Bayesian learning algorithm correctly labelled training data to work with. In our setting the agents are self-interested as opposed to cooperative, and there are no labelled training examples. Buffett and Spencer (Buffett & Spencer 2005) apply Bayesian preference learning to the related setting of bilateral multi-issue negotiation. They utilise the assumption that agents increasingly concede as time passes.

A variety of approaches to the multiagent meeting scheduling problem have been proposed in the last 15 years, including negotiation based approaches (Sen & Durfee 1998) (Garrido & Sycara 1995), Distributed Constraint Reasoning approaches (Modi & Veloso 2005), and Market-based approaches (Ephrati, Zlotkin, & Rosenschein 1994). Most of the negotiation approaches assume the agents are cooperative, but Crawford and Veloso (Crawford & Veloso 2007) look at negotiation from a more strategic perspective. In particular, they apply experts style algorithms to the problem of learning to select negotiation strategies . They show that an agent can learn online what negotiation strategies work well with particular agents. An interesting direction for future work would be to combine an approach that learns to select strategies with a model-learning approach.

Conclusion

In this paper, we defined the Extended Incremental Multiagent Agreement Problem with Preferences. EIMAPPs reflect many real-world agreement problems including multiagent meeting and task scheduling. We showed how agents can solve EIMAPPs through negotiation and learn models of each other's preferences and constraints online from very limited information. We also showed that agents can use these models to significantly reduce the number of rounds required for negotiation, without having to compromise their own preferences

References

Buffett, S., and Spencer, B. 2005. Learning opponents' preferences in multi-object automated negotiation. In *ICEC '05: Proceedings of the 7th international conference on Electronic commerce*, 300–305. New York, NY, USA: ACM Press.

Bui, H. H.; Kieronska, D.; and Venkatesh, S. 1996. Learning other agents' preferences in multiagent negotiation. In Shrobe, H., and Senator, T., eds., *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, Vol. 2*, 114–119. Menlo Park, California: AAAI Press.

Crawford, E., and Veloso, M. 2007. An experts approach to strategy selection in multiagent meeting scheduling. In *To Appear, Journal of Autonomous Agents and Multiagent Systems, Special Issue on Multiagent Learning*.

Ephrati, E.; Zlotkin, G.; and Rosenschein, J. 1994. A non–manipulable meeting scheduling system. In *Proc. International Workshop on Distributed Artificial Intelligence*.

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

Modi, P. J., and Veloso, M. 2005. Bumping strategies for the multiagent agreement problem. In *Proceedings of Autonomous Agents and Multi-Agent Systems, (AAMAS)*.

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