Preferences in Semi-Cooperative Agreement Problems

Elisabeth Crawford and **Manuela Veloso**
Computer Science Dept., Carnegie Mellon University,
Pittsburgh PA, 15213

Abstract

We define the Extended Incremental Multiagent Agreement Problem with Preferences (EIMAPP). In EIMAPP, variables arise over time. For each variable, a set of distributed agents receives reward for agreeing on which option to assign to the variable. Each of the agents has an individual, privately owned preference function for choosing options. EIMAPPs reflect real world multiagent agreement problems, including multiagent meeting scheduling and task allocation. We analyze negotiation in EIMAPP theoretically. We introduce *semi-cooperative agents*, which we define as agents with an increasing incentive to reach agreements as negotiation time increases. Agents necessarily *reveal* information about their own preferences and constraints as they negotiate agreements. We show how agents can use this limited and noisy information to learn about other agents, and thus to negotiate more effectively. We demonstrate our results experimentally.

Introduction

In Agreement Problems, multiple parties receive reward for reaching agreement on some issue. Agreement problems are generally solved via negotiation. Some examples include: (i) the exchange of availability information to schedule meetings; (ii) negotiation about role or task assignments; and (iii) a sports team agreeing on a game plan.

Automating agreements has been studied from multiple directions, including centralized approaches, e.g., (Ephrati, Zlotkin, and Rosenschein 1994), cooperative distributed constraint satisfaction approaches e.g., (Wallace and Freuder 2005) and game-theoretic negotiation approaches, e.g., (Endriss 2006). In negotiation, particularly involving both humans and automated agents, it has been shown that some cooperation is beneficial, and expected by the human negotiators (Grosz et al. 2004). In this paper, we study agreement problems where variables arise over time, preferences are privately owned, and where agreements are negotiated.

Agreement problems are neither fully cooperative (agents have individual preferences), nor fully adversarial (agents need to reach agreement in order to receive reward). We capture this "middle-ground" as *semi-cooperative* behavior, in which the agent's utility relaxes its faithfulness to the user's preferences as a function of the negotiation round. In our approach, the discount in the utility function acts as an increasing incentive to compromise as time increases. Our approach contrasts to existing work on agreement problems that focuses on either fully co-operative or simply rational behavior in the traditional game theoretic sense.

We further focus on the private aspect of the distributed agreement problem. We define preferences and assignments as privately owned. However, as negotiation proceeds, agents necessarily reveal information about their preferences and properties. We use the incremental partial disclosure of privately owned information, to learn to adapt an agent's negotiation to the other agents. In this paper, we contribute a learning algorithm which uses negotiation histories to model the behavior of other agents. The negotiation histories are challenging and noisy, in particular because of the impact of multiple agents adapting at once. We show how our learned models can be successfully exploited to increase the agent's utility while continuing to learn online.

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Agreement Problems

We define the Extended Incremental Multiagent Agreement Problem with Preferences (EIMAPP). In EIMAPPs, variables, v, arise over time. For each variable, the set of decision makers, $\mathcal{D}_v$, receive reward if they agree upon an option, o, from the set of all options, $\mathcal{O}$, to assign to v. Each agent has a privately owned preference function that specifies its payoff for assigning v to each option. We build upon the Incremental Multiagent Agreement Problem (IMAP) Modi & Veloso (2005). We extend IMAP to cover a wider class of values/options and to include preferences.

Definition ***The Extended Incremental Multiagent Agreement Problem (EIMAP)*** consists of:

- A set of agents $\mathcal{A}$.
- A set of values/options $\mathcal{O}$.
- A set of variables $\mathcal{V}$, not provided up-front.
- For each variable $v \in \mathcal{V}$, $\mathcal{D}_v \subseteq \mathcal{A}$ is the set of agents that decide v. To decide v each of the agents in $\mathcal{D}_v$ must agree to assign a particular option $o \in \mathcal{O}$ to v. When the agents $\mathcal{D}_v$ decide a variable they each receive some reward according to their utility function (defined later).

Preferences in EIMAP

IMAP (Modi and Veloso 2005), does not include a built-in notion of preferences. Nonetheless, in many domains, agents have preferences about which options are assigned to variables. For instance, in a multi-robot task domain, some options for completing a task may be more expensive to a robot than others.

Definition ***EIMAP with Preferences (EIMAPP)*** is an instance of EIMAP where each agent $a \in \mathcal{A}$ has a function $\text{pref}_a \colon \mathcal{O} \times \mathcal{V} \rightarrow \Re^{+}$. The function indicates a's preference for assigning o to a variable v. In this paper we look at the case where $\text{pref}_a(o, v) = \text{pref}(o, v') \; \forall v' \in \mathcal{V}$

We do not include a specific *goal* in the problem definition, since the appropriate goal depends on the agent model.

To represent some real-world problems, it may be necessary to add restrictions to the EIMAPP definition. Currently, any option can be used to decide any variable. Agents a_1 and a_2 could assign o_1 to v_1 and o_1 to v_2. To represent domains such as meeting scheduling where resources are limited we can add restrictions on option usage to represent constraints, e.g., in one week an agent can't assign the option "Monday 9am" to more than one variable.

Privately Owned Information

In EIMAPP domains certain information is 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 is communicated.

Variables: Only the agents $\mathcal{D}_v$ need to know v exists to assign it an option. In some domains it is undesirable for all agents to know of all the variables, e.g, meeting scheduling.

Assignments: Only the agents $\mathcal{D}_v$ need to know which option, o, is assigned to v.

Preferences: Privately owned by the agents.

Negotiation

To decide a variable v, the agents, $\mathcal{D}_v$ negotiate until they reach an agreement by exchanging offers of options. A *negotiation protocol* is a public set of rules, governing the mechanics of the negotiation. Our protocol operates in rounds, and offered options remain on the table. Every round the initiator of the variable, offers a set of options to the other decision makers. The decision makers respond by sending a set of options back to the initiator . The process ends when there is an intersection in the offers. Algorithm 1, shows an un-restricted form of the protocol.

Algorithm 1 Un-restricted negotiation protocol

```
for each $a$ in $\mathcal{D}_v$, let $\mathcal{O}_a$ be the set of options $a$ has offered
if $\bigcap_{\forall a \in \mathcal{D}_v} \mathcal{O}_a = \emptyset$ then
    initiator, $i$, offers $\mathcal{O}_i^r \subset \mathcal{O}$ to the other agents in $\mathcal{D}_v$
    $a \in \{\mathcal{D}_v - i\}$ offers $\mathcal{O}_a^r \subset \mathcal{O}$ to the initiator
else
    intersection $= \bigcap_{\forall a \in \mathcal{D}_v} \mathcal{O}_a$
    initiator selects an $o_c$ from intersection to assign to the variable and notifies all $a \in \mathcal{D}_v$
```

The protocol allows for a wide variety of agent behavior. e.g., agents can send each other empty offers, offer all options, etc. A common protocol restriction is that the agents monotonically concede, e.g., we could require that at least one new option is offered each round. The choice of protocol affects what privately owned information is revealed, and thus the difficulty of learning about other agents. By restricting agent behavior, protocols can reveal with certainty some of an agent's privately owned information. For instance, Modi & Veloso (2005) require agents to accept any proposal that does not violate a constraint. In their meeting scheduling domain, when an agent does not agree to a time, it has a meeting. Wallace & Freuder (2005) also require agents to agree to any feasible proposal. Our protocol does not place agreement requirements of this type upon the agents. Others, e.g., (Jennings and Jackson 1995; Bui, Kieronska, and Venkatesh 1996) require agents to reveal their preferences for the options they propose. Bui et al., (1996) use a Bayesian learning to learn the preferences of other agents, using meeting scheduling as their test domain. The agents are assumed to truthfully reveal their preferences for different options. This gives the learner correctly labeled training data to learn the preference function of the other agents.

At the other end of the spectrum, some researchers have focused on privacy. For example, in the context of Distributed Constraint Optimization (DCOP), Yokoo, Suzuki & Hirayama (2005) used cryptographic techniques to preserve privacy. However, even if a cryptographic approach is used to obscure the agreement process, the agents gain some information from seeing the options that are agreed upon.

Negotiation Strategies

A ***Negotiation Strategy*** is an algorithm that subject to a particular negotiation protocol, determines the agent's actions. In our protocol negotiation strategies decide what options to offer in each round. Negotiation strategies are privately owned. There is a large range of negotiation strategies. [1] We will analyze preference based strategies.

Definition ***1-Preference-Order Strategy***: Offers one option per round. If $\text{pref}_a(o_i) > \text{pref}_a(o_j)$, o_i offered before o_j. If $\text{pref}_a(o_i) = \text{pref}(o_j)$ then a offers the option already offered by the largest number of agents, tie-breaking in favor of the earliest offered option (further tie-breaking is random). Where possible, a Pareto-Optimal option is selected from the intersection by the initiator (assuming all agents use the strategy).

A typical performance criteria is whether a negotiation strategy leads to a *Pareto Optimal* outcome in self-play.

Definition ***Pareto Optimality in terms of Preferences***: An outcome o_i is Pareto Optimal for v if there does not exist another outcome o_j such that for some $a \in \mathcal{D}_v$ $\text{pref}_a(o_j) > \text{pref}_a(o_i)$ and $\forall b \neq a \in \mathcal{D}_v$ $\text{pref}_b(o_j) \geq \text{pref}_b(o_i)$.

[1] If we require agents to offer a new option each round, the space of negotiation strategies, which don't depend on the offers of other agents, is every ordering of options, i.e., for k options there are $k!$ distinct strategies.

Theorem 1. *Let there be two agents a and b, who both use the 1-Preference-Order Strategy, then the outcome o_c is Pareto Optimal in terms of preferences.*

Proof. Let r_c be the agreement round. For a contradiction, suppose $\exists o_p$ such that $\text{pref}_a(o_p) > \text{pref}_a(o_c)$ and $\text{pref}_b(o_p) \geq \text{pref}_b(o_c)$. **Case 1**: Let a be the initiator. Since $\text{pref}_a(o_p) > \text{pref}_a(o_c)$, a must have offered o_p prior to r_c. If $\text{pref}_b(o_p) > \text{pref}_b(o_c)$ then b must have offered o_p prior to r_c. If $\text{pref}_b(o_p) = \text{pref}_b(o_c)$ then b must also have offered o_p prior to r_c (tie-breaking rule). Hence o_p would be the agreement. **Case 2**: Let b be the initiator. If $\text{pref}_b(o_p) > \text{pref}_b(o_c)$ the same argument applies. If $\text{pref}_b(o_p) = \text{pref}_b(o_c)$ then b may offer o_c before o_p, but a will still offer o_p before o_c. This could result in both o_c and o_p appearing in the intersection. However, b knows that a is either indifferent or prefers o_p, so by the Pareto-Optimality condition in the strategy, b will select o_p as the agreement. $\square$

For two agents (offering one option a round) preference order leads to a Pareto Optimal outcome. We can extend the result to agents *offering k options a round*, if we require that agents *don't offer options of different preference in the same round*. For n agents, we run into difficulties, since the option intersection can contain options that are Pareto Dominated.

Lemma 2. *When n agents negotiate using the 1-Preference-Order Strategy the intersection can contain a Pareto Dominated option, which is indistinguishable to the initiator from a Pareto Optimal option (Crawford 2008).*

Corollary 3. *The outcome of n agents using the 1-Preference-Order Strategy is not always Pareto-Optimal.*

This problem could be avoided by using a protocol where only the initiator proposes options and the others just accept/reject. This could be very slow however. A strategy that offers options in preference order makes sense if the agent only cares about preferences. However, in many agreement problems agents have concerns about negotiation time. In the following section we address this by defining semi-cooperative agents and utility functions.

Semi-Cooperative Agents

In general, agreement problems are neither fully cooperative nor fully adversarial. We note that if an agent compromises, by offering an option out of preference in order, we can't guarantee Pareto Optimality.

Lemma 4. *If an agent offers an option out of order of preference, an outcome that is not Pareto-Optimal (in terms of preferences) can result.*

Proof. Suppose there are two agents, a, b and a offers o_c out of order to agree with b. Then, $\exists o_p$, such that $\text{pref}_a(o_p) > \text{pref}_a(o_c)$. Suppose $\text{pref}_b(o_c) = \text{pref}_b(o_p)$, then o_c is not Pareto Optimal. A similar argument works for $n > 2$ agents. $\square$

In many agreement problems preferences are not the only consideration - negotiation efficiency is important. For example a personal assistant agent may need to negotiate electronically with humans as well as with computer agents. We define *semi-cooperative agents* to address the dual concerns of preference satisfaction and cooperation in agreement problems. We model the agents as having a self-interested component to their utility - the reward according to the agents' preference functions. This reward is discounted over negotiation time (measured in rounds) to represent the agents' cooperative tendencies.

Definition ***Preference-Round Semi-Cooperative Utility*** Let agent a's utility from agreeing to assign option o to variable v in round r be given by:

$$U(o, v, r) = \gamma^r \text{pref}_a(o, v), \gamma \in (0, 1)$$

Agents with a high cost for negotiation time (low γ) are incentivized to compromise earlier in the negotiation (by agreeing to an offered option) than agents with a high γ. Due to privacy considerations, it is undesirable for agents to offer many options in order to reach agreement quickly. We assume the agents have a maximum number of options they can offer per round (e.g., set by the agent's user). In our experiments this maximum is set to one, and we use the protocol restriction where agents offer one option per round.

Definition ϵ ***Semi-Cooperative Behavior*** For a variable, v, agent a selects an option o from $O_{\text{un-Offered}_a}$ (the options it has not yet offered for v) to offer in round r as follows.

$$o_v(r) = \begin{cases} \underset{o \in O_{\text{un-Offered}_a}, r' > r}{\text{argmax}} \quad U(o, v, r') \text{ if } \exists o, r' : U(o, v, r') \geq \epsilon \\ \underset{o \in O_{\text{un-Offered}_a} \,\&\, \nexists o' \text{offered by more agents}}{\text{argmax}} \quad \text{pref}_a(o). \end{cases}$$

In other words, the agent offers the option it believes will maximize its utility, if the utility will exceed the parameter ϵ, for some r'. The round r' is the estimated (from learning) agreement round for the option. If the agent cannot achieve utility greater than ϵ, then it seeks to reach an agreement quickly, by selecting from amongst the options that have been offered by the most agents, the option it prefers.

Other models of semi-cooperativeness have been considered in the agents literature. Gal et al. (2004) look at modelling an individual agent's utility as a function of self-interest and others' interest. This works when the preferences of the agents are public knowledge. Manisterski, Katz & Kraus (2007) define semi-cooperativeness, not according to the agents' utility functions, but using other properties, e.g., they require Pareto Optimality in self-play. Zhang and Lesser (2007) also base their definition on agent properties, in particular they define an agent that is truthful and collaborative, but *not willing to voluntarily sacrifice its own interest* for others, as semi-cooperative. Negotiation with discount factors has been looked at in game theory, e.g., in the context of the split the pie game analyzed by Rubinstein (1982). Game theoretic analysis is most applicable when there is complete information or when the agents share accurate prior beliefs about type distributions.

Previously, we showed that when agents only cared about preferences, that if they offered an option out of preference order it could result in a Pareto Dominated outcome. When agents are semi-cooperative we get the following:

Lemma 5. *An agent may receive greater utility if it offers an option* **out of preference order**, *than in order, when it has a Preference-Round Semi-Cooperative utility function (Crawford 2008).*

To design algorithms for semi-cooperative agents and evaluate them, we need to appropriate performance criteria. Unfortunately, *achieving Pareto Optimality relative to semi-cooperative utility is not a realistically achievable when agents negotiate EIMAPPs*. To show this we need to define Pareto Optimality relative to semi-cooperative utility. Since the round and option both determine the agents' payoffs, we let an outcome be an option round pair, e.g. $(o_i, 2)$. Pareto Optimality is then defined as previously,using utility functions instead of preference functions.

Lemma 6. *For an outcome of the negotiation process to be Pareto Optimal in terms of Semi Cooperative utility it must be agreed upon in the first round.*

Proof. Let (o_i, r), s.t $r > 1$ be the outcome of negotiating v. Then by the definition of Semi Cooperative utility, $U_a(o_i, 1) > U_a(o_i, r) \forall a \in \mathcal{D}_v$. □

Since agent preferences and any constraints are privately owned, we can't guarantee Pareto Optimality. Instead, our agents aim to achieve high semi cooperative utility. Because the round of agreement influences utility, and depends on the behavior of other agents, we present an approach where agents learn online to improve their negotiation.

Learning to Maximize Estimated Utility

To compare the utility of offering different options, agents need a way to estimate when options will be agreed upon. The negotiation history contains relevant data.

Data

For each variable an agent negotiates, it can record the complete negotiation history.

Definition Let a *negotiation history* for a particular variable v, h_v, have the following form. For each: $a \in \mathcal{D}_v$,

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

, where $\text{offer}_a(v, t_i)$ is the set of options proposed by agent a for variable v in round t_i of the negotiation. If an agent is a decision maker for a variable, but not the initiator, then it only has the record of its own offers and the initiator's offers.

From the negotiation histories of all the variables it has negotiated, an agent can extract training data of the following form, for each of the other agents it negotiations with: $\text{data}_a = \{(\text{option,round})\}$, e.g., $\text{data}_a = \{(o_1, r_1), (o_2, r_10), (o_1, r_3), ..\}$.

Learning to Predict Rounds

To the data gathered the agent can apply a *regression algorithm* to predict for each option the round in which the agent is most likely to offer it, given the data (in the experiments we use Linear Regression). To estimate the round in which an option *will be agreed upon*, the *agent takes the maximum of the round estimates for each of the decision makers*.

One important consideration is how to handle options that are never offered. For a given number of decision makers and option set size, we can calculate the maximum number of rounds required (assuming the agents must offer a new option in each round). Initially we provide a training example for every option where the negotiation round is set to be this maximum. This represents the concept that the option is never offered. The first time an option is offered, the initial training example is removed from the set. For options that are never offered the *never* training examples remain.

Learning Online: Exploration

When an agent negotiates for the first time with another agent it has no way to estimate when that agent will offer options. Agents need to *learn online* as they negotiate. Learning online also allows the agent to adjust to changes in the other agent's behavior. In this paper we use a scheme where the agent *explores* with some probability, or instead *exploits* what it has learnt. The probability of exploration is decreased as the number of variables negotiated with an agent increases. When the agent explores, it uses the 1-preference order negotiation strategy. However it still uses the ϵ condition on utility and will seek to compromise if utility is going to fall below ϵ.

Exploiting the Learned Information

We consider the protocol restriction where the agent must offer one new option each round. When exploiting the learned information, the agent would like to select the option, o_w, that it estimates will maximize its utility. o_w has an associated predicted agreement round r_w. The agent would like to simply offer o_w and wait for it to be accepted (hopefully by round r_w). However, if the agent must offer a new option every round and there are no other options to offer of equal estimated utility, the agent will have to offer an option of lower estimated utility. As such, there is a danger that the option of lower estimated utility could become the agreed upon option. To try and avoid this, after offering o_w the agent can offer options in the order of highest estimated round first. If r_w is exceeded without an agreement being reached the agent needs to revise its utility estimate for o_w, and start to offer any options with utility equal to the new estimate. The process described here is the idea behind the Strategic-Semi-Cooperative-Learner. This process is summarized in Algorithm 2. Note that when a round estimate is required for an agent, if the agent has already offered the option this is taken into account instead of querying the regression learner for an estimate.

A drawback of the Strategic-Semi-Cooperative-Learner is that if the round estimates are inaccurate the agent could be wasting rounds. To address this, we also explore the use of a strategy we call Semi-Cooperative-Ordered-Learner (summarized in Algorithm 3). In this strategy, the agent offers options in order of estimated utility.

Each of the learning approaches described requires the agent to estimate the utility of offering the different options. To estimate the utility of offering option o_w, the agent queries the regression learner for the estimate of when each of the decision maker agent will offer o_w. Let the highest

Algorithm 2 Strategic-Semi-Cooperative-Learner: Select-Option(v)

```
First-Call: sort-by-est-util(unoffered$_v$)
$o_w$ = pop(unoffered), offer($o_w$)
Method:
sort-by-est-util(unoffered$_v$)
if util-est($o_w$)> util-est(top(unoffered$_v$)) then
    if util-est($o_w$) $< \epsilon$) then
        execute SC-$\epsilon$ behavior
    else
        cp = sort-by-highest-est-round(copy(unoffered$_v$))
        offer(remove(unoffered$_v$, top(cp)))
else
    if util-est(top(unoffered$_v$))$< \epsilon$) then
        execute SC-$\epsilon$ behavior
    else
        $o_w$ = pop(unoffered$_v$)
        offer($o_w$)
```

Algorithm 3 Semi-Cooperative-Ordered-Learner: Select-Option(v)

```
First-Call: sort-by-est-util(unoffered$_v$)
$o_w$ = pop(unoffered), offer($o_w$)
Method:
sort-by-est-util(unoffered$_v$)
if max (util-est($o_w$),util-est(top(unoffered$_v$))) $< \epsilon$) then
    execute SC behavior
else
    $o_w$ = pop(unoffered$_v$)
    offer($o_w$)
```

such round be r_w. This is the agent's estimate for when o_w will be agreed upon. To estimate the utility of offering o_w the agent then uses its own utility function with the values of o_w and r_w.

The overall online learning algorithm is summarized in Algorithm 4.

Algorithm 4 Online-Learning-Negotiator: receive-offer(o,a,v,r)

```
First-Call: explore = 1
data$_a$ $\cup$ $(o, r)$
if explore $>$ rand then
    1-Preference-Order.Select-Option(o,v)
else
    Exploit using Semi-Cooperative-Ordered-Learner or
    Strategic-Semi-Cooperative-Learner
if finished negotiating v then
    run regression on data
reduce-probability(explore,r)
```

Experiments and Results

We have conducted a large number of experiments in EIMAPP (without restrictions on option use) to determine the efficacy of our approaches . In the experiments presented, variables were negotiated between two agents. Thus, when we make comparisons to the 1-Preference-Order Strategy, we are comparing to an outcome that is Pareto Optimal in terms of preferences. Focusing on the two agent case here allows us to most readily analyze the effect of the exploitation strategies. However, we have also experimented with more agents and found the learning very successful. The number of options in all the experiments was 40. The graphs displayed don't show error bars since the standard deviation over the trials was insignificant.

Approaches Effectively Adapt in Self-Play

Figure 1 demonstrates both our approaches effectively adapt over time in self-play. In the experiment shown $\gamma = 0.3$ and the preference range of the agents was 40. Using the 1-Preference-Order strategy resulted in the agent receiving zero utility, however, our learning agents performed much better. The graph shows the utility of of Agent 1 increasing as the number of variables negotiated increases, relative to the non-learning approach (the graph for Agent 2 is similar). The problem of learning when another agent will offer options is non-trivial in the self-play, since the semi-cooperative agents are not consistent in when they offer options. This is because both agents are exploring, and seeking to compromise when they exploit. As such it is impressive how quickly the agents adapt. Adapting quickly online is important in domains like meeting scheduling, where two agents may have relatively few opportunities to learn about each other. For other γ values and different preference ranges we got similar results.

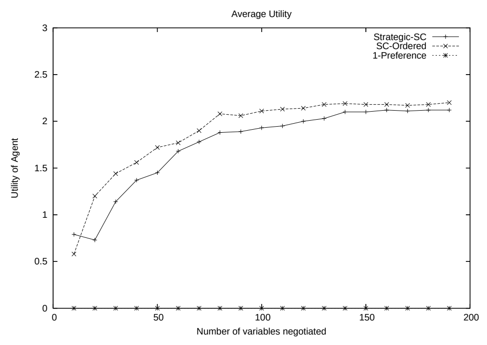

Figure 1: The Strategic-Semi-Cooperative-Learner and the Semi-Cooperative-Ordered-Learner increase the average utility they achieve over time in self-play

Effect of γ on Improvement Due to Learning

We examined the effect (Figure 2) different values of γ had on how much learning improved performance (as a percentage of the performance of the 1-Preference-Order strategy). For very low γ, $\gamma = 0.1$ we found that all the approaches achieved a utility of 0 when rounded to two decimal places. All the preferences in this experiment fell in the range 1 to 9. When γ was high for the agent we were evaluating (0.9), the percentage improvement was moderate (17 to 27%), regardless of whether the other agent had a high or a low γ. As the γ of the agent we were evaluating decreased to medium high (MHi$= 0.7$) and medium (MLow$= 0.3$) the percentage improvement increased dramatically, first to a little under 200%, and then to over 500%. The MLow γ setting highlights how much the learning and exploitation algorithms help when the agent cares about negotiation time. We didn't find a signifigant difference between the two exploitation approaches. The pattern described here was similar for other preference ranges.

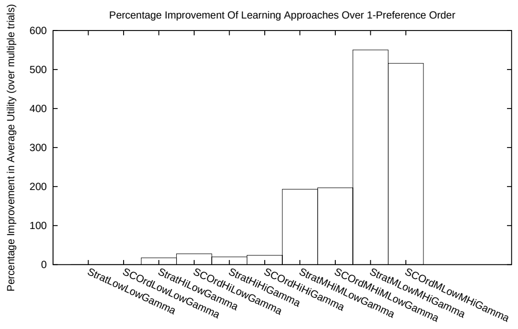

Figure 2: Percentage improvement of the Strategic-Semi-Cooperative-Learner(Strat) and Semi-Cooperative-Ordered-Learner(Ord) over the 1-Preference-Order strategy. Marker "HiLowGamma" indicates that the agent being evaluated had a high γ and the other agent had a low γ. γ values: Hi=0.9 MHi=0.7, MLow=0.3, Low=0.1. 200 variables were assigned and the std. dev across trials was negligible.

Discussion

The experiments presented are for the general case of EIMAPP (options are reusable). Without restrictions, EIMAPP can represent problems such as a sports team repeatedly picking team plays. In domains like meeting scheduling, where a time-slot can't be used again until the calendar is reset for the next week, we have also found learning to predict rounds improves performance.

In addition to evaluating the exploitation approaches in self-play, we have evaluated them against each other, and against the 1-Preference-Order strategy. The best results are achieved when both the agents learn. In the future we would like to evaluate against a wider variety of exploitation approaches. We have also looked at the case where more than two agents negotiate variables. For example, when a group of agents negotiated variables with up to 10 decision makers we found that learning online increased the performance of our agent approximately 4-fold.

Conclusions

In this paper we formalized agreement problems by defining EIMAPP, emphasizing the importance of preferences and the relevance of privately owned information. We introduced a definition of semi-cooperative agents in which agents become more cooperative as negotiation time increases. We contributed an approach for semi-cooperative agents to *learn to improve their negotiation* in agreement problems. We showed experimentally that our approach significantly outperforms not learning. The learning aspect of our work differs from previous work on learning in agreement problems, which has largely sort to learn explicit models of agents' properties and constraints.

Acknowledgements

This work was partly supported by DARPA under Contract No. FA8750-07-D-0185. The presented views and conclusions are those of the authors only. Thanks to the anonymous reviewers and to Andrew Carlson for helpful discussions regarding regression.

References

Bui, H. H.; Kieronska, D.; and Venkatesh, S. 1996. Learning other agents' preferences in multiagent negotiation. In *Proceedings of AAAI*.

Crawford, E. 2008. Improved negotiation in multiagent agreement problems through learning.

Endriss, U. 2006. Monotonic concession protocols for multilateral negotiation. In *Proceedings of AAMAS*.

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

Gal, Y.; Pfeffer, A.; Marzo, F.; and Grosz, B. J. 2004. Learning social preferences in games. In *Proceedings of AAAI*.

Grosz, B.; Kraus, S.; Talman, S.; Stossel, B.; and Havlin, M. 2004. The influence of social dependencies on decision-making: Initial investigations with a new game. In *Proceedings of AAMAS*.

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

Manisterski, E.; Katz, R.; and Kraus, S. 2007. Providing a recommended trading agent to a population: a novel approach. In *Proceedings of IJCAI*.

Modi, P. J., and Veloso, M. 2005. Bumping strategies for the multiagent agreement problem. In *Proceedings of AAMAS*.

Rubinstein, A. 1982. Perfect equilibrium in a bargaining model. *Econometrica* 50(1):97–109.

Wallace, R. J., and Freuder, E. C. 2005. Constraint-based reasoning and privacy/efficiency tradeoffs in multi-agent problem solving. *Artif. Intell.* 161(1-2):209–227.

Yokoo, M.; Suzuki, K.; and Hirayama, K. 2005. Secure distributed constraint satisfaction: reaching agreement without revealing private information. *Artif. Intell.* 161(1-2):229–245.

Zhang, X., and Lesser, V. 2007. Meta-level coordination for solving negotiation chains in semi-cooperative multi-agent systems. In *Proceedings of AAMAS*.