Negotiation in Semi-Cooperative Agreement Problems

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

Abstract

In this paper we introduce the Semi-Cooperative Extended Incremental Multiagent Agreement Problem with Preferences (SC-EIMAPP). In SC-EIMAPPs, variables arise over time. For each variable, a set of distributed agents gain utility for agreeing on an option to assign to the variable. We define semi-cooperative utility as an agent's privately owned preferences, discounted as negotiation time increases. SC-EIMAPPs reflect real world agreement problems, including meeting scheduling and task allocation. We analyze negotiation in SC-EIMAPPs theoretically. We note that 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 to negotiate more effectively. We demonstrate our results experimentally.*

1. 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., [5], cooperative distributed constraint satisfaction approaches e.g., [11] and game-theoretic negotiation approaches, e.g., [4]. In negotiation, particularly involving both humans and automated agents, it has been shown that *some cooperation is both beneficial, and expected by the human negotiators* [7]. 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 to receive reward). We capture this "middle-ground" as *semi-cooperativeness*, where the agent's utility relaxes its faithfulness to the user's preferences as a function of the negotiation round. This gives the agent an increasing incentive to compromise. Our approach contrasts to most existing work on agreement problems that focuses on either fully co-operative or simply rational agent behavior (in the traditional game theoretic sense).

We define preferences and assignments as privately owned by the agents concerned. However, in negotiating, agents necessarily reveal some 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 other agents. We show how agents can exploit learned models to increase their utilities while continuing to learn online. Furthermore, we show that each agent benefits when the other agents also adapt.

2. Related Work

Other models of semi-cooperativeness have been considered in the agents literature, e.g., Gal et al. [6] looked at modeling an agent's utility as a function of self-interest and others' interest. This is a good choice when the preferences of the agents are public knowledge. Manisterski, Katz & Kraus [8] define semi-cooperativeness, not according to utility functions, but use other properties, e.g., they require a Pareto Optimal outcome when identical agents negotiate. In Section 4, we show that in EIMAPP, when agents care about negotiation time, and have privately owned preferences, Pareto Optimality is not a suitable goal. Zhang and Lesser [13] 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.

Since our model of semi-cooperative utility is based on an agent's preferences and the negotiation length, we use learning, to estimate the rounds when the other agents will offer each option. We use a *negotiation protocol that does* *not reveal much information about the agents with certainty* because we consider agent preferences, etc., to be privately owned. Hence, we try to estimate the rounds directly, instead of building a preference and constraint model as others (e.g., [11, 1] have done in agreement problems. In a variant of the meeting scheduling problem, Wallace & Freuder [11] require agents to agree to any feasible proposal. This allows constraints to be induced. In the meeting scheduling domain, Bui et al. [1] require agents to reveal their preference for the options they propose. Assuming truthfulness they use Bayesian learning to build preference models.

Much of the work on learning in agreement problems has focused on learning preferences and properties of other agents, including work in buyer/seller domains e.g., Zeng and Sycara [12] and Coehorn and Jennings [2]. In contrast, we focus on learning about a specific aspect of the agents' *behavior*, and this is an aspect that changes over time as the agents explore and exploit what they have already learned.

3. Negotiated Agreement Problems

We define the Extended Incremental Multiagent Agreement Problem with Preferences (EIMAPP). In EIMAPPs, variables, v, arise over time. The decision makers, $\mathcal{D}_v$ negotiate to agree upon an option, o, from the set of all options, $\mathcal{O}$ to assign to v. The agents negotiate by proposing options. We build upon the Incremental Multiagent Agreement Problem (IMAP) [9], extending it to cover a wider class of problems, and to include agent preferences.

3.1 Agreement Problems and Negotiation

We start by introducing EIMAP.

Definition An **Extended Incremental Multiagent Agreement Problem (EIMAP)** is a tuple : $< \mathcal{A}, \mathcal{O}, \mathcal{V}, \mathcal{T} >$. Where:

- $\mathcal{A}$ is a set of agents.
- $\mathcal{O}$ is a set of values/options.
- $\mathcal{V}$ is a set of variables. For each variable, v, there is a set of decision maker agents $\mathcal{D}_v \in \mathcal{A}$, that decide which option $o \in \mathcal{O}$ to assign to v. Each variable has an *initiator* agent. The initiator agent i is an agent in $\mathcal{D}_v$ with a special job with regards to v. When v arises, i_v is notified, and it is $i_v's$ job to then start the negotiation. In the negotiation, i_v sends negotiation proposals to all the other agents in $\mathcal{D}_v$, collects their responses, and determines when the negotiation is complete.
- $\mathcal{T}$ is the distribution (and associated parameters) that determines the arrival of new variables over time. The variables in $\mathcal{V}$ are not provided up-front, giving rise to the incremental aspect of the problem.

3.2 Solving EIMAPs

We define the negotiation process through which EIMAPs are solved as a tuple, $\mathcal{N} =< \mathcal{R}, \omega, \mathcal{M}, \mathcal{H} >$, where $\mathcal{R}$ restricts how options can be assigned to variables, ω defines the negotiation process, $\mathcal{M}$ stores the mapping between variables and their agreed upon options, and $\mathcal{H}$ is the history of the agents' interactions. In more detail, we define the components of as follows:

- $\mathcal{R}$ is a function that can be used to restrict how options can be assigned to variables. We write $\mathcal{R}(a, \mathcal{O})$ to indicate a's eligible options. For instance, consider the multiagent meeting scheduling domain, where the variables are meetings, the options time-slots, and the decision makers the participants. We could define $\mathcal{R}(a, \mathcal{O})$: all $o \in \mathcal{O}$ s.t $(o, v') \notin \mathcal{M}, \forall v' : a \in \mathcal{D}_{v'}$, to say an agent can't have two meetings at the same time. An EIMAP with this particular restriction is similar to IMAP [9].
- ω is a tuple, $< \mathcal{N}_p, \mathcal{N}_{\text{range}}, \mathcal{N}_{\text{rule}} >$ where:
  - $\mathcal{N}_p$ is the negotiation protocol the agents, $\mathcal{D}_v$, use to assign an option o to the variable v when it arises. We use a round-based negotiation protocol (Algorithm 1), where once an option is offered, it remains on the table. Agreement is reached when there is an intersection in offered options. In each round the initiator of the variable sends a set of options (an offer) to the other decision makers. In the same "round", they respond with an offer to the initiator. The round ends when the initiator has all the responses. When there is an intersection in offered options, the initiator chooses which option to assign to the variable. In practice, it is necessary to add standard conventions, e.g., time-outs, to ensure the process cannot deadlock etc.
  - $\mathcal{N}_{\text{range}}$ is a set of integer pairs (s_a, e_a), one per agent. s_a is the minimum number of options a must offer per round, and e_a is the maximum. The minimum can only be violated if there are not enough eligible options according to $\mathcal{N}^a_{\text{rule}}$.
  - $\mathcal{N}_{\text{rule}}$ is a set of rules, one per agent, determining which options, in $\mathcal{R}(a, \mathcal{O})$, can be offered. For example, $\mathcal{N}^a_{\text{rule}}$: options offered for v in round r, must not have been offered by a for v before r.
- $\mathcal{M}$ is the mapping from variables to agreed upon options, $\mathcal{M} : \mathcal{V} \rightarrow \mathcal{O}$.

Algorithm 1 Negotiation protocol (v)

```
Let round $r = 1$
For each $a$ in $\mathcal{D}_v$, let offers$_a = \emptyset$ be the set of options $a$ has offered
while $\bigcap_{\mathcal{D}_v}$ offers$_a == \emptyset$ do
  initiator, $i$, offers $i_r \subset \mathcal{R}(i, \mathcal{O})$ to the other agents in $\mathcal{D}_v$, s.t $i_r \subset \mathcal{N}_{\text{rule}}$ and $s_i \leq |i_r| \leq e_i$.

  for each $o \in i_r$ do
    $\mathcal{H} = \mathcal{H} \cup (o, r, i)$
  offers$_i$ = offers$_i \cup i_r$

  for each $a \in \{\mathcal{D}_v - i\}$ do
    offers $a_r \subset \mathcal{R}(a, \mathcal{O})$ to the initiator, s.t $a_r \subset \mathcal{N}_{\text{rule}}$ and $s_a \leq |a_r| \leq e_a$.

    for each $o \in a_r$ do
      $\mathcal{H} = \mathcal{H} \cup (o, r, a)$
    offers$_a$ = offers$_a \cup a_r$
  $r+ = 1$
$x = \bigcap_{a \in \mathcal{D}_v}$ offers$_a$
initiator selects an $o_c$ from $x$ to assign to $v$ and notifies all $a \in \{\mathcal{D}_v - i\}$.
$\mathcal{M} = \mathcal{M} + (v, o_c)$.
```

- $\mathcal{H} = \bigcup_v \mathcal{H}_v$, where $\mathcal{H}_v$ contains a tuple for each offer made by $a \in \mathcal{D}_v$. The tuple contains the set of options offered, the agent name, and the round of the offer.

Definition A valid EIMAP solution is an assignment of a subset of $\mathcal{V}$ to options/values via a set of valid executions of the negotiation $\mathcal{N}$.

In general, every time a new variable, v, arises, the agents $\mathcal{D}_v$ execute the negotiation to *augment* the mapping, $\mathcal{M}$. The negotiation process produces the histories, $\mathcal{H}$, parts of which can be used to inform the negotiation of each agent.

3.3 EIMAP with Preferences

In many IMAP and EIMAP domains, the agents have privately owned preferences about which options are assigned to variables. For instance, in a multi-robot task domain, some options for completing a task (the variable) might be more expensive to a particular robot than other participants (decision makers). We introduce preferences into EIMAP, defining EIMAPP – $< \mathcal{A}, \mathcal{O}, \mathcal{V}, \mathcal{T}, \mathcal{P} >$.

Definition *EIMAP with Preferences, EIMAPP*, is an instance of EIMAP, augmented by a set of preference functions, one per agent: pref$_a$: $\mathcal{O} \times \mathcal{V} \times \mathcal{M} \rightarrow \Re^+$. The function, indicates a's preference for assigning o, to v. We focus mostly on the case where pref$_a(o, v, \mathcal{M})$ = pref(o).

3.4 Negotiating with Preferences

A ***Negotiation Strategy*** is an algorithm that subject to a negotiation protocol, (e.g., $\mathcal{N}_p$) and other restrictions (e.g., $\mathcal{N}_{\text{rule}}, \mathcal{N}_{\text{range}}$) determines the agent's actions. While the protocol is public knowledge, each agent's negotiation strategy is privately owned. One reasonable strategy, is to offer options in order of preference.

Definition *1-Preference-Order Strategy*: Offers one option per round. If pref$_a(o_i) >$ pref$_a(o_j)$, o_i offered before o_j. If pref$_a(o_i)$ = 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 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 $a \in \mathcal{D}_v$ pref$_a(o_j) >$ pref$_a(o_i)$ and $\forall b \neq a \in \mathcal{D}_v$ pref$_b(o_j) \geq$ pref$_b(o_i)$.

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 pref$_a(o_p) >$ pref$_a(o_c)$ and pref$_b(o_p) \geq$ pref$_b(o_c)$. **Case 1**: Let a be the initiator. Since pref$_a(o_p) >$ pref$_a(o_c)$, a must have offered o_p prior to r_c. If pref$_b(o_p) >$ pref$_b(o_c)$ then b must have offered o_p prior to r_c. If pref$_b(o_p)$ = 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 pref$_b(o_p) >$ pref$_b(o_c)$ same argument applies. If pref$_b(o_p)$ = 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 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. □

Lemma 2. *When n agents use the 1-Preference-Order Strategy the intersection can contain a Pareto Dominated option, indistinguishable to the initiator from a Pareto Optimal option (proof omitted due to space) [3].*

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.

4 Semi-Cooperative Agents

In general, agents don't only care about their individual preferences in agreement problems. However, if an agent compromises, by offering an option out of preference order, we can't guarantee Pareto Optimality.

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

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

In many agreement problems efficiency is important, e.g., when agents communice with humans, furthermore,users may want agents representing them to compromise as they would themselves. Grosz et al. [7] have demonstrated, that humans negotiating with computer agents expect co-operative behavior and that some co-operative behavior is beneficial to the agents.

We define *Semi-Cooperative EIMAPPs* to address the dual concerns of preference satisfaction and cooperation. 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 the negotiation rounds to represent the agents' cooperative tendencies.

Definition *Semi-Cooperative EIMAPP, SC-EIMAPP*, is an instance of EIMAPP augmented by a set of semi-cooperative utility functions and a semi-cooperative behavior rule, one for each agent: $< \mathcal{A}, \mathcal{O}, \mathcal{V}, \mathcal{T}, \mathcal{P}, U, \zeta >$:

- U is a set of utility functions, $U_a : \mathcal{O} \times \mathcal{V} \times I^+ \rightarrow \Re^+$, where I^+ is the positive integer round, r, at which the negotiation ends with the agreement of o for v. Specifically we consider U_a given by the following function:

$$U_a(o, v, r) = \gamma_a^r \text{pref}_a(o, v\mathcal{M}), \gamma_a \in (0, 1)$$

- ζ is a set of values, ϵ_a, one per agent, specifying the minimum estimated utility the agent will delay agreement to try to achieve.The *SC-behavior rule* defines ζ formally.

Agents with a high cost for negotiation time (low γ) are incentivized to compromise earlier (by agreeing to an offered option) than agents with a low cost for time (high γ).

Due to privacy considerations, it is not desirable for agents to simply offer many options to reach agreement quickly. Restrictions on the minimum and maximum number of options offered per round can be set by the user in $\mathcal{N}_{\text{range}}$. In our experiments, $\mathcal{N}^a_{\text{range}} = (1, 1) \forall a \in \mathcal{A}$.

Definition SC-behavior rule - Each agent has the following $\mathcal{N}^a_{\text{rule}}$: $o \in \mathcal{R}(o, a)$ is in $\mathcal{N}^a_{\text{rule}}$ for variable v at round r only if $\nexists r^- < r$ s.t $(o, a, r^-) \in \mathcal{H}^a_v$. Call this eligible set of options $\mathcal{E}$, a then selects its offer as follows:

$$o_v(r) = \begin{cases} \underset{o \in \mathcal{E}, r' > r}{\text{argmax}}\, U(o, v, r') \text{ if } \exists o, r' : U(o, v, r') \geq \epsilon_a \\ \underset{o \in \mathcal{E}}{\text{argmax}}\, \text{pref}_a(o) : \nexists o' \in \mathcal{E} \text{offered by more agents} \end{cases}$$

In other words, the agent offers the option it believes will maximize its utility, if this utility will exceeds ϵ_a, for some r'. The round r' is the estimated (from learning) round the option will be agreed upon. If the agent cannot achieve utility greater than ϵ_a, 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.

Negotiation with discount factors has been looked at in game theory, e.g., in the context of the split the pie game [10]. Game theoretic equilibrium 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 SC-utility function (proof omitted due to space) [3].*

To design algorithms for SC-agents and evaluate them, we need an appropriate performance criteria. Unfortunately, *achieving Pareto Optimality relative to SC-utility is not realistically achievable in SC-EIMAPPs.* Let an outcome be an option round pair, e.g. $(o_i, 2)$, then Pareto Optimality relative to SC-utility is defined as previously, but using utility functions instead of preference functions.

Lemma 6. *For an outcome to be Pareto Optimal in terms of SC-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. By the definition of SC-utility, $U_a(o_i, 1) > U_a(o_i, r) \forall a \in \mathcal{D}_v$. $\square$

Since agent preferences and any constraints are privately owned, we can't guarantee Pareto Optimality. Instead, we show how our agents aim to achieve high semi cooperative utility through learning.

4.1 Privately Owned Information

We consider preferences, negotiation strategies, and γ and ϵ values as privately owned by the individual agents, and variable assignments as privately owned by the decision makers. Sometimes, each agent's $\mathcal{N}_{\text{range}}$ and $\mathcal{N}_{\text{rule}}$ may also be privately owned. When agents represent users, it is important that automated agreement processes respect that a lot of information is not public by default. Our negotiation protocol does not require agents to declare their preferences, or to agree to any feasible option (revealing blocked options when they refuse). Using the parameters $\mathcal{N}_{\text{range}}$ and $\mathcal{N}_{\text{rule}}$ a user can control a lot about the information revealed. Nonetheless, a SC-agent reveals something about its behavior over time. SC-agents can use this information to adapt to each other and improve their utility.

5 Learning to Increase 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.

5.1 Data

For each variable the agent negotiates, it records the part of the negotiation history it sees. From these records, it extracts training data for each of the other agents it negotiations with: data$_a$ $= \{$(option,round)$\}$, e.g., data$_a$ $= \{(o_1, r_1), (o_2, r_10), (o_1, r_3), ..\}$. Depending on the domain, the agent may include more features in the training data.

5.2 Learning to Predict Rounds

To predict the rounds when another agent will offer each option, the agent can apply a regression algorithm to the training data. Then for a particular variable, with decision makers $\mathcal{D}_v$, the agent can predict when option o will be agreed upon, by taking the maximum of the predictions for the agents in $\mathcal{D}_v$. The agent can use the predicted round r to estimate its utility $U(o, v, r)$ for offering o.

An important consideration is how to handle options that are never offered. For a maximum number of decision makers and option set size, we can calculate the maximum number of rounds required (assuming the agents offer a new option in each round). Initially we provide a training example for every option where the round is this maximum, to represent the concept that the option is never offered. The first time an option is offered, the training example is removed. If an option is never offered, the training example remains.

5.3 Learning Online: Exploration

Since SC-EIMAPPs are incremental, agents need to *learn online*. This also allows agents to adjust to changes in other agents' behavior. When an agent negotiates a variable with another agent, it *explores* with some probability, or *exploits*. The probability of exploration decreases as the number of variables negotiated with an agent increases. When exploring, agents use the 1-preference order strategy, and without learning, the SC-behavior rule.

5.4 Exploiting the Learned Information

We consider the case where the agent offers one new option per round (according to its $\mathcal{N}_{\text{rule}}$). Let, o_w, be the option the agent 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 r_w). Since the agent must offer a new option every round it may have to offer an option of lower estimated utility. There is a danger that such an option could be agreed upon. To try and avoid this, the agent can start offering options in the order of highest estimated round. If r_w is exceeded, the agent needs to revise its utility estimate for o_w, and offer any options with that utility. This is the idea behind the Strategic-Semi-Cooperative (Algorithm 2).

Algorithm 2 Strategic-SC-Learner: Select-Option(v)

```
First-Call for $v$: sort-by-est-util(unoffered$_v$)
$o_w$ = pop(unoffered), offer($o_w$)
Subsequent Calls:
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($o_w$)
```

A drawback of the Strategic-SC-Learner is that if the estimates are inaccurate the agent could waste rounds. To address this, we also explore a strategy we call Semi-Cooperative-Ordered-Learner (Algorithm 3), where the agent offers options in order of estimated utility. The overall online learning algorithm is summarized in Algorithm 4.

6 Experiments and Results

We have conducted a large number of experiments in SC-EIMAPP (with $\mathcal{R} = \emptyset$) to determine the efficacy of our approaches and to understand the effects of different utility functions and preference profiles. We say the preference

Algorithm 3 SC-Ordered-Learner: Select-Option(v)

```
First-Call for $v$: sort-by-est-util(unoffered$_v$)
$o_w =$ pop(unoffered), offer($o_w$)
Subsequent Calls:
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($o_w$)
```

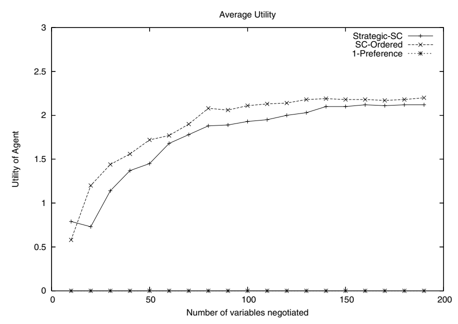
Algorithm 4 Online-Learning-Negotiator: receive-offer(o,a,v,r)

```
First-Call for $v$: explore = 1
All-Calls:
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)
```

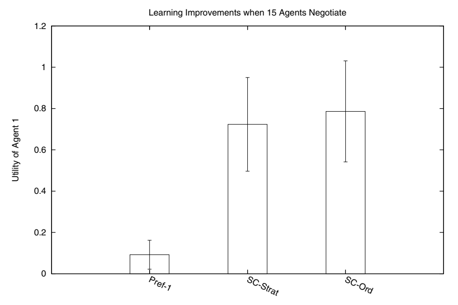
range is N to indicate that for each option each agents' preference is chosen from the set of integers 1 to N uniformly at random. When graphs displayed do not show error bars the standard deviation calculated over the trials was insignificant. In the results shown we used the same formula to set the value of ϵ. $\epsilon = (\gamma^{\frac{\text{no. options}}{2}+1})$*preference range*. For a 2 agent variable, the number of rounds before agreement will not exceed the exponent. We chose this ϵ so the effects of different γ values would be clear.

6.1 Adaptation in Self-Play

Figure 1(a) demonstrates that both our exploitation approaches effectively adapt over time in self-play. In the experiment, $\gamma = 0.3$, the number of options is 40, and the preference range is 40. Using the 1-Preference order strategy resulted in the agent receiving approximately zero utility, The graph shows the utility 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 setting, since the SC-agents are not consistent in when they offer options. This is because SC-agents explore and seek to compromise when they exploit. However, after only 10 variables have been negotiated, the learning approaches are already performing significantly better than the non-learning approach. *Adapting quickly online is important in many agreement problems*, e.g., in a domain like meeting scheduling, there may be few opportunities to learn about an agent. For other experimental settings we got similar results.



(a) The Strategic-SC-Learner and the SC-Ordered-Learner increase the average utility they achieve over time in self-play



(b) 15 agents negotiate variables with up to 10 decision makers.

6.2 Learning With More than Two Agents

Our learning approach also significantly out-performs not learning when there are more than two agents. Figure 1(b) shows results from an experiment with 15 Agents. There were 20 options and the preference range was 40. A total of 300 variables were assigned each trial (200 trials). Agent 1 was a decision maker for every variable, and each variable had up to 10 participants. The number of decision makers for a variable was chosen uniformly at random from $[2 - 10]$. Decision makers, and the initiator for a variable were also randomly selected. The figure shows results for when all agents had a γ value of 0.5. Negotiating variables with many decision makers makes the exploitation process more challenging since an agent must rely on estimates about multiple agents (and if it is not the initiator of the current variable, the agent doesn't see the offers of most these agents for the current variable). Despite this challenge, Figure 1(b) clearly shows that both our exploitation approaches greatly out perform not learning online.

6.3 Effects on an agent with low γ

Table 1, shows the effect agents' with different γ values have on a very cooperative agent. Agent 1's γ value is 0.3. The table shows the utility (averaged over 200 trials) Agent 1 achieves when it negotiates with three different "Other" agents. The first agent's γ is 0.3, the second's 0.7, and the third's 0.9. We know from extensive experiments, that *regardless of the other agent's γ, an SC-agent always benefits from learning*. However, we see that a low γ agent, does receive less utility (reduction is less than 50%) when negotiating with agents with increasing γs. This is intuitive, since higher γ values mean an agent is slower to compromise its preferences and agree. However, in many SC-domains, users' may be encouraged by the presence of agents that agree quickly to emulate this behavior for social reasons. This points to an interesting area for further study.

γ		Av. Utility		Std. Dev.	
A1	Other	A1	Other	A1	Other
0.3	0.3	0.98116	1.01526	0.50872	0.51495
0.3	0.7	0.78669	10.5464	0.5013	2.3873
0.3	0.9	0.63201	23.4496	0.5295	3.1823

Table 1. Low fixed γ vs varying γ (SC-Ord)

6.4 Discussion

The experiments presented are for an SC-EIMAPP with $\mathcal{R} = \emptyset$. Such SC-EIMAPPs can represent problems like a sports team repeatedly picking team plays. In domains with restrictions on option use, e.g., meeting scheduling, where a time-slot can't be used twice on the same day, learning can be more challenging. However, we have found predicting the round agents will offer options can also be used to significantly improve performance in this domain [1]. In the future we would like to evaluate against a wider variety of exploitation approaches, and further explore SC-EIMAPP domains with restrictions on option use.

7 Conclusions

In this paper we formalized agreement problems by defining EIMAPP and SC-EIMAPP, emphasizing the importance of preferences and the relevance of privately owned information. We introduced a notion of semi-cooperative agents, where agents become more cooperative as negotiation time increases. Constructing approaches for effectively representing a human's tendency to both look out for their own interests and to compromise, is important for building personal assistant agents. We analyzed EIMAPPs and SC-EIMAPPs theoretically, leading us to design an approach for semi-cooperative agents to *learn to improve their negotiation*. 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.

[1]Paper in preparation.

Acknowledgements

The work presented is partially supported by the Defense Advanced Projects Research Agency (DARPA) under Contract No. FA8750-07-D-0185. The views and conclusions contained in this document are those of the authors only.

References

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

[2] R. M. Coehoorn and N. R. Jennings. Learning on opponent's preferences to make effective multi-issue negotiation trade-offs. In *ICEC '04: Proceedings of the 6th international conference on Electronic commerce*, pages 59–68, New York, NY, USA, 2004. ACM Press.

[3] E. Crawford. Improved negotiation in multiagent agreement problems through learning. In *PhD Thesis, In preparation*, 2008.

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

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

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

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

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

[9] P. J. Modi and M. Veloso. Bumping strategies for the private incremental multiagent agreement problem. In *AAAI Spring Symposium on Persistant Agents*, 2005.

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

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

[12] D. Zeng and K. Sycara. Bayesian learning in negotiation. *Int. J. Hum.-Comput. Stud.*, 48(1):125–141, 1998.

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