

# Facing the Challenge of Human-Agent Negotiations via Effective General Opponent Modeling\*

Yinon Oshrat<sup>1</sup>, Raz Lin<sup>1</sup> and Sarit Kraus<sup>1,2</sup>

<sup>1</sup> Department of Computer Science  
Bar-Ilan University  
Ramat-Gan, Israel 52900  
{linraz,sarit}@cs.biu.ac.il

<sup>2</sup> Institute for Advanced Computer Studies  
University of Maryland  
College Park, MD 20742 USA

## ABSTRACT

Automated negotiation agents capable of negotiating efficiently with people must deal with the fact that people are diverse in their behavior and each individual might negotiate in a different manner. Thus, automated agents must rely on a good opponent modeling component to model their counterpart and adapt their behavior to their partner. In this paper we present the *KBAgent*. The *KBAgent* is an automated negotiator that negotiates with each person only once, and uses past negotiation sessions of others as a knowledge base for general opponent modeling. The database is used to extract the likelihood of acceptance and proposals that may be offered by the opposite side. Experiments conducted with people show that the *KBAgent* negotiates efficiently with people and even achieves better utility values than another automated negotiator, shown to be efficient in negotiations with people. Moreover, the *KBAgent* achieves significantly better agreements, in terms of individual utility, than the human counterparts playing the same role.

## Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial IntelligenceIntelligent agents

## General Terms

Experimentation

## Keywords

automated bilateral negotiation, opponent modeling

## 1. INTRODUCTION

Multi-issue multi-attribute negotiations surrounds our every day life. Most of the time negotiations involve only two parties that share an interest or conflicting preferences. This field has long been

\*This research is based upon work supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant number W911NF-08-1-0144 and under NSF grant 0705587.

**Cite as:** Facing the Challenge of Human-Agent Negotiations via Effective General Opponent Modeling, Yinon Oshrat, Raz Lin and Sarit Kraus, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.  
Copyright © 2009, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

the focus of researchers in the agents community and many automated agents designed for bilateral negotiations can be found in the literature (e.g., [3, 4, 8, 13, 22]). Yet, only a few of these agents are specifically designed to efficiently negotiate with people and are evaluated in actual negotiation settings. One of the main difficulties when negotiating with people is incomplete information about the other party's preferences and its dynamic behavior. Human behavior is diverse and cannot be captured by a monolithic model. Humans tend to make mistakes, and they are affected by cognitive, social and cultural factors, etc. [1, 10].

While the strategy of an automated agent might be the most efficient one, the effect of identifying and efficiently modeling the other party has a tremendous effect on the agent's success. For example, negotiation over the same issues against parties from different countries, can result in distinct agreements, and it is vital that the negotiator will be aware of these variations. It has been shown that in different countries the attitude regarding negotiations and the actions performed during them are quite different. A Chinese negotiator will appear to concede more often while in the UK it is common to use pressure tactics to impose a deal on the other party. The same tactic, however, against a negotiator from Greece will most likely backfire [5, 11].

In this paper we present the *KBAgent* who negotiates with each person only once in multi-issue multi-attribute environments. The *KBAgent* is an automated agent that was designed to improve the performance of the *QOAgent* [13], an automated agent shown to be an efficient automated negotiator with human negotiators in such environments. The *KBAgent* incorporates a better learning mechanism which enables it to perform better than the *QOAgent*. In essence, the *KBAgent* uses a database of past negotiation sessions between specific agents (types of negotiators) to allow it to be more efficient in negotiations with agents of that specific type. Based on a database with past negotiation sessions, the agent performs offline learning, which is based on the kernel based density estimation ([23], Chapter 2). From the database the agent estimates the probability of an offer to be accepted, the probability of it to be offered and the expected average utility for the other party. These probabilities are then used in its decision making component, either when accepting an offer or to determine the agent's concession rate. Using the kernel based density estimation the *KBAgent* is capable of using even small databases and does not have to rely on many past negotiation sessions.

We consider a setting of a finite horizon bilateral negotiation with incomplete information between an automated agent and a human counterpart. The incomplete information is expressed as

uncertainty regarding the utility preferences of the opponent, and we assume that there is a finite set of different agent types. The negotiation itself consists of a finite set of multi-attribute issues and time constraints. In the experiments the *KBAgent* was matched with people in a negotiation on a single domain. The agent performed better, as measured in the average utility score, than the *QOAgent*. When playing one of the parties, these results were also statistically significant. This was the side in which the *QOAgent*'s performance was not significantly better than those of people. Encouraged by these results, we also compared the *KBAgent*'s performance to people's performance and we show that it performed significantly better than people in both roles. Consequently we achieved our goal of developing an agent that can negotiate significantly better than people in any role it plays. In addition, we demonstrate that our proposed solution also conforms to some properties of the *Nash bargaining solution*. This gives us the theoretical basis required for use of our technique in bilateral negotiation, and for the assumption that offers proposed by the *KBAgent* will also be considered to be accepted by the opponent.

This work contributes to research on the design of automated negotiators by suggesting an innovative method for designing an agent's strategy based on past history of other agents. Previous work in this field (e.g., [4, 8]) attempt to learn the opponent's utility directly and assume that utility model of the opponent exists. However, it has been shown that people do not follow equilibrium strategies [6, 15] nor do they maximize their expected utility or behave rationally. Our work stems from our observation that the agent's strategy is mainly influenced by the probability of an agreement to be accepted or offered by the other party. The main focus of our work deals with the learning of these probabilities rather than modeling the utility of the other party. Moreover, the learning method described in our work does not rely on the structure of the utility function of the other party, but rather constructs a preference relation between the possible offers. As a result, our agent is capable of learning from previous existing data, no matter which type of agents participate in these negotiations. Thus, the *KBAgent* has a generic learning mechanism, allowing it to be matched and negotiate efficiently with many possible types of negotiators, given that there are previous sessions (even a relatively small number of sessions) with similar negotiators. Since most negotiations are one-shot and not repeated we focus on *general opponent modeling* and try to avoid modeling the exact person with whom the *KBAgent* negotiates.

The rest of the paper is organized as follows. Section 2 provides an overview of bilateral negotiation with incomplete information and opponent modeling. Section 3 provides an overview of the negotiation context, followed by Section 4 which presents the design of the *KBAgent*, including the opponent modeling and decision making mechanisms. Section 5 describes the experimental setting and methodology and reviews the results. Finally, Section 6 provides a summary and discusses future work.

## 2. RELATED WORK

Many of the approaches for modeling agents in the context of multi-issue bilateral negotiations focus on modeling the agent utility function [4, 8], with the underlying assumption that the utility function of the other negotiator has the strongest effect on the agent's strategy. While this assumption might be true for rational agents, it is quite problematic when automated negotiations should be matched with people. Results from social sciences suggest that people do not follow equilibrium strategies [6, 15] nor do they maximize their expected utility or behave rationally.

Some researchers suggest the use of special protocols to help in

the modeling process of the other side. Pasquier *et al.* [18] propose using an argumentation based protocol, which enables each side to learn the interest of the other side and thus obtaining more information regarding its preferences. Saha and Sen [20] devise a protocol that promises envy-free Pareto Optimal solution when dealing with rational agents. However, in this paper we confine ourselves to the simple protocol of alternating offers ([17], p. 118-121) and try to take a different approach to achieve general opponent modeling.

Saha *et al.* [19] propose a specific opponent modeling technique. They use Chebychev's polynomials to approximate the probability function of the other side. Their method requires exploration stage where information from random offers is used to approximate underlying probability function. However, this also raises questions whether one would want to propose random offers which might be suboptimal only to obtain a better estimation of the counterpart. Moreover the amount of data needed for this method to be practical is quite larger than the database size our agent works on and achieves using it a good general opponent modeling.

Hindriks and Tykhonov [8] propose a Bayesian learning scheme to learn the opponent model, which is based on learning its utility function. However, their method is parametric and works well only when the structure of the utility function is known. Moreover, they did not evaluate the efficacy of their approach against people, but rather using automated agents. We, on the other hand, try to benefit from the knowledge we have regarding previous negotiation sessions of people of the same population serving as the other negotiator, and we also demonstrate the efficacy of our agent against people.

Coehoorn and Jennings [4] suggest the use of non parametric kernel density estimation to learn the opponent model. While we use the same method suggested by Coehoorn and Jennings, they assume that their opponent strategy is known. This assumption is not true when negotiating with people since their behavior is diverse. Furthermore, if we want to learn from past negotiations conducted with people, the sparsity of the samples is a major issue and the three dimensional learning scheme Coehoorn and Jennings describe in their paper may provide insufficient data, making their approach impractical. Lastly, Coehoorn and Jennings dealt only with automated agents and did not show the efficacy of their approach when negotiating with people.

Gal *et al.* [7] also deal with the problem of learning agents' models from past interactions between humans. However, they are focused on one shot games, while we focus on bilateral multi-issue multi-attribute negotiation domains. Their learning method is based on expectation maximization (EM), which is a parametric learning method and thus assumes that a model of how the opponent behaves exists. This, however, is not the case in our situation.

Katz *et al.* [9] also suggested the use of general opponent modeling. They were focused on single issue auctions and utilized a reinforcement learning algorithm, which integrates virtual learning with reinforcement learning. However, their agent is tested in a single-issue domain with repeated interactions that are used to improve the learning and decision making mechanism. It is not clear whether their approach would be applicable to negotiation domains in which several rounds are made with the same opponent and multi-issues offers are negotiated.

Lin *et al.* [13] also designed an automated agent and showed its efficacy in negotiating well with people. However, their agent has several drawbacks that we try to address in this paper. First, they state that most of the agreements were reached by the agent accepting the other party's offers. Thus, their offer generation mechanism seems to be an inefficient one. Moreover, their acceptance of offers depended on a random threshold. Also, their agent's strategy is

not flexible and in many of the negotiation sessions it repeatedly proposes the same offer to the other party. Our agent, on the other hand, incorporates a useful mechanism for generating offers and the experiments indeed have shown that more agreements are reached by the other party accepting the automated agent’s offers.

### 3. PROBLEM DESCRIPTION

We consider the problem of efficient general opponent modeling as a key to improving the performance of an automated negotiator. We deal with a bilateral negotiation in which two agents negotiate *once* to reach an agreement on conflicting issues. The negotiation can also end if one of the agent opts out or if the deadline, denoted  $dl$ , is reached, whereby, if no agreement is reached, a status quo outcome, denoted  $SQ$ , is implemented. Let  $I$  denote the set of issues in the negotiation,  $O_i$  the finite set of values for each  $i \in I$  and  $O$  a finite set of values for all issues ( $O_1 \times O_2 \times \dots \times O_{|I|}$ ). We allow partial agreements,  $\perp \in O_i$  for each  $i \in I$ . Therefore an offer is denoted as a vector  $\vec{o} \in O$ .

The negotiation session is divided into time periods,  $\mathbf{Time} = \{0, 1, \dots, dl\}$ . Each agent is assigned a time cost which influences its utility as time passes. In each period  $t \in \mathbf{Time}$  of the negotiation, if the negotiation has not terminated earlier, each agent can propose a possible agreement, and the other agent can either accept the offer, reject it or opt out. Each agent can either propose an agreement which consists of all the issues in the negotiation, or a partial agreement. We use an extension of the model of alternating offers ([17], p. 118-121), in which each agent can perform up to  $M > 0$  interactions with its counterpart in each time period.

The negotiation problem also involves incomplete information about the preferences of the opponent. We assume that there is a finite set of agent types. These types are associated with different additive utility functions (e.g., one type might have a long term orientation regarding the final agreement, while the other type might have a more constrained orientation). Formally, we denote the possible types of agents as  $\mathbf{Types} = \{1, \dots, k\}$ . Given  $l \in \mathbf{Types}$ ,  $1 \leq l \leq k$ , we refer to the utility of an agent of type  $l$  as  $u_l$ , and  $u_l : \{(O \cup \{SQ\}) \cup \{OPT\}\} \times \mathbf{Time} \rightarrow \mathbb{R}$ . Each agent is given its exact utility function. The negotiators are aware of the set of possible types of the opponent. However, the exact utility function of the rival is private information.

### 4. AGENT DESIGN

The design of the *KBAgent* is built on top of the design of another automated agent, the *QOAgent* [13], which has been shown to be an efficient automated negotiator, especially with respect to negotiating with humans. For this reason, we also compare the *KBAgent*’s results to those of the *QOAgent* and find whether indeed our agent can negotiate even better than the *QOAgent*. In addition, we also use the simulation environment provided by the *QOAgent*, which is rich and supports bilateral multi-issue and multi-attribute negotiations, both with human counterparts and automated agents.

The main difference between the *KBAgent* and other automated agents is its inherent design, which builds a general opponent model. *KBAgent* utilizes past negotiation sessions of other agents as a knowledge base for the extraction of the likelihood of acceptance and offers which will be proposed by the other party. These data are used to determine which offers to propose and which offers to accept. One of the main advantages of the *KBAgent* is that it can also work well with small databases of negotiation sessions. In the following subsections we elaborate on the offline general opponent modeling method and the online decision making method during the negotiations.

### 4.1 General Opponent Modeling Component

This component is responsible for generating the general opponent modeling. We also show below that the overall complexity of the proposed algorithm is  $O(|\mathbf{Types}| \cdot |\mathbf{Time}| \cdot n \log(n))$ , where  $n$  is the total number of possible offers in the domain. Note that the general opponent modeling is done offline and not during the negotiation process itself.

The *KBAgent* performs what is called a *Kernel based Density Estimation (KDE)* of the negotiation sessions in the database ([23], Chapter 2). *KDE* is a nonparametric technique for estimating the probability functions based on samples, and can also be viewed as a smoothing technique based on samples. The use of kernel based density estimation was chosen for several reasons. First, it is an a-parametric approach that does not assume a model of the other side, just like the data we expect to have in our database. Second, it has a low computation complexity. The complexity of performing *KDE* on a sample of size  $m$  is  $O(m \log(m))$ . Third, it can perform well even with small data sets.

The *KBAgent* uses the *KDE* method to estimate the probability of each offer to be proposed by the other party during a given turn. In addition, the database is used to calculate the probability of each offer to be accepted by the other side during any turn. We denote by  $Q(\vec{o})$  the estimated probability of each offer  $\vec{o} \in O$  to be accepted at any given time by the opposite party and by  $P(\vec{o}, t)$  the probability that the opposite party would propose offer  $\vec{o}$  during a given turn  $t$ . These probabilities will be used later in the decision making process, as described in Section 4.2.

The database of past negotiations includes logs of past negotiation sessions between two sides in the specified domain. The negotiation sessions include all offers made by the two parties in each turn and whether or not the offers were accepted. The negotiation sessions may be collected from any population, not necessarily similar to the one which the *KBAgent* negotiates with.

The estimation of the probabilities is done separately for each possible agent type. If the negotiation sessions in the database are not labeled with the type of the agent, the type is elicited using a simple Bayes’ classifier (similar to the one used for estimating the believed type of the other party during the negotiations, as discussed in Section 4.3). To estimate the probability of offers being proposed by the agent of a given type we extract from the database all offers proposed by the agents of that type in any given turn  $t$ ,  $1 \leq t \leq dl$ . To apply the *KDE* algorithm all offers have to be assigned a unique numerical value, thus we order the offers by their utility values of the other side and rank them, such that the offer with the highest possible utility value is ranked 1. Then, the *KDE* is applied and the results also include a smooth probability value even for agreements that were not part of the samples in the database. Since the *KDE* algorithm is used each turn, the overall complexity is  $O(|\mathbf{Time}|n \log(n))$ .

To estimate the probability the other side will accept an offer during the negotiation we extract from the database all the offers ever accepted or proposed by at least one negotiator playing the role of the other party (under the assumption that if anyone had proposed an offer they would also accept it when proposed the same offer). We will refer to them as “acceptable offers”. For each offer during the negotiation that the *KBAgent* proposes, the acceptance probability is calculated by computing the ratio between (a) the number of the offers from the acceptable offer list that have a utility value for the other party lower than the utility value of the proposed offer, and (b) the total number of offers in the acceptable list of the other party. The complexity of this process is basically dominated by the sorting of the offers, and thus it is  $O(n \log(n))$ . We use the following example to demonstrate this. Assume that the database contains

sessions in which people proposed offers with the following utilities: {400, 380, 300, and 200}. Also assume that they accepted an offer with a utility value of 280. Thus, the acceptable offers list will contain the offers with the following utility values: {400, 380, 300, 200, and 280}. Now assume that we would like to calculate the probability that the other party will accept an offer with a utility value of 290. Since there are two agreements in the acceptable list with a utility value lower than 290 (280 and 200) the estimated probability of acceptance is  $2/5 = 0.4$ .

Finally, from the database we calculate the average expected utility of the other party. This is done by averaging the final utility scores of all agreements reached in the negotiation sessions (no matter which turn the agreements were reached). We denote this average as *ExpectedOppAvg*. The complexity for calculating this is  $O(n)$ .

## 4.2 The Decision Making Component

The decision making valuation component takes into account the agent's utility function, as well as the believed type of opponent (note that the believed type of opponent is also influenced by the offers proposed by the opponent, as described in Section 4.3). This data is used both for deciding whether to accept or reject an offer and for generating an offer. In our settings, although several offers can be proposed each time period, we restrict the agent to a single offer in each period. This is done due to the fact that our mechanism for generating offers only produces one distinct offer at a given time period. The opponent, on the other hand, is free to propose several offers, and the agent can respond to all the offers, which actually occurred in the experiments. We demonstrate the efficacy of *KBAgent*'s decision making process using experiments with people in an environment of incomplete information, as described in Section 5.

### 4.2.1 Generating Offers

One of the main drawbacks that Lin *et al.* [13] state concerning their *QOAgent* is that most of the agreements reached in negotiations in which their agent was involved were offered by the human counterpart. In addition, most of the time, their agent repeatedly proposed the same offer and did not generate new offers as time progressed. We build on the same generating offer mechanism that was implemented in the *QOAgent* but invoke a different strategy for the generation of offers. As opposed to the *QOAgent*, the *KBAgent* implements a concession oriented strategy when generating offers. We also show in Section 4.2.3 that the offers proposed by the *KBAgent* conform to some properties from classical negotiation theory, which are mainly used by mediators.

The process of deciding what offers to propose at each turn consists of three phases. First, a list of offers ranked by their *QOValue* is generated. The *QOValue* of an offer is an alternative to the Nash bargaining solution. In short, it tries, in a qualitative manner, to evaluate the offers based on the agent's utility and based on the likelihood of their acceptance by the other party (c.f. [13] for more details). The *QOValue* of an offer is calculated using the following Formula:

$$\begin{aligned} QOValue(\vec{o}) &= \min\{\alpha_o, \beta_o\} \\ \text{where } \alpha_o &= rank_a(\vec{o}) \cdot lu_a(\vec{o}) \\ \text{and } \beta_o &= [lu_a(\vec{o}) + lu_b(\vec{o})] \cdot rank_b(\vec{o}) \end{aligned} \quad (1)$$

where  $rank(\cdot)$  is the ranking value of an offer, which is associated with each offer and a given utility function  $u$ . The rank number of

an offer  $\vec{o} \in O$  is calculated using the following formula:

$$rank(\vec{o}) = \frac{order(\vec{o}, O)}{|O|} \quad (2)$$

where  $order(\cdot, \cdot)$  is the ordering function which orders the offer  $\vec{o}$  in an ordinal scale between 1 and  $|O|$  according to its utility value compared to all other offers in  $O$ .  $lu(\cdot)$  denotes the Luce number of an offer [14], which is a non-negative number that is associated with each offer. The Luce number of an offer  $\vec{o} \in O$  is calculated using the following formula:

$$lu(\vec{o}) = \frac{u(\vec{o})}{\sum_{\vec{x} \in O} u(\vec{x})} \quad (3)$$

After calculating the *QOValue* of each offer, the *KBAgent* orders all offers by their *QOValue*. The offer with the maximal *QOValue* is the first offer the *KBAgent* proposes. The second phase is to construct a new list which will be the one from which offers will be proposed. This new list is based on the sorted offers by their *QOValue*, in which the utility values for the *KBAgent* are above the value of the status quo. In addition, the *KBAgent* discards all offers with a lower *QOValue* that do not improve the other party's utility from offers with a higher *QOValue*. That is, if  $QOValue(\vec{o}_i) < QOValue(\vec{o}_j)$  and  $u_{opp}(\vec{o}_i) < u_{opp}(\vec{o}_j)$  the offer  $\vec{o}_i$  will be removed from the proposal list. The last phase is deciding which offer to propose each turn. As mentioned earlier, the first offer that is proposed is the one with the maximal *QOValue*. The other offers are picked from the ordered list based on the concession rate the *KBAgent* applies and are chosen with a decreasing *QOValue* for the agent and an increasing utility value for the other party. Note that while we assume that the other party is not rational, we refer to the case that it is not maximizing its expected utility, thus it should still prefer accepting offers with higher utility values than lower ones. The concession rate is determined such that after passing 80% of the negotiation turns, the *KBAgent* will propose the offer which is as closest to the *ExpectedOppAvg*, that is, the average expected utility value of the other party, as estimated from the database of past negotiations. In this way the *KBAgent* makes concessions while offering agreements which are still efficient for it. The concession rate basically determines the order in which offers in the offer list will be proposed. Thus, even if after 80% of the turns the other side rejects the offer, the *KBAgent* continues to propose offers from its list based on the concession rate.

Listing 1 describes the pseudo-code of the algorithm for generating the proposed offers and the concession rate. It is easy to observe that the complexity of the algorithm for generating the offers list is dominated by the complexity of sorting, and thus the overall complexity is  $O(n \log(n))$ . We will demonstrate *KBAgent*'s offer generation mechanism using the following example. Assume that in a given negotiation session there are 5 turns and 10 possible offers. Table 1 lists the possible offers sorted by their *QOValue*. Based on the utility value of each offer for the other party it also lists the final ordered list of offers from which the *KBAgent* will propose offers to its counterpart. Now, assume that the average expected utility value of the opposite side, as estimated from the database, is 440. Based on *KBAgent*'s algorithm, after 80% of the turns, that is, after turn 4 ( $5 \cdot 0.8 = 4$ ), it should propose offer #7, which is the first one with a higher utility value for the other party than the average expected utility determined from the database. This offer is the fifth offer in the ordered list of offers and thus the agent can now establish its concession rates and the offers to be proposed each turn. Since the *KBAgent* aims to propose the fifth offer after turn 4 the concession rate is determined as  $5/4 = 1.25$ . Thus, the agent should first propose the first offer in the list and then follow the concession rate.

**Listing 1** Generating Proposed Offers

```

1: for all  $\vec{o} \in Database$  do
2:    $qoValue = QOValue(\vec{o})$ 
3:   Insert  $\vec{o}$  to QOSortedList
4: end for
5: QuickSort(QOSortedList) based on  $qoValue$ 
6: Insert QOSortedList(0) to OfferList
7:  $j = 0, maxUtil = u_{opp}(QOSortedList(0)), concessionIndex = 0$ 
8: for  $i = 1$  to QOSortedList.length-1 do
9:    $util = u_{opp}(\vec{o}_i)$ 
10:  if ( $util > maxUtil$ ) and ( $u_{KB}(\vec{o}) > u_{KB}(SQ)$ ) then
11:    Insert QOSortedList( $i + 1$ ) to OfferList
12:     $maxUtil = util$ 
13:     $j = j + 1$ 
14:    if ( $concessionIndex = 0$ ) and ( $util > ExpectedOppAvg$ ) then
15:       $concessionIndex = j$ 
16:    end if
17:  end if
18: end for
19:  $concessionRate = concessionIndex / (0.8 \cdot |Time|)$ 

```

Offer Idx	$QOValue(\vec{o})$	$u_{opp}(\vec{o})$	Ordered List of offers
0	13.45	<b>350</b>	<b>0</b>
1	12.5	300	-
2	12	<b>400</b>	<b>1</b>
3	11.22	<b>430</b>	<b>2</b>
4	10.3	350	-
5	10	<b>435</b>	<b>3</b>
6	9.87	470	4
7	9.8	<b>490</b>	<b>5</b>
8	9	410	-
9	8.8	500	6

**Table 1: Example of deciding which offers will be proposed. Proposed offers are marked in bold.**

That is, it should propose offers from its list in the the following order: 0, 1.25, 2.5, 3.75, 5, 6.25. The numbers are rounded down, so the *KBAgent* eventually proposes the offers ordered 0, 1, 2, 3, 5 in its offer list, by that order. These offers are also marked in bold in Table 1.

#### 4.2.2 Accepting Offers

An important aspect of the agent's strategy is the decision of whether to accept or reject an offer. The *KBAgent* determines a time dependent threshold to decide whether to accept or reject an offer. Obviously, a good threshold should be used and not an arbitrary one. In order to decide on the optimal threshold, the probabilities learned from the database of past negotiations, are used.

Let  $o_{KB}(t)$  be the offer proposed by the *KBAgent* at time  $t$ . The acceptance threshold for the *KBAgent* is calculated per every turn  $t$  of the negotiation and denoted by  $\alpha_t$ . We aim that  $\alpha_t$  will allow to maximize the expected utility value of the agent for turn  $t$ . Thus, we first calculate the expected utility value of the *KBAgent* for each turn of the negotiation, denoted  $E(t, \alpha_t)$ . Since the negotiation terminated at turn  $dl$  we can use backward induction to calculate

this. The expected utility of the agent if the deadline is reached and no agreement has been made equals the utility of the status quo. Thus:

$$E(dl, \alpha_{dl}) = u_{kb}(SQ) \quad (4)$$

In the preceding turn, if *KBAgent's* offer was not accepted, it should accept any agreement with a utility value higher than  $SQ$ . Otherwise, the negotiation will terminate with a status quo outcome and a lower utility value for the agent. Thus, its expected utility depends on the probability that offers with a utility value higher than the status quo would be proposed by the other party. For all offers which are above the acceptance threshold of the *KBAgent* we sum their probability of being proposed by the counterpart multiplied by their utility value. For all other agreements, as the agent will reject them, we sum the probability of the counterpart proposing them multiplied by the value of the status quo. Formally,

$$\begin{aligned}
E(dl - 1, \alpha_{dl-1}) = & \sum_{u_{KB}(\vec{o}, dl-1) \geq \alpha_{dl-1}} P(\vec{o}, dl - 1) u_{KB}(\vec{o}, dl - 1) + \\
& \sum_{u_{KB}(\vec{o}, dl-1) < \alpha_{dl-1}} P(\vec{o}, dl - 1) u_{KB}(SQ, dl - 1) \quad (5)
\end{aligned}$$

For every other turn, we calculate the expected utility of the *KBAgent* using a recursion. Basically, the expected utility of the agent at turn  $i$  depends whether or not it accepts the proposed offer:

- For all offers above the agent's acceptance threshold, the *KBAgent* sums the multiplication of their utility value with their probability of being proposed by its counterpart.
- For all offers below the agent's acceptance rate, the *KBAgent* calculates the probability that its offer will be accepted by its counterpart in the next turn or if it is rejected, the agent's expected utility for the subsequent turn.

Formally,

$$\begin{aligned}
E(t, \alpha_t) = & \sum_{u_{KB}(\vec{o}, t) \geq \alpha_t} P(\vec{o}, t) u_{KB}(\vec{o}, t) + \\
& \left( \sum_{u_{KB}(\vec{o}, t) < \alpha_t} P(\vec{o}, t) \{ Q(o_{KB}(t+1)) \cdot u_{KB}(o_{KB}(t+1), t+1) + \right. \\
& \left. [1 - Q(o_{KB}(t+1))] E(t+1, \alpha_{t+1}) \} \right) \quad (6)
\end{aligned}$$

The last calculation needed is the extraction of  $\alpha_t$  to determine the acceptance threshold for each turn. To this end, we use the derivative of Formula 6 by  $\alpha_t$ <sup>1</sup>, and as we would like to maximize the expected utility we find the  $\alpha_t$  value when the derivative equals 0. Thus we need to find the solution for the following equation:

$$\begin{aligned}
\alpha_t = & Q(o_{KB}(t+1)) u_{KB}(o_{KB}(t+1), t+1) + \\
& (1 - Q(o_{KB}(t+1))) E(t+1, \alpha_{t+1}) \quad (7)
\end{aligned}$$

Based on Formulas 5, 6 and 7 we can backtrack and calculate the optimal acceptance threshold for each turn.

Note that these calculations are done separately for each different believed agent type of the counterpart.

<sup>1</sup>In order to perform this derivation we replace  $\alpha_t$  with  $u_{KB}(\beta_t, t)$  where  $\beta_t$  is the threshold agreement, that is all offers with utility values higher than the utility if it were accepted. We then take derivative by  $\beta_t$ .

### 4.2.3 An Alternative to the Nash Bargaining Solution

The Nash bargaining solution is defined by several characteristics and is usually designed for a mediator in an environment with complete information ([17], Chapter 15). The solution for the bargaining problem is said to be a Nash solution if it satisfies symmetry, efficiency, invariance and independence of irrelevant alternatives. Basically, *symmetry* states that if both players have the same bargaining power, then neither player will have any reason to accept an agreement which yields a lower payoff for it than for its opponent. For example, for the solution to be symmetric, it should not depend on the agent who started the negotiation process. *Efficiency* states that two rational agents will not agree on an agreement if its utility is lower for both of them than another possible agreement. This solution is said to be Pareto-optimal. *Invariance* states that for all equivalent problems the solution is also the same. That is, two positive affine transformations can be applied on the utility functions of both agents and the solution will remain the same. Finally, *independence of irrelevant alternatives* asserts that if new agreements are added to the problem in such a manner that the status quo remains unchanged, either the original solution is unchanged or it becomes one of the new agreements.

It was shown by Nash [16] that the only solution that satisfies all of these properties is the product maximizing of the agents' utilities. However, as we stated, the Nash solution is usually designed for a mediator. Since the *KBAgent* negotiates with people and proposes several offers, it cannot satisfy all of these properties. However, we can prove that the *KBAgent's* strategy for proposing offers conforms to most of the properties and a revised independence of irrelevant alternatives property that allows for a set of possible solutions instead of one unique solution.

**THEOREM 4.1.** *The KBAgent's proposing offers mechanism satisfies the properties of efficiency, invariance and independence of irrelevant alternatives solutions.*

Due to space limitations, we do not present the proof of the theorem in the paper<sup>2</sup>. It is also interesting to note that the *KBAgent's* strategy does not guarantee that the offers it proposes will be better for it than the Nash solution. There are cases in which, due to its concession strategy, offers, with a lower utility for it than the Nash solution, will be proposed. Nevertheless, our experiments showed that the *KBAgent's* strategy is efficient and it negotiates efficiently with people and even achieves better utility values than another automated negotiator which offers always yields higher utility values than the Nash bargaining solution.

### 4.3 Identifying The Opponent's Type

The *KBAgent* uses the same reasoning mechanism as suggested by [13] for identifying an opponent's type. This mechanism is based on the Bayesian updating rule that updates or revises beliefs in light of new evidence a posteriori ([12], Chapter 2). As incomplete information in the environment is modeled by the existence of several agent types, in each time period, the agent consults the component in order to update its belief regarding the other party's type.

Recall that there are  $k$  possible types of agents. At time  $t = 0$  the prior probability of each type is equal, that is,  $P(\text{type}_{t=0}^i) = \frac{1}{k}, \forall i \in \mathbf{Types}$ . Then, for each time period  $t$  we calculate the a posteriori probability for each of the possible types, taking into account the history of the current negotiation. This is done *incrementally* after each offer is received or accepted. That is, the believed

<sup>2</sup>The proofs are accessible at <http://www.cs.biu.ac.il/~linraz/opponentModeling/proofs.pdf>

type is updated every time an offer is received or accepted, thus eventually it is based on the overall total history thus far. Thereafter, this value is assigned to  $P(\text{type}_t)$ . Using the calculated probabilities, the agent selects the type whose probability is the highest and proposes an offer as if it were the opponent's type. Formally, at each time period  $t \in \mathbf{Time}$  and for each type  $\in \mathbf{Types}$  and  $\vec{o}_t \in O$  (the offer at time period  $t$ ) we compute:

$$P(\text{type}^i | \vec{o}_t) = \frac{P(\vec{o}_t | \text{type}^i)P(\text{type}_t^i)}{P(\vec{o}_t)} \quad (8)$$

where  $P(\vec{o}_t) = \sum_{i=1}^k P(\vec{o}_t | \text{type}^i) \cdot P(\text{type}_t^i)$ .

Now we can deduce the believed type of the other party for each time period  $t$ ,  $BT(t)$ , using the following equation:

$$BT(t) = \arg \max_{i \in \mathbf{Types}} P(\text{type}^i | \vec{o}_t), \quad \forall t \in \mathbf{Time} \quad (9)$$

## 5. EXPERIMENTS

We used a simulation environment, which is adaptable such that any scenario and utility functions, expressed as multi-issue attributes, can be used. We matched the *KBAgent* with people in negotiations on a given domain. The *KBAgent* played the two different roles in the negotiations, while the human counterpart accessed the negotiation interface via a web address. The negotiation itself was conducted using a semi-formal language. Each agent constructed an offer by choosing the different values constituting the offers. Then, the offer was constructed and sent in plain English to its counterpart. To test the efficiency of our proposed agent, we conducted experiments on a specific negotiation domain, in which the *QOAgent* [13] was also run. We then compared the *KBAgent's* performance with that of the *QOAgent* and with that of people. In the following subsections we describe our domain and the experimental methodology and we review the results.

We begin by describing the domain which was used in the experiment and then continue to describe the experimental methodology and results.

### 5.1 The Negotiation Domain

For the negotiation domain we choose a Job Candidate domain, which is related to the subjects' experience, and thus they could better identify with it. In this domain, which was first described in [13], a negotiation takes place after a successful job interview between an employer and a job candidate. In the negotiation both the employer and the job candidate wish to formalize the hiring terms and conditions of the applicant. The issues under negotiation are: (a) salary, (b) job description, (c) social benefits, (d) promotion possibilities, and (e) Working hours. In this scenario, a total of 1,296 possible agreements exist.

Each turn in the scenario equates to two minutes of the negotiation, and the negotiation is limited to 28 minutes. If the sides do not reach an agreement by the end of the allocated time, the job interview ends with the candidate being hired under a standard contract, which cannot be renegotiated during the first year. This outcome is modeled for both agents as the status quo outcome.

Each side can also opt-out of the negotiation if it feels that the prospects of reaching an agreement with the opponent are slim and it no longer possible to negotiate. Time also has an impact on the negotiations. As time advances the candidate's utility decreases, as the employer's good impression of the job candidate decreases. The employer's utility also decreases as the candidate becomes less motivated to work for the company.

The utility values range from 170 to 620 for the employer role and from 60 to 635 for the job candidate role. The status quo value

in the beginning of the negotiation was 240 for the employer and 160 for the job candidate. Both players had a fixed loss per time period – the employer of -6 points and the job candidate of -8 points per period.

As there is also incomplete information, we assume that there are three possible types of agents for each role. These types are associated with different additive utility functions and characterized as ones with short-term orientation regarding the final agreement, long-term orientation and a compromising orientation.

## 5.2 Experimental Methodology

We tested our agent against human subjects, all of whom are computer science undergraduate and graduate students. 28 human subjects were used to evaluate the performance of the *KBAgent*. The people negotiated against the *KBAgent* on the Job Candidate domain. We then compared the performance of the *KBAgent* in each role it played to similar simulations that were done using the *QOAgent* (which involved 34 people). The subjects did not know any details regarding the automated agent with which they were matched. The outcome of each negotiation was either reaching a full agreement, opting out, or reaching the deadline.

The *KBAgent* performed offline learning using the *KDE* technique on past negotiation sessions to estimate the different probabilities of accepting and proposing offers by the three different possible types of negotiators for each role (the Employer and the Job Candidate). The database consisted of 20 past negotiation sessions of 40 people negotiating one against the other as one specific agent type. To allow learning of the other types as well, we used automated agents that were designed by people to create additional negotiation sessions. 14 people had to design an automated negotiator agent to perform negotiations in the same domain and settings. These agents were matched against each other, each time given a different role and a different type of utility. 40 negotiation sessions were created for each of the other two types of agents. These sessions were added to the database to facilitate learning based on the *KDE* method.

Prior to the experiments, the subjects were given oral instructions regarding the experiment and the domain. The subjects were instructed to play based on their score functions and to achieve the best possible agreement for them.

## 5.3 Experiment Results

The main goal of the experiments was to verify that the *KBAgent* is an efficient negotiator capable of negotiating with human counterparts. In addition, we wanted to evaluate its performance in comparison to the *QOAgent*'s performance, which has been shown in the literature to be effective when negotiating with people [13]. Throughout this section, we also evaluate the significance of the results. The significance test was performed by applying the *t-test* and the *Wilcoxon* test on the results. The *t-test* is a statistical hypothesis test in which the test statistics has a *t-distribution* if the null hypothesis is true. This test requires a normal distribution of the measurements ([2], Chapter 3). Thus, it is used in our analysis to compare the utility values of the different simulation methods, which have continuous values. To analyze the significant differences in the end turn we use the *Wilcoxon signed-rank test*, which is a non-parametric alternative to the paired t-test for the case of two related samples or repeated measurements on a single sample. This test does not require any assumptions regarding the distribution of measurements ([21], Chapter 5).

Table 2 summarizes the results of the *KBAgent* when negotiating against people compared to the results of the *QOAgent* negotiating against people. First, we examine the final utility values of all the

	<i>KBAgent</i>	<i>QOAgent</i>	<i>p</i> -value
<b>Employer</b>			
Average utility	468.86	417.37	0.08
Stdev of utility	36.96	135.92	
% offers accepted by human	21.43%	7.14%	
Average sum of utilities	839.36	737.87	0.04
Average end turn	5.57	5.06	0.76
Stdev of end turn	3.06	2.1	
<b>Job Candidate</b>			
Average utility	482.71	397.83	0.001
Stdev of utility	57.51	85.96	
% offers accepted by human	50%	11.76%	
Average sum of utilities	863.14	829.61	0.17
Average end turn	7.14	5.37	0.08
Stdev of end turn	2.88	4.32	

**Table 2: Average utility scores, standard deviations, % of accepted offers, sum of utility scores and average and standard deviations of the end turns of humans versus automated agents.**

negotiations for each role, and the sums of the final utility values. When the *KBAgent* played the role of the Employer (Job Candidate) it achieved an average utility value of 468.86 (482.71), the *QOAgent* playing the same role achieved an average value of 417.37 (397.83). While in both roles the *KBAgent* achieved higher utility values, it is only significant when it played the role of the Job Candidate (using the 2-sample *t-test*:  $p < 0.001$ ). In addition, more agreements were reached due to offers proposed by the *KBAgent* when compared to the *QOAgent*. While only 7.14% and 11.76% of the agreements that were reached when the *QOAgent* was involved (in the Employer and Job Candidate roles, respectively) were due to offers proposed by the automated agent, 21.43% and 50% of the agreements that were reached when the *KBAgent* played the Employer and Job Candidate role, respectively, were the result of offers proposed by the automated agent. In the case of the Job Candidate this increase is also significant (using  $\chi^2$  test,  $p < 0.02$ ). The high increase in the percentage of agreements reached when the *KBAgent* played the role of the Job Candidate can also explain the fact that in this role the utility values for the *KBAgent* were significantly higher than values for the *QOAgent*.

The sum of both agents can be used as an indication regarding the *social welfare* of the automated agent. Comparing the sum of utility values of both negotiators when the *KBAgent* played a role to the cases in which the *QOAgent* played a role, we can see that the sum is higher in the cases in which the *KBAgent* was involved (839.36 compared to 737.87 in the Employer role and 863.14 compared to 829.61 in the Job Candidate role). These results are significant only in reference to the Employer role (using the 2-sample *t-test*:  $p < 0.04$ ). That is, the *KBAgent* not only achieves better agreements, but also enables an increased distribution of the utility points. In the case of the Job Candidate role, people negotiating against the *KBAgent* achieved higher utility values than when playing against other people or against the *QOAgent*.

We also examined the end turn in which agreements were reached when either of the automated agents were involved. While negotiations ended faster when the *QOAgent* was involved, the results are insignificant statistically. The fact that negotiations lasted longer when the *KBAgent* was involved can be contributed to the fact that its acceptance threshold was more sophisticated than the random one implemented by the *QOAgent*, allowing the *KBAgent* to achieve better agreements.

	Average Utility	Stdev	<i>p</i> -value
<b>Employer</b>			
<i>KBAgent</i>	468.86	36.96	
People vs. People	408.89	106.46	0.02
People vs. <i>QOAgent</i>	431.78	80.83	0.05
People vs. <i>KBAgent</i>	380.43	48.55	<0.001
<b>Job Candidate</b>			
<i>KBAgent</i>	482.71	57.51	
People vs. People	310.28	143.60	<0.001
People vs. <i>QOAgent</i>	320.50	112.71	<0.001
People vs. <i>KBAgent</i>	370.50	58.87	<0.001

**Table 3: Average utility scores and standard deviations of human negotiators. *p*-values are compared to the *KBAgent*'s scores.**

These results are encouraging due to the efficacy of the *KBAgent* and owing to the fact that it enables the parties to achieve better agreements than the *QOAgent*. We continued to compare the results of the *KBAgent* to the results of human negotiators. The negotiation sessions involved 36 undergraduate and graduate students playing once against each other (a total of 18 simulations). Table 3 summarizes the average utility values, standard deviations and *p*-values comparing the results of the *KBAgent* to that of the people. The results indicate that the *KBAgent* achieves significantly better utility values than those achieved by human negotiators either against other human negotiators or against other automated agents (the *QOAgent* and the *KBAgent*).

## 6. CONCLUSIONS

In this paper we presented a novel approach for general opponent modeling, proposing offers using a concession method and accepting offers using a sophisticated threshold. We showed that our approach allows the automated agent negotiate efficiently with people and even perform better than another state-of-the-art automated agent. The results demonstrate that the *KBAgent* achieved significantly higher utility values than the human players. In comparison to the other automated agent (the *QOAgent*) it achieved higher utility values and in the case of one of the two roles, even achieved significantly higher utility values.

As people negotiate in diverse ways and mostly in one-shot negotiations a general opponent modeling approach could yield better results than specific opponent modeling, as we indeed shown in the experiments. Moreover, the approach we apply has a low computation complexity and can work well on sparse databases.

Future work will match the *KBAgent* on additional domains to test the generality of its design and to bolster the confidence of the generated results.

## 7. REFERENCES

- [1] M. H. Bazerman and M. A. Neale. Negotiator rationality and negotiator cognition: The interactive roles of prescriptive and descriptive research. In H. P. Young, editor, *Negotiation Analysis*, pages 109–130. The University of Michigan Press, 1992.
- [2] S. R. Brown and L. E. Melamed. *Experimental Design and Analysis*. Sage Publications, Inc., CA, USA, 1990.
- [3] A. Byde, M. Yearworth, K.-Y. Chen, and C. Bartolini. AutONA: A system for automated multiple 1-1 negotiation.

- In *Proceedings of CEC'03*, pages 59–67, 2003.
- [4] R. M. Coehoorn and N. R. Jennings. Learning on opponent's preferences to make effective multi-issue negotiation trade-offs. In *Proceedings of ICEC'04*, pages 59–68, 2004.
- [5] R. Cohen. *Negotiating Across Cultures: Communication Obstacles in International Diplomacy*. United States Institute of Peace Press, Washington, D.C., 1991.
- [6] I. Erev and A. Roth. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibrium. *American Economic Review*, 88(4):848–881, 1998.
- [7] Y. Gal, A. Pfeffer, F. Marzo, and B. J. Grosz. Learning social preferences in games. In *Proceedings of AAAI-04*, pages 226–231, 2004.
- [8] K. Hindriks and D. Tykhonov. Opponent modelling in automated multi-issue negotiation using bayesian learning. In *Proceedings of AAMAS'08*, pages 331–338, 2008.
- [9] R. Katz, Y. Amichai-Hamburger, E. Manisterski, and S. Kraus. Different orientations of males and females in computer-mediated negotiation. *Computers in Human Behavior*, 24(2):516–534, 2008.
- [10] D. A. Lax and J. K. Sebenius. Thinking coalitionally: party arithmetic, process opportunism, and strategic sequencing. In H. P. Young, editor, *Negotiation Analysis*, pages 153–193. The University of Michigan Press, 1992.
- [11] M. LeBaron and V. Pillay. *Conflict Across Cultures: A Unique Experience of Bridging Differences*. Nicholas Brealey Publishing, Boston, MA, 2006.
- [12] T. Leonard and J. S. J. Hsu. *Bayesian Methods - An Analysis for Statisticians and interdisciplinary Researchers*. Cambridge University Press, Cambridge, UK, 1999.
- [13] R. Lin, S. Kraus, J. Wilkenfeld, and J. Barry. Negotiating with bounded rational agents in environments with incomplete information using an automated agent. *AIJ*, 172(6-7):823–851, 2008.
- [14] R. D. Luce. *Individual Choice Behavior: A Theoretical Analysis*. John Wiley & Sons, NY, 1959.
- [15] R. D. McKelvey and T. R. Palfrey. An experimental study of the centipede game. *Econometrica*, 60(4):803–836, 1992.
- [16] J. F. Nash. The bargaining problem. *Econ.*, 18:155–162, 1950.
- [17] M. J. Osborne and A. Rubinstein. *A Course In Game Theory*. MIT Press, Cambridge MA, 1994.
- [18] P. Pasquier, R. Hollands, F. Dignum, I. Rahwan, and L. Sonenberg. An empirical study of interest-based negotiation. In *Proceedings of ICEC'07*, pages 339–348, 2007.
- [19] S. Saha, A. Biswas, and S. Sen. Modeling opponent decision in repeated one-shot negotiations. In *Proceedings of AAMAS'05*, pages 397–403, 2005.
- [20] S. Saha and S. Sen. Negotiating efficient outcomes over multiple issues. In *Proceedings of AAMAS'06*, pages 423–425, 2006.
- [21] S. Siegel. *Non-Parametric Statistics for the Behavioral Sciences*. McGraw-Hill, NY, USA, 1956.
- [22] D. Traum, S. Marsella, J. Gratch, J. Lee, and A. Hartholt. Multi-party, multi-issue, multi-strategy negotiation for multi-modal virtual agents. In *Proceedings of IVA'08*, 2008.
- [23] M. Wand and M. Jones. *Kernel Smoothing*. Chapman & Hall, London, 1995.