Learning an Opponent's Preferences to Make Effective Multi-Issue Negotiation Trade-Offs

Robert M. Coehoorn
rc1603@ecs.soton.ac.uk

Nicholas R. Jennings
nrj@ecs.soton.ac.uk

School of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, United Kingdom.

ABSTRACT

Software agents that autonomously act and interact to achieve their design objectives are increasingly being developed for a range of e-commerce applications. In this context, automated negotiation is a central concern since it is the de facto means of establishing contracts for goods or services between the agents. Now, in many cases these contracts consist of multiple issues (e.g. price, time of delivery, quantity, quality) which makes the negotiation more complex than when dealing with just price. In particular, effective and efficient multi-issue negotiation requires an agent to have some indication of its opponent's preferences over these issues. However, in competitive domains, such as e-commerce, an agent will not reveal this information and so the best that can be achieved is to learn some approximation of it through the negotiation exchanges. To this end, we explore and evaluate the use of *kernel density estimation* for this purpose. Specifically, we couch our work in the context of making negotiation trade-offs and show how our approach can make the negotiation outcome more efficient for both participants.

1. INTRODUCTION

Software agents that autonomously act and interact to achieve their design objectives are increasingly being developed for a range of e-commerce applications (see [8] for a review). In such agent-mediated applications, a key component of the solution is the way in which the agents negotiate to establish contracts with one another to provide particular services or goods under particular terms and conditions.

In many cases, it is important that the agents do not only bargain over the price of a service, but also take into account aspects like delivery time, quality, and payment method. Moreover, in such *multi-issue negotiations*, it is often possible to reach an agreement that is mutually beneficial for both parties [11]. This opportunity of joint improvement is provided by the difference in importance (weight) attached to the different issues by the different agents. However, an impediment to this win-win scenario occurs in many e-commerce settings because the agents are unlikely to truthfully reveal their preferences, utility functions or reservation values for fear of being exploited. In such circumstances, the best that can be achieved is to try and approximate these preferences based upon past experience and the offers and counter-offers that the opponent makes during the current negotiation encounter.

To this end, this paper reports on the development of a novel method for attempting to learn the negotiation preferences of the opponent. Specifically, we try to learn this information with respect to the provision of a particular service (since an agent's preferences may vary for different services). The particular approach we use is *kernel density estimation* (*KDE*) which is a statistical method known to provide a simple way of finding structure in data sets without the imposition of a parametric model [15]. It works in the following way; any data that is available about previous negotiation encounters for the provision of a particular service is processed offline (as described in section 3.2) to acquire a probability density function over the opponent's likely weights for the various issues. This function can then be augmented by online learning that reflects new information emerging from the ongoing encounter.

The KDE-method was chosen for two main reasons. First, the computational complexity of the model is important because agents are bounded in their computational power. With KDE, the lookup of a prediction is constant, much of the learning can be performed offline and the online learning has $n \log n$ complexity (as discussed in section 3.2). Second, we want to make as few explicit assumptions about the relation between the negotiation history and the importance of the negotiation issue as possible (so the method can be used for a wide variety of opponents with varying strategies without having to fundamentally alter the model). Since KDE is a non-parametric method, we do not have to make any assumptions about the relation between time, negotiation history, and issue-weight. In contrast, when using parametric regression methods (such as linear regression or the expectation-maximization algorithm) an assumption must be made about the underlying distribution function (respectively linear and gaussian). Similarly, in Bayesian learning an a priori distribution must be given for the weights of the opponent, whereas in KDE the initial distribution is based

solely on the training data.

We choose to evaluate the efficacy of KDE in the context of making negotiation *trade-offs* in bilateral encounters. We focus on trade-offs because they are a key feature of bargaining behaviour and cannot be achieved without a reasonable degree of information about an opponent. In making a trade-off an agent concedes on one issue and demands more on another. Overall, the aim is for the agent to keep the same utility for itself, but increase the utility of the opponent, hence making the trade-off more likely to be accepted by the opponent. For the actual computation of the trade-offs, we use Faratin's algorithm [4] which evaluates possible trade-offs based on fuzzy similarity. Thus while this algorithm makes use of the fact that different negotiation issues are of different degrees of importance to the agent, it does not actually provide a method for learning these weights.

Against this background, this research advances the state of the art in the following ways. First, KDE has never been used to learn the preferences of a negotiation opponent. Moreover, although we demonstrate it in the context of making negotiation trade-offs, the method can be applied to other aspects of negotiation where more accurate knowledge about an opponent's preferences can lead to better negotiation outcomes. Second, we extend Faratin's trade-off algorithm by incorporating a learning model, thereby making it more effective in finding trade-offs in a wider variety of circumstances.

The remainder of the paper is structured as follows. Section 2 discusses related work in the area of negotiation and learning in negotiation. After that, we describe the theoretical models that underpin this work; namely, the negotiation model, the KDE-method and the trade-off algorithm (section 3). The experiments and their analysis are presented in section 4. Finally, in section 5 we present the conclusions and outline avenues for future work.

2. RELATED WORK

In this section we will briefly review relevant work in the areas of automated negotiation (in general) and learning in negotiation (in particular).

When considering research in automated negotiation, three broad topics need to be dealt with [9]. First, *negotiation protocols* are the set of rules that govern the interaction. These fall into two broad camps: *auctions* and *bilateral negotiations*. Here, we consider the latter. Such protocols involve two parties (a service supplier and a service consumer) and, generally speaking, an alternating offers protocol (in which the parties take turns to submit offers and counter-offers until they come to a mutually acceptable agreement over the terms and conditions of a trade or one of the parties withdraws (typically because its negotiation deadline has passed)).

The second broad topic for research are the *negotiation contracts* which specify the range of issues over which agreement must be reached. In this area, Fatima et al. [5, 6] have analysed the optimal outcomes of single-issue negotiations, and multi-issue negotiations in which the issues are negotiated sequentially. A key question that arises in the latter case is the order in which the issues are negotiated or the *agenda*. When the agenda is *endogenously* defined, i.e. the agents are allowed to decide which issue they will negotiate next during the process of negotiation, a unique equilibrium exists [6]. In [5] the agenda is defined partly endogenously and partly exogenously (i.e. before the negotiations). However, this scenario requires a mediator to identify the optimal scenario, and is therefore not compatible with our requirements. Given this, we consider agendas that are set entirely exogenously and in which all the issues are considered simultaneously.

Finally, the most important topic for this research is the *reasoning models* which provide the decision making methods the agents employ to compute their negotiation moves. Within this area, the importance of learning from past negotiation experiences was first recognised in Sycara's work on the PERSUADER system [14] which modeled an iterative process of multi-issue negotiation. It uses past agreements between similarly situated parties to suggest proposals that might succeed in the current negotiations. When conflicts arise, a mediator engages in parallel negotiations with the parties, either to change the proposal to something that is acceptable, or to attempt to change the belief of the disagreeing parties using persuasive argumentation. In our domain, however, mediation is undesirable because of the competitive nature of the encounter and the desire to keep information private.

After this, a variety of learning techniques have been used to try and improve the effectiveness of the agents' negotiation capabilities. For example, genetic algorithms have been used to discover effective negotiation strategies (e.g. [7, 10]). In this research, agents are modelled as chromosomes and the parameters of the negotiation model are genes in the chromosome. By evolving these agents, the benefits and drawbacks of a number of negotiation strategies are assessed. However, this method typically focusses on finding the most optimal offer/counter-offer strategy, whereas we want an explicit reasoning model about the opponent.

Probably the most widely used paradigm is *Bayesian learning* (e.g. [1, 16]). In this line of work, the estimates of the probability of a set of hypotheses about the opponent's preferences or reservation prices are produced, given the previous negotiation encounters. However, a significant drawback of Bayesian learning is that the agent has to have a priori knowledge about the probability distribution of the likely outcome of the negotiation. This is difficult to provide because of the private nature of the information needed to compute this. Furthermore, many encounters are necessary before a good model can be provided because individual opponents are modeled, whereas in our work the strategy (in terms of the agent's preferences) with respect to the provision of a particular service is modeled.

Probably the work that is most closely related to ours is [13]. In this work, a reinforcement learning algorithm is presented that learns to propose a solution that is more likely to be accepted by the opponent. Specifically, offers that are rejected by its opponent are treated as negative instances for the learning algorithm, while counter-proposals from its opponents give a positive reward. However, their model does not combine the learning model with a reasoning model that models the properties of the problem domain as the fuzzy similarity method of Faratin does (see section 3.3 for details). This means their model is less robust when the prediction about preference weights is not completely precise.

3. THEORETICAL MODELS

In our work, three models can be distinguished. First, the model that describes the negotiation (see section 3.1). After that, section 3.2 will describe the KDE-method we use for predicting the weights of the opponent. Finally, we present the method for calculating the trade-offs (section 3.3).

3.1 The Negotiation Model

Before evaluating the negotiation model, let us first start with defining some of the necessary parameters: If I is a pair of (self-interested) negotiating agents ($I = \{a, b\}$), let i ($i \in I$) represent a specific negotiating agent, and J ($J = \{1, \ldots, n\}$) be the issues under negotiation in a given encounter. For each issue j ($j \in J$), every agent has a lower and an upper reservation price, respectively min_j and max_j, resulting in a domain for each particular issue: $\mathcal{D}_j = [min_j, max_j]$. These values represent the value which is the best reasonable value expected and the worst value still acceptable for the agent. In all negotiation interchanges, every issue gets assigned a value $x_j \in \mathcal{D}_j$. To evaluate a value of an issue, each agent has a scoring function over this domain: $V_j^i : \mathcal{D}_j \rightarrow [0, 1]$ which assigns a valuation to every possible value x_j. Finally, each agent has a weight vector over the issues, representing the relative importance it attaches to the issues, where w_j^i is the importance agent i attaches to issue j. We assume these weights are normalized (i.e. $\forall i \in I : \sum_{1 \leq j \leq n} w_j^i = 1$). Thus, the utility of agent i over a *contract* $\vec{x}$, a set of values for all issues, can be defined as:

$$u^i(x) = \sum_{1 \leq j \leq n} w_j^i V_j^i(x_j) \tag{1}$$

Now, if all this information is known to both agents, the *Pareto-optimal* set can be calculated by both of them [11]. However, in e-commerce settings, several of the assumptions needed for this calculation are not tenable. First, an agent will not give out information about its reservation prices, the weights over the issues and its utility function because doing so would enable it to be exploited. Second, each agent will have distinctive reservation prices (whereas in [11] these were defined to be equal for all agents). The zone of agreement $\mathcal{Z}$ we will use is thus defined as the intersection of the individual domains: $\mathcal{Z}_j = \bigcap_{i \in I} \left[min_j^i, max_j^i\right]$. Finally, the resources, and especially the time available for the negotiations, will be bounded. Specifically, we assume each agent has a (hard) deadline, denoted as t_{max}^i, by when it must have completed the negotiation. The negotiation protocol is a two-agent variant of Rubinstein's model of *alternating offers* [12]. Specifically, let $x_{a \rightarrow b}^t$ be the offer of agent a to b, at time t and $x_{a \rightarrow b}^t[j]$ denote the value of issue j of this offer. Note that in this model time is discrete. The agent who has the first turn is chosen randomly. After the first offer, at every timestep the agent who received the last offer decides whether to accept the offer, propose a counter-offer or withdraw from the negotiation. This continues until the reaction is one of the communication particles $\{accept, withdraw\}$.

The agents decide which of the alternatives to choose from definition 1 (taken from [3]). When the deadline of the agent has passed, the agent withdraws. Otherwise, the reaction depends on the offer of the opponent. If this offer has a higher utility than the offer the agent itself is prepared to make at that moment ($x_{a \rightarrow b}^{t'}$, calculated as per equation 3), the offer of the opponent is accepted. In the other case, the calculated counter-offer will be made.

DEFINITION 1. *Given an agent a and its associated scoring function V^a, the **interpretation** by agent a at time t' of an offer $x_{b \rightarrow a}^t$ sent at time $t < t'$, is defined as:*

$$I^a\left(t', x_{a \rightarrow b}^{t'}\right) = \begin{cases} \texttt{withdraw} & \texttt{If } t' > t_{\max}^a \\ \texttt{accept} & \texttt{If } u^a(x_{b \rightarrow a}^t) \geq u^a(x_{a \rightarrow b}^{t'}) \\ x_{a \rightarrow b}^{t'} & \texttt{otherwise} \end{cases}$$

The scoring function over the issues, as required by equation 1, is given by the distance to the worst bid acceptable to this agent, relative to its range of acceptable values. Hence this scales the acceptable bids for the agent to the domain $[0, 1]$:

$$V_j^a(x_j) = \begin{cases} \frac{x_{a \rightarrow b}^t[j] - \min_j^a}{\max_j^a - \min_j^a}, & \text{if increasing} \\ \frac{\max_j^a - x_{a \rightarrow b}^t[j]}{\max_j^a - \min_j^a}, & \text{if decreasing} \end{cases} \tag{2}$$

where increasing and decreasing refer to the direction of change in score with increasing value of the issue.

To calculate the counter-offers used in definition 1, a range of different strategies could be deployed. Given the time-constrained nature of our domain, however, we use the family of *polynomial strategies* (as advocated by [4]):

$$x_{a \rightarrow b}^t[j] = \begin{cases} \min_j^a + (1 - \alpha_j^a(t))(\max_j^a - \min_j^a) & \text{if increasing} \\ \min_j^a + \alpha_j^a(t)(\max_j^a - \min_j^a) & \text{if decreasing} \end{cases} \tag{3}$$

where the function α is a polynomial function dependent on the time remaining, and β is the strategy parameter, defining the form of the function:

$$\alpha_j^a(t) = \left(\frac{\min(t, t_{max}^a)}{t_{max}^a}\right)^{\frac{1}{\beta_j}} \tag{4}$$

When this strategy parameter is between 0 and 1, the agent will make an offer and stick to it: only when the deadline comes near it will concede towards the upper reservation price. When the parameter is greater than 1, the higher the value, the faster the agent will concede towards its upper reservation price.

Note that since the counter-offer acquired in this manner is dependent on the time remaining, the interpretation of an offer as per definition 1 will also be (indirectly) time-dependent.

3.2 Kernel Density Estimation

The only information we can definitely assume to be available to the agent is its negotiation history. This covers the offers and counter-offers of all its previous negotiations for a particular service[1]. Therefore, our aim is to obtain an estimate of the opponent's weights by only looking at this history. In particular, we consider the difference between the opponent's last two offers in a given encounter and try to find a relation between this difference and the weight the opponent places on various issues. For example, a relatively small change in the bid on one issue at the beginning of the negotiation process might indicate that it is more important to the opponent than the other issues. Likewise,

[1]This history can be on a per agent basis or can cover all agents with which the modeller has interacted for the particular service in question.

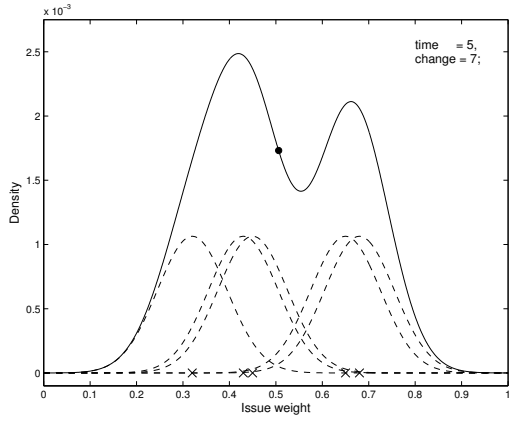


Figure 1: Kernel density estimation example. The dashed lines are the kernels and the solid line the estimate. The individual observations are marked with a cross and the predicted weight for the negotiation issue in question with a dot.

a relatively large concession towards the end of the negotiation might indicate that this issue is important and the opponent is performing a final concession on that issue to save the agreement.

The method we will use to find this relation is *kernel density estimation* (for reasons outlined in section 1). In more detail, the basis of this method is the *kernel*: a function K satisfying $\int K(X)dx = 1$. Intuitively, these kernels can be seen as representing a "probability distribution" of size $1/n$ (where n is the number of observations) associated with each data point, about its neighbourhood [15]. This is illustrated in figure 1, where a two-dimensional estimate is based on five observations[2]. This example could be viewed as the estimate of the opponent's weight for a particular negotiation issue, given a specific time in the negotiation process and the relative difference of the two last bids on an issue. In this case, as indicated in the top-right corner of the graph, we see that these observations, for example from simulations, took place at the beginning of a negotiation encounter (time = 5) and there was a relatively small change between two consecutive offers (7 percent of the total change). Here, the dashed lines are the *kernels*. It can be shown that the particular unimodal distribution used as a kernel does not degrade performance [15] and so we choose the $N(0, 1)$ distribution for reasons of computational efficiency. The kernels are formed by centering a kernel at each observation (e.g. the difference between two consecutive offers and the believed weight of the issue). Note the kernels are scaled by the total number of observations and thus the value of the kernel estimate at point x is simply the sum of the scaled kernels. For example, in figure 1 it can be seen that there are no observations for a very low weight and therefore the density of 0.1 has a low value. On the contrary, there are relatively many observations around a weight of 0.4 and hence the probability estimate has a high value here. From this distribution the weight predicted for the negotiation issue in question is the expected value, using the estimate as a probability density function (indicated with a circle in the graph).

In this illustrative example we used two-dimensional kernels. However, the kernels we use in this research are in fact three-dimensional (the difference of two consecutive offers, the weight of the issue and the probability density of this weight given this difference). Furthermore, since we assume the opponent uses a time-dependent strategy, we expect the agents to behave differently over time. Therefore, for each time t, we make such a density estimate to predict the density of a certain weight given the relative difference between the last two offers.

One of the most important issues in this work is the *bandwidth* or the amount of spread of the kernels. In figure 1, a smaller bandwidth would lead to higher but smaller kernels, whereas a higher bandwidth would lead to broader and lower kernels. Here we use the *solve-the-equation* (STE) rule as suggested by [15] to calculate this bandwidth. This method takes an initial guess of the bandwidth and calculates a new value, using the error in the resulting density estimate, until the process converges. The reasons for choosing this method are its good performance on our domain (due to the direct feedback from the results of previous values), and the ease of computation.

Since the negotiating agents have bounded computational resources, it is important to determine the computational complexity of KDE. To this end, when the probability density estimate is learnt, the prediction of a weight is a table lookup of constant time. The learning algorithm calculates this estimate by using a fourier transformation of the kernels. The complexity is determined by the use of a convolution, which can be performed in $\mathcal{O}(n \log n)$ time (where n is the sampling rate, taken to be 128). This is also the complexity of the bandwidth estimation, since the error of the estimate is used in the computation and thus the estimate has to be formed.

3.3 Trade-Offs Based on Similarity Criteria

To solve the problem of finding a trade-off when agents have incomplete information about their opponent we exploit Faratin's algorithm [4]. This works by performing an iterated hill-climbing search in the landscape of possible contracts. The search starts at the opponent's last offered contract and proceeds by successively generating contracts whose utility is progressively closer to the desired threshold of the agent making the trade-off. During this search the contract that maximizes the similarity to the opponent's last offering is used as the starting point of the next iteration. This algorithm is chosen because it has a number of desirable properties: (i) its complexity is linearly proportional to the number of issues under consideration and (ii) by using the notion of fuzzy similarity the uncertainty of an agent's belief over the preferences of the other agents are modelled as fuzzy relationships between values of the domain (and not the other agent's actual preferences). Hence, the algorithm models the domain of the issues under consideration (the problem domain), instead of the individual agents (recall the discussion at the end of section 2). By extending this basis with a separate learning model, as per section 3.2, we obtain a hybrid algorithm which makes the trade-off algorithm more adaptive and robust.

In more detail, the algorithm consists of two steps, assu-

[2]We should point out that we use just five observations here purely for clarity in illustrating how the kernel method works. Practical density estimation usually involves a much higher number of observations.

ming that agent a has to make a bid:

i) Find the iso-curve for a, that is, the set of contracts $x^{t'}_{a\to b}$ that have the same utility for agent a as its previous offer (i.e. $x^{t}_{a\to b}$),

ii) Out of this set, take the contract that agent a *believes* is most preferable to the opponent, and send this as the trade-off counter-offer.

In the second step, the aforementioned *fuzzy similarity* is used. The algorithm thus tries to find the deal that is most "similar" to the previous offer. The rationale behind this is that a deal which is similar to an acceptable offer of your opponent has a reasonable probability of being acceptable itself.

In this context, the notion of similarity between two valuations of issue j, $x_j, y_j \in \mathcal{D}_j$ uses a criteria evaluation function $h : \mathcal{D} \to [0, 1]$ which maps the value of the issue to a valuation between zero and one[3]. When comparing two values for a specific issue, the issue-similarity is defined by comparing the values of the function h:

$$\mathrm{Sim}_j(x_j, y_j) = 1 - |h(x_j) - h(y_j)| \tag{5}$$

The similarity of two contracts is then defined by the sum of the issue-similarities weighted by the opponent's weight of that issue:

$$\mathrm{Sim} = \sum_{j\in J} w^b_j \cdot \mathrm{Sim}_j(x_j, y_j) \tag{6}$$

This results in the following formal definition of the algorithm:

DEFINITION 2. *Given an offer x from agent a to b, and a subsequent counter offer y from agent b to a, and given that $\theta = u^a(x)$, agent a defines its trade-off proposal with respect to y as:*

i) $\mathrm{iso}_a(\theta) = \{x \mid V^a(x) = \theta\}$.

ii) $\textit{trade-off}_a(x, y) = \arg\max_{z\in \mathrm{iso}_a(\theta)}(Sim(z, y))$.

To increase the exploration of the space of possible deals, the algorithm starts at the utility of the contract of the opponent (y), and takes S steps in increasing the value of θ, until the utility of the previous proposal x is reached. The number of elements generated in step i is defined by N.

While this algorithm has been showed to be effective in a number of scenarios [3], a major shortcoming is that the weights the opponent used in calculating the similarity between two offers (see equation 6) are private information and thus unknown to agent a. Given this, the aim of our research is to see if the KDE-method outlined in section 3.2 is effective at learning this information using only the negotiation history.

4. EXPERIMENTAL ANALYSIS

The aim of these experiments is to examine the effects of using a weight vector, as predicted by kernel density estimation, on the performance of the trade-off algorithm. Since the utility of this trade-off remains the same for the agent itself (see definition 2), we measure performance in terms of the opponent's utility of the proposed trade-off contract. Specifically, we start by looking at a single offer to investigate how the prediction of the weights by KDE influences the performance of the trade-off algorithm. After that, we investigate the effects of the KDE-method on a complete negotiation process by analysing agents which use the predictions in their negotiation strategy.

4.1 Single Offers

These experiments investigate the effect of using the KDE-method to calculate a single trade-off. Thus a specific instance of a negotiation session is considered, resulting in a single offer – counter-offer pair. Due to the number of private and thus unknown dependent variables, we do not expect the prediction of the KDE to be completely precise. Therefore, we analyse the results when a random perturbation is added to the real weight of the opponent. After that, we explore to what extent a priori knowledge of the opponent is necessary for good performance of the trade-off algorithm. We do this by looking at the concrete effects of using KDE's formed under different levels of knowledge about the opponent. Finally, to test whether using KDE increases the performance with respect to the method proposed by Faratin, we compare the performance to a situation in which uniform weights are used. All experiments described here follow the settings as described in 4.1.1 unless stated otherwise.

4.1.1 Random Perturbation of Real Weights

For this experiment a negotiation between two agents (that generate their offers according to equation 3) was undertaken to get a range of plausible contracts (using the model described in section 3.1). Specifically, the contracts consist of four issues of unequal weight that remain fixed throughout the negotiations ($w_b = \{0.5, 0.1, 0.05, 0.35\}$, $w_s = \{0.1, 0.5, 0.25, 0.15\}$). For simplicity it is assumed for all issues that an increase of value results in a utility gain for one agent ("the seller", or agent s), and a utility decrease for the other agent ("the buyer", or agent b).

From this negotiation, we take two consecutive proposals: a bid x of b and a counterproposal y of s. Using these two proposals, a trade-off for the buyer is calculated (as per definition 2). Specifically, we consider two environments (A, in the beginning of the negotiations, and B, later in the negotiations) in which the utility of b varies. In environment A, the utility of agent b is high (because it has not conceded much with its time-dependent strategy), hence there will be fewer contracts on its iso-curve. In environment B, however, the utility of b is much lower (because time has passed and it has conceded utility). This results in more contracts on the iso-curve, and, consequently, more opportunity to improve the utility of agent s.

For the parameters of the trade-off algorithm we will use the values of [4]: 40 steps (S) using 100 children per step (N). More steps or more children are shown not too improve the performance of the algorithm. The similarity function $h(x)$ used in equation 5 is parameterized to be a linear function. In this way, the discrimination of two contracts will be similar across the domain of the issue. Since there is a random factor in the trade-off algorithm, the proposals are calculated 50 times (a significant sample size, at the

[3]The function used here has the form of a sigmoid to have an optimal discriminability in the, most important, middle part of the domain of reservation prices:
$h(x)=\frac{1}{\pi}\tan^{-1}\left[\left(\frac{2|x-\min|}{x-\min}\cdot\left|\frac{x-\min}{\max-\min}\right|^{\alpha}-1\right)\tan\left(\pi(\frac{1}{2}-\varepsilon)\right)\right]+\frac{\pi}{2}$

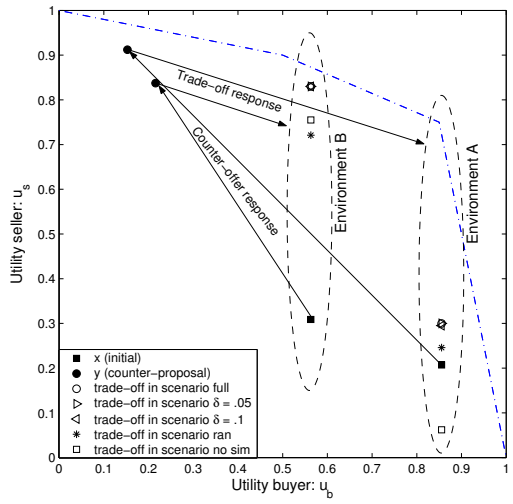


Figure 2: The mean utility of the proposed trade-offs using perturbed weights.

0.95 confidence level, with confidence interval of 15%). We will use the mean utility of these contracts to describe the results; meaning that a higher value indicates better performance.

The experimental variable in this experiment is the correctness of the weights used for the opponent (as used in equation 6). In the first scenario, the agent has perfect knowledge (scenario *full*). The second scenario involves a small perturbation of the opponent's real weight vector to evaluate the effect of a small error in the prediction of the weights. This perturbation is chosen uniformly out of the domain [-0.05, 0.05] (scenario $\delta = 0.05$) and [-0.1, 0.1] (scenario $\delta = 0.1$). For reference purposes, the results of two benchmark situations are added. First, to check what happens if the prediction is completely arbitrary, a randomized weight vector (satisfying $\sum_J = 1$) is used in the scenario *ran*. In the second situation, we do not use any prediction about the opponent at all, but just choose a child randomly in each step of the trade-off algorithm to recurse into (*no sim*). This gives us a means of comparing our results to the case in which similarity criteria are not used at all.

Since the change of the bid for the opponent depends on multiple unknown variables (as can be seen in equation 3), we cannot expect the prediction to be completely precise. The objective of this experiment is therefore to analyse the effects of using such inaccurate information. It is important to note that here, in contrast to Faratin [4], we do not make the assumption about the preservation of ordinality in the opponent's weights. Thus, we will evaluate the following hypotheses:

HYPOTHESIS 1. *A small error in the prediction of the opponent's weights will not significantly degrade the performance when compared with the perfect information case.*

HYPOTHESIS 2. *The prediction of the weight vector does not have to be ordinally perfect to improve performance compared with the case of using no knowledge about the opponent.*

The results of this experiment are presented in figure 2, where the utilities of the agents are plotted against each other. The line joining (0,1) to (1,0) is the Pareto-optimal line, calculated using the weighted method [11]. In this case an ideal trade-off would be on the Pareto-optimal line. Therefore, the closer our trade-offs are to this, the better they are. Since environments A and B are reached using non-optimal methods, however, we will just look at the order of the contracts, and not the concrete utility reached. In more detail, the figure illustrates a part of a negotiation process. The filled squares are the initial contracts proposed by the buyer in the two environments. As indicated by the arrows, the counter-offers of the seller are indicated with the filled circles. After receiving this counter-offer, the buyer calculates a trade-off response (for the different scenarios mentioned above) which are plotted within the different environment ellipses.

As can be seen, for both environments, the performance of using weights that are perturbed does not decrease the performance of the trade-off algorithm. This is tested by an *analysis of variance* (*ANOVA*) extended with a *Tukey-Kramer* test as post-hoc test [2], which indeed shows that in both environments there is no significant difference between the proposed contracts[4] when using full knowledge of the weights of the opponent or a perturbed weight. This can be explained by the combination of the KDE-method with the similarity-based method. Specifically, the resulting algorithm uses both the prediction of the weights by the KDE-method and the fuzzy similarity as a basis for computing trade-offs. This combination leads to a robust overall performance because errors in the prediction are compensated for by the fuzzy similarity method, and the fuzzy similarity method is enhanced by better information about the opponent.

The contracts produced by the control methods are significantly worse than the performance of the full knowledge (as expected). If we look at the *no sim* case, the contract to use in the next iteration is selected randomly from the children generated which leads to poor performance because no attempt is made to maximize the opponent's utility. In the *ran* case, the weights of the opponent are randomized which means that although the algorithm does perform a maximization, it maximizes the situation in which the weights are different from their actual values.

Note that given the weight vectors used, the ordinality of the weights can change in both environments due to the perturbation. However, in the $\delta = 0.05$ scenario, only the order of the first and fourth issue can change, whereas in scenario $\delta = 0.1$ also the third issue can be at different places in the ordering. In either case, however, the absence of a significant difference between these scenarios indicates that this does not make a significant difference.

Overall, these results indicate that both hypotheses can be accepted. The algorithm is robust enough to achieve similar performance to the full information setting when the information is less than perfect. Moreover, this result is independent of the preservation of ordinality of the weights. When taken together, these results show that the prediction by the KDE does not have to be completely accurate to improve the performance of the trade-off algorithm.

4.1.2 Kernel Density Estimates of Weights

As we have shown that approximate values for negotiation preferences can be used to produce effective trade-offs, the

[4]Using a confidence level of 0.05.

	β	$t^s_{\max}$	RP
strat20	$[[0.1, 0.3]^{0.01}, [4, 6]^{0.1}]$	65	fixed
strat100	$[[0.1, 1]^{0.01}, [1, 10]^{0.1}]$	65	fixed
bdl	$[[0.1, 1]^{0.01}, [1, 10]^{0.1}]$	[30,100]	fixed
all	$[[0.1, 1]^{0.01}, [1, 10]^{0.1}]$	[30,100]	normal

Table 1: Variable initialisation when learning the kernel density estimates.

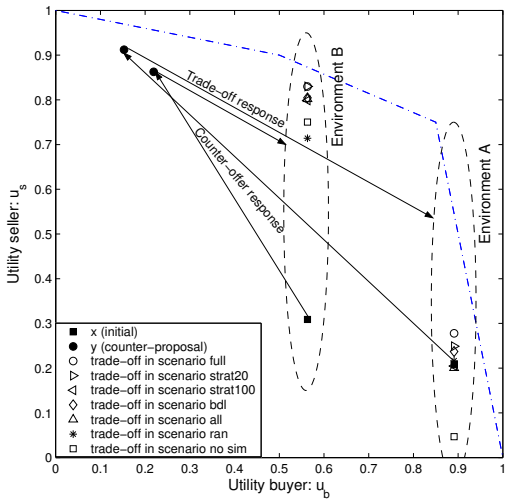
next step is to verify that the KDE-method can indeed give a prediction that is sufficiently accurate to use in the trade-off algorithm. Thus, instead of the real weights with a random perturbation, we now use the prediction of the opponent's weights under different levels of knowledge (which are represented by different kernel density estimates). The key difference between these estimates is the information about the opponent that is known to the agent. Since the opponent's bidding strategy is influenced by the deadline of the negotiation, $t^a_{\max}$, the reservation prices, RP, and the strategy parameter β (as can be seen in equation 3), these are the parameters varied in the different scenarios presented below. By this means we analyse the effect of the amount of knowledge, and therefore the precision of the prediction, on the efficiency of the trade-off algorithm.

It is our belief that the parameter which will be most difficult to obtain information about in our domain is the strategy parameter. It is hard to deduce from past experiences and may change for each individual encounter. Therefore, we start in the scenarios *strat20* and *strat100* with the strategy as a private parameter and the reservation price and the deadline as public information. In this case, a KDE is used which has learnt from opponents with varying βs, chosen uniformly out of the domain. The difference between the two is that the size of this domain for *strat100* is much bigger (two times one hundred possible values versus two times twenty possible values with the same stepsize). This means the buyer is much less sure about the opponent's strategy in *strat100* than in *strat20*. In the next scenario *β-and-deadline*, or *bdl*, again the strategy parameter is varied, but now the deadlines are uncertain as well. This reflects the fact that agents in e-commerce will have deadlines which change over time, depending on the starting time of the negotiation. Therefore, a probability distribution based on past experiences may not be particularly useful and it will be hard to get. Finally, in *all*, all the parameters are uncertain. It should be noted however, that in all cases the reservation prices are known to be in a normal distribution (scaled to a lower $\overline{\text{rp}}$ of 10, an upper $\overline{\text{rp}}$ of 20, and a standard deviation of 1.5)[5]. See Table 1 for the specific values of all the scenarios. Note that the domains of the uniform distributions are made discrete and the stepsize, if it is not equal to one, is stated as a superscript.

Given this, the particular hypotheses we sought to evaluate here are as follows:

HYPOTHESIS 3. *Using kernel density estimates will result in a higher utility than not using any knowledge about the opponent.*

[5]This standard deviation is based on the price-distribution of products often sold on the internet – like digital cameras and laptops – on price comparison sites (e.g. *kelkoo*: www.kelkoo.co.uk) and auction-sites (e.g. *eBay*: www.ebay.co.uk).



	Without KDE			
	x	*full*	*rand*	*no sim*
A: u_s	*0.206*	0.278	0.217	0.047
B: u_s	*0.309*	0.830	0.714	0.750

	With KDE			
	strat20	*strat100*	*bdl*	*all*
A: u_s	0.249	0.205	0.237	0.201
B: u_s	0.830	0.799	0.804	0.802

Figure 3: The mean utility of the proposed trade-off using different knowledge levels.

HYPOTHESIS 4. *The more knowledge about the opponent that is used in the kernel density estimation, the higher the achieved utility will be.*

The first hypothesis captures the expectation that when using KDE to predict the weights of the opponent, the performance of the proposed trade-offs will improve. If this were true, it would justify the use of KDE as a learning method that can be added to the trade-off algorithm. The second hypothesis reflects the expectation that KDE should perform better when there is more information about the opponent. In particular, we expect the performance of the trade-off algorithm to improve when the predictions are more precise, and thus we expect a higher utility when the agent has more a priori knowledge about its opponent.

The results are presented in figure 3. Since some of the values lie close to each other, the results are also presented in the accompanying table. As can be seen, especially in environment A, all values are close to each other. This can be explained by the fact that there are fewer contracts on the iso-curve compared to environment B. Using an *ANOVA* shows that the contract reached when using complete knowledge is significantly better than the other contracts, while the contract reached when not using similarity criteria performs far worse than the rest. The analysis further shows that *strat20* and *bdl* perform better than *strat100* and *all* and the random strategy performs surprisingly well in this environment (the results of the ANOVA show that there is no significant difference between the random strategy and the KDE-strategies). These results occur because the space of improvements is small due to the comparatively small number of points on the iso-curve, and, in turn, the error when using incorrect information is smaller.

In the second environment, the differences are more visible. Again, the performance using the correct weights of the opponents performs best. This time, however, the prediction of the KDE using *strat20* is good enough to result in the absence of a significant difference compared to the complete knowledge scenario. The *no sim* situation is, as before, worse than all the outcomes using a KDE, although it outperforms the *random* situation now. We believe the two most likely scenarios in e-commerce are *bdl* and *all* (where the knowledge about the opponent is minimal) and, as our results show, in these cases KDE performs well, indeed nearly as well as the situation where full information is assumed. This indicates that the predictions made by the KDE in such situations are sufficiently precise to use in the trade-off algorithm.

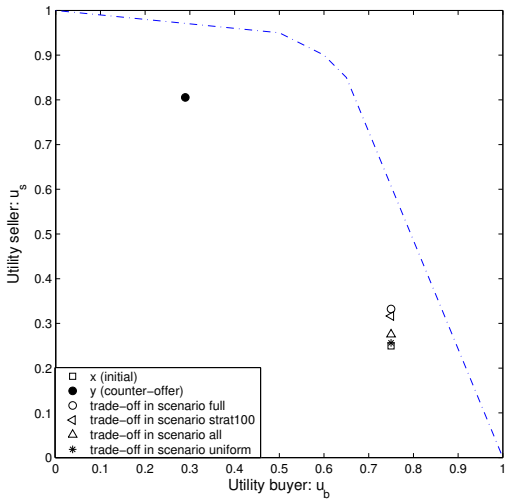
In general, our results can be explained by the precision of the prediction of the weights. When this information is completely correct (as in situation *full*) the performance is best. When the knowledge about the opponent degrades, the prediction gets less accurate, and the performance of the algorithm gets worse. We show that the kernel density method performs at least as well as having no knowledge at all when there are few contracts on the iso-curve. However, the results are far better when more possible trade-offs are available. This indicates that hypothesis 3 can be accepted. Moreover, the amount of information used while learning the KDE does not necessarily improve the performance (as is the case in environment A). However using partial information does affect the performance (as shown in environment B). This implies hypothesis 4 has to be accepted, although the similarity basis of the trade-off algorithm ensures the effect is not as strong as expected (because it makes the trade-off algorithm more robust to errors in the prediction).

4.1.3 Uniform Weights

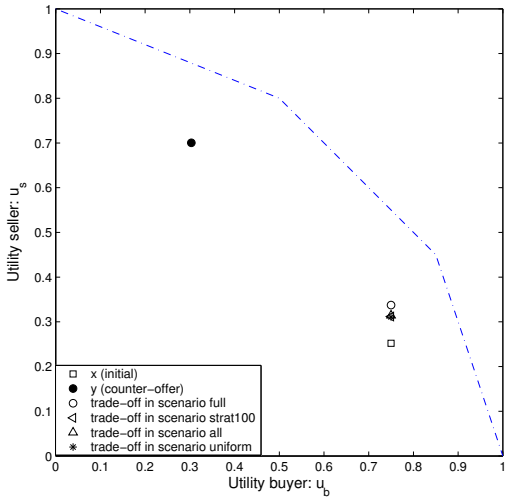
Now we have shown that KDE can be used effectively in combination with the trade-off algorithm, we want to show that our method outperforms the main method advocated by Faratin. Following his experiments on the performance of the similarity-algorithm, he concluded that the best policy for computing trade-offs is to assign uniform (equal) weights to all decision variables which can be updated by some learning rule (which is unspecified) [4]. To this end, we want to see how this works compared to using a KDE. Note that when the results are analysed, it must be taken into account that when the real weight vector of the opponent is close to uniform, the results of using a uniform weight vector are obviously going to be better than when the weight vector of the opponent is far from uniform. To account for this, we consider two environments; one with a uniform opponent and one with a strongly skewed weight vector (i.e. the former uses $w = \{0.2, 0.3, 0.15, 0.35\}$ as its weight vector, whereas the latter uses the weight vector $w = \{0.85, 0.05, 0.05, 0.05\}$).

To make a better comparison between these two environments, we want the utility of the starting contract to be similar. However, since the weight vectors used for the seller differ, this results in different contracts.[6] To achieve this we set the weight vector for the buyer as before. Since we showed in the last experiment that the influence of the level of knowledge is limited, the results will only be compared between the situations *strat100* and *all*, and to the performance when having all knowledge (*full*). Specifically, we hypothesize that:

[6]Note that a different weight vector also results in a different Pareto-optimal set.



(a) Non-uniform opponent



(b) Uniform opponent

Figure 4: The mean utility of the proposed trade-off compared to the performance using uniform weights.

HYPOTHESIS 5. *Using kernel density estimation to predict the weights of the opponent will result in a utility at least as good as using uniform weights.*

The results of the experiment are shown in figure 4. In the upper graph, the opponent uses the skewed weight vector. In this case, an analysis of variances shows that the four different groups are significantly different from one another. In particular, assigning uniform weights performs significantly worse than using the kernel density estimate for the group *all*. This is as expected due to the discrepancy of the real weight vector and the uniform vector used in the computations. When the weight vector of the opponent is near uniformity, as plotted in figure 4(b), assigning uniform weights performs as well as using the kernel density. This is as expected because the real weights and the weight vector used

do not differ from each other by much. Thus hypothesis 5 is accepted.

4.2 Complete Negotiations

Having shown that using KDE in combination with Faratin's trade-off algorithm improves the performance in the single-offer case, we will look at the results of using KDEs in a complete negotiation encounter. Henceforth, we will use the KDE-method in combination with a *meta-strategy* (i.e. a strategy of choosing which offer-generation strategy to use) which depends extensively on the trade-off algorithm. Our aim is to determine whether the more efficient individual trade-off offers will also result in a better negotiation outcome over a complete encounter.

The particular meta-strategy we will use is the *smart*-strategy [4], which is shown to have a good performance against a variety of other strategies. The smart-strategy consists of deploying a trade-off mechanism (as per section 3.3) until the agent observes a deadlock in the closeness of two offers. Such a deadlock situation is observed when the similarity between these two offers is smaller than δ (set to 0.05 in this experiment). If this occurs, the algorithm starts looking for a contract with a value of the previous offered contract $V_b(x)$ reduced by a predetermined amount η. The average closeness is measured by the similarity between the last two bids of the agent. This leads to the following bidding strategy for agent a:

```
if (|sim_b(x_{t-1}) − sim_b(x_t)| < δ)
    choose (x_{t+1} : x_{t+1} ∈ X ∧ u_b(x_{t+1}) = u_b(x_t) − η),
else
    choose (x_{t+1} : x_{t+1} ∈ X ∧ u_b(x_{t+1}) = u_b(x_t)).
```

where η is the decrease in utility when a deadlock occurs (also set to 0.05 in this experiment). Note that this method always uses the trade-off strategy, and therefore makes extensive use of the prediction of the opponent's weights by the KDE-method.

In the experiment, both agents use the above trade-off strategy, the seller has full knowledge about the opponent, and the amount of knowledge the buyer has about the seller is varied (in a similar way to section 4.1).

The performance of the strategy is tested in two ways. First, the utility of the deal is calculated by taking the product of the utilities of the agents, where a higher product means a better performance. By looking at the product, we make sure that the deal is symmetric: there is not one agent performing better at the cost of the other. Second, we will look at the time at which agreement is reached. In the domain of e-commerce, it is often desirable to reach an agreement as soon as possible (e.g. to reduce the communication load or because the service is required urgently). We will define this by the number of offers and counter-offers before an agreement is reached[7].

In undertaking this experiment, we expect that more efficient trade-off offers will lead to a more efficient eventual negotiation outcome. In the single-offer experiments we showed that by having more knowledge, the performance increased and we expect this to also be the case for the multi-offer situation. Furthermore, since the single offers are more efficient, we expect an agreement to be reached earlier:

[7]In all cases, an agreement was reached before one of the agents reached its deadline, so the communication load is defined in all cases.

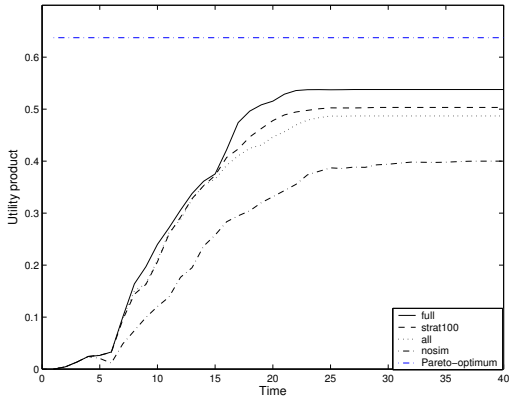


Figure 5: Utility of the contract proposed at given time

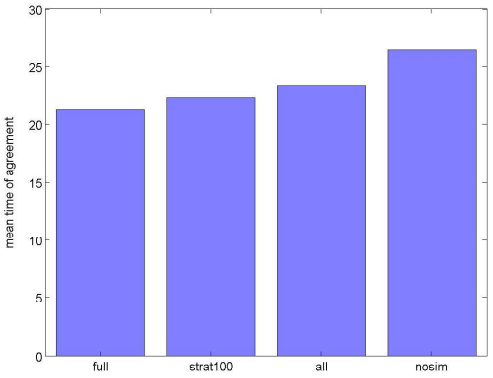


Figure 6: Mean time of agreement

HYPOTHESIS 6. *The more knowledge that is used in the KDE, the higher the product of the utilities will be over a complete negotiation.*

HYPOTHESIS 7. *The more knowledge is used in the KDE, the lower the communication load will be.*

First, we look at the utility. In figure 5, the Pareto-optimal solution which maximizes the product of utilities is indicated by the dash-dotted line. As expected, the strategy performs best when we assume full knowledge of the opponent, and worst when we do not use any knowledge at all (the *no sim* scenario). The performance of *strat100* is not significantly better than the performance of *all*, implying that the quality of the prediction of the KDE is comparable in both situations for the trade-off algorithm. As expected, the scenario in which the buyer has full knowledge about its opponent performs significantly better than the scenarios in which a KDE is used. Most important, however, the performance of the KDE-method is far better than that of not using any knowledge at all. We therefore accept hypothesis 6.

Second, we look at the mean time on which agreement is reached in figure 6. This shows, when the knowledge of the opponent is complete, agreement is reached fastest (mean time of 21.2 steps). *Strat100* does not perform significantly worse, indicating that the prediction of the KDE-method is precise enough for the trade-off algorithm. Slightly worse is

all which needs significantly more steps to achieve an agreement. These results can be explained by the lower performance of the single offers (as described in section 4.1). Specifically, these individual offers have a lower utility to the opponent, therefore they are less likely to be accepted. Not using the similarity criteria performs, as expected, worst. Based on this results, we also accept hypothesis 7.

5. CONCLUSIONS AND FUTURE WORK

In this paper we showed that the preferences of a negotiation opponent in bilateral multi-issue negotiations can be effectively learnt by using *kernel density estimation*. Specifically, by applying it to Faratin's trade-off model, we showed that it can make negotiations more efficient (in terms of utility as well as time of agreement). By choosing kernel density estimation as the learning paradigm, we did not have to make any explicit assumptions about the relation between time, negotiation history and the opponent's preferences (as many other learning methods have to). Also, the method has reasonable computational complexity for the bounded nature of e-commerce domains.

In more detail, our experiments showed that applying kernel density estimation to a single-offer case improved the performance of the trade-off algorithm. The resulting algorithm is robust when the prediction of the weights is imprecise and is not dependent on the ordinality of the weights being kept. Moreover, the amount of knowledge used in the KDEs does not have a major influence on the performance, which means it can work effectively in competitive environments in which minimal information is made available. Furthermore, we showed that using the KDE-method outperformed the uniform weight strategy for the opponent when it has a skewed weight vector and performed at least as well when it uses a near uniform weight vector. Finally, we showed that using the KDE-method in a complete negotiation encounter which uses the predictions extensively also leads to a better performance. Specifically, the more knowledge is used in the KDEs, the higher the utility and the lower the communication load.

For the future, there are two main ways in which this research can be extended. Firstly, we would like to consider the performance of our method against additional meta-strategies. In this work, we only consider an opponent that uses the *smart* meta-strategy and other meta-strategies may also be adopted in practice. Secondly, our method for predicting the relative weight of the opponent's preferences for its various negotiation issues could be applied to a variety of other negotiation models where it is important to have approximations of these values. Thus, for example, it could be used in purely competitive encounters in incomplete information settings or situations in which an agent engages in multiple concurrent negotiations in order to procure a particular service. In both cases, the better the approximation, the better the deal the agent will be able to make.

6. ACKNOWLEDGMENTS

We would like to acknowledge Dr. Steve Gunn for bringing the kernel density estimation method to our attention.

7. REFERENCES

[1] H. Bui, S. Venkatesh, and D. Kieronska. An architecture for negotiating agents that learn. Technical report, Department of Computer Science, Curtin University of Technology, Perth, Australia, July 1995.

[2] P. R. Cohen, editor. *Empirical Methods for Artificial Intelligence*. The MIT Press, 1995.

[3] P. Faratin. *Automated Service Negotiation Between Autonomous Computational Agents*. PhD thesis, University of London, Department of Electronic Engineer Queen Mary & Westfield College, December 2000.

[4] P. Faratin, C. Sierra, and N. R. Jennings. Using similarity criteria to make issue tradeoffs in automated negotiations. *Artificial Intelligence*, 142(2):205–237, 2002.

[5] S. Fatima, M. Wooldridge, and N. R. Jennings. Optimal agendas for multi-issue negotiation. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 129–136. ACM Press, 2003.

[6] S. S. Fatima, M. Wooldridge, and N. R. Jennings. Multi-issue negotiation under time constraints. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-02)*, pages 143–150, Bologna, Italy, July 2002.

[7] E. Gerding and D. van Bragt. Multi-issue negotiation processes by evolutionary simulation, validation and social extensions. *Computational Economics*, 22(1):39–63, August 2003.

[8] M. He, N. R. Jennings, and H.-F. Leung. On agent-mediated electronic commerce. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):985–1003, july/august 2003.

[9] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: prospects, methods and challenges. *Int. J. of Group Decision and Negotiation*, 10(2):199–215, 2001.

[10] J. R. Oliver. A machine-learning approach to automated negotiation and prospects for electronic commerce. *Journal of Management Information Systems*, 13(3):83–112, 1997.

[11] H. Raiffa. *The Art and Science of Negotiation*. Harvard University press, Cambridgem USA, 1982.

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

[13] V.-W. Soo and C.-A. Hung. On-line incremental learning in bilateral multi-issue negotiation. *AAMAS'02*, 2002.

[14] K. Sycara. Multi-agent compromise via negotiation. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence (Vol. 2)*. Morgan Kaufmann, Los Altos, CA, September 1989.

[15] M. Wand and M. Jones. *Kernel Smoothing*. Chapman & Hall, London, 1995.

[16] D. Zeng and K. Sycara. Bayesian learning in negotiation. *International Journal Human–Computer Studies*, 48:125–141, 1998.