Avoiding Approximation Errors in Multi-Issue Negotiation with Issue Dependencies

Koen Hindriks
Man-Machine Interaction Group
Delft University of Technology
Mekelweg 4, Delft, The Netherlands
+31.15.2781315
k.v.hindriks@tudelft.nl

Catholijn Jonker
Man-Machine Interaction Group
Delft University of Technology
Mekelweg 4, Delft, The Netherlands
+31.24.2782523
c.m.jonker@tudelft.nl

Dmytro Tykhonov
Man-Machine Interaction Group
Delft University of Technology
Mekelweg 4, Delft, The Netherlands
Telephone number, incl. country code
d.tykhonov@tudelft.nl

ABSTRACT

Searching for good bids in a utility space based on multiple, dependent issues in general is intractable. Tractable algorithms do exist for independent issue sets, so one idea is to eliminate the dependencies by approximating the more complex utility space with issue dependencies. It has been shown that an approximation may give reasonable results when some structural features of the negotiation domain and preference profile are exploited. Of course, there is a risk that approximation results in significantly different negotiation outcomes. In this paper, we present a checking procedure to mitigate this risk and show that by tuning the parameters of this procedure the outcome deviation can be controlled. These parameters allow for a trade-off between computational cost and accuracy of negotiation outcome. Based on experimental results we propose specific values for the parameters of the checking procedure that provide a good balance between computational costs and accuracy. Additionally, we show how different values of these parameters influence the computational costs of negotiating multiple issues with dependencies.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence **–** *intelligent agents, multi-agent systems.*

General Terms

Algorithms, Performance, Economics, Experimentation, Theory.

Keywords

Efficient multi-issue negotiation, issue dependencies, tunable algorithm, approximating utility spaces.

1. INTRODUCTION

Negotiation is a process by which a joint decision is made by two or more parties (cf. [10]). The parties or agents first express conflicting demands and then move towards agreement by a process of concession making. During the negotiation both agents make various offers, called *bids*, to each other that more or less match with their own preferences. The negotiation outcome is either a failure, or a deal, i.e., a bid accepted by all parties. If multiple issues are at stake then these issues may need to be negotiated simultaneously and the bids made may vary on each of these issues.

One of the complicating factors in a computational approach to negotiation is that the value associated with a bundle of multiple issues may not be a simple function of the value associated with individual issues. In [10], Raiffa explains how to mathematically model a preference profile of an agent that can be used during the negotiation to determine the utility of exchanged bids. The representation of an agent's preferences by mathematical functions, called *utility functions*, which map values of issues to the utility of bids, i.e. bundles of issue values, allows the development of software support for negotiations. In negotiation domains with issue dependencies which influence the overall utility of a bid, however, the utility space is non-linear in the issues (cf. [1]). In [7], Klein et al. show that in that case there is no efficient method to compute alternative bids during a negotiation, even if the agent tries to guess the opponent's profile.

Some proposals have been made to reduce the computational complexity of multi-issue negotiation with issue dependencies. For example, [7] propose the use of a mediator which may be more computationally efficient when both agents in a negotiation reveal their preferences to this mediator. An alternative, interesting option is to investigate the complexity of the utility space itself and try to eliminate the dependencies between issues. In [5], an approximation method is proposed to eliminate issue dependencies, see Figure 1. This method exploits some structural features of preference profiles of agents to approximate the original profile. The resulting approximated utility function without dependencies can be handled by negotiation algorithms that can efficiently deal with independent multiple issues and have a polynomial time complexity (see e.g. [6]).

It is clear that the method proposed in [5] removes the computational intractability of multi-issue negotiation with issue dependencies by transforming the original profile input to one that can be linearly decomposed. Tested over numerous random spaces of interdependent issues, the negotiation outcome using this approximation is reasonably good, see [5]. The negotiation outcome, however, does not only depend on the preference profile but also on the process of negotiation itself. It is to be expected that the risk of obtaining a bad outcome due to the use of an approximation cannot be avoided completely even if the approximation is quite good.

Conference'04, Month 1–2, 2004, City, State, Country.

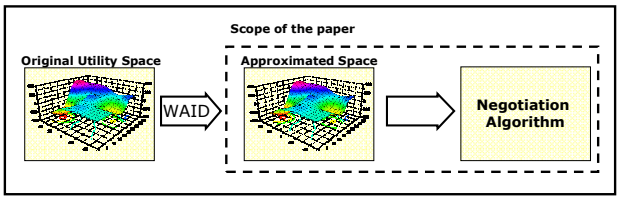

Figure 1. Multi-issue negotiation with issue dependencies using approximated utility spaces

In this paper, we analyze the risk of a bad negotiation outcome when using an approximation of the agent's preference profile. It turns out that in some domains this risk may still be unreasonably high. The results show that using the approximated space a bid might be proposed that in the original utility space would have a too low utility. The risk of such an erroneous bid can be quite high and, as a consequence, the risk of obtaining a bad negotiation outcome is significant. In order to control this risk, we therefore also have to look at the process of negotiation. More precisely, we investigate a way to incorporate a method to control the risk of an erroneous bid in the negotiation algorithm itself. This paper presents a checking procedure to control the risk of erroneous bids which can be incorporated in any negotiation algorithm.

Of course, this checking procedure introduces some additional computational costs. A procedure that completely eliminates the risk of erroneous bids, moreover, would make the negotiation process intractable again. One of the main contributions of this paper is that it shows that a trade-off can be made between computational efficiency and approximation accuracy, which is directly related to the negotiation outcome. The parameters of the checking procedure allow the tuning of a negotiation algorithm to increase either the computational efficiency or decrease the risk of erroneous bids. Derived from experimental results, we propose specific values for these parameters that ensure a reasonable balance between computational costs and outcome deviation (in terms of utility) in many domains. Finally, we present experimental results that show that the approach of adding a checking procedure to the negotiation algorithm is scalable and allows an agent to negotiate about high-dimensional utility spaces.

The paper is organized as follows. First, some of the basic notions to model preference profiles that characterize our approach to multi-issue negotiation are introduced. In section 3 a brief overview of the approximation method for eliminating issue dependencies is presented. Then in section 4 the outcome deviation that results from using an approximated space as input for a negotiation algorithm is analyzed and the need for an additional method to prevent erroneous bids is argued for. In section 5, the negotiation algorithm is adapted by incorporating a checking procedure. Successively, the performance of this adapted negotiation algorithm is investigated. Experimental results are presented that confirm that a significant improvement can be obtained by incorporating the checking procedure. The impact of the checking procedure is analyzed in Section 6. The impact of on the computational tractability of the negotiation algorithm is investigated in Section 7. Section 6 shows that by varying certain parameters of the method a trade-off can be made between outcome deviation, caused by erroneous bids, and computational costs. Furthermore, specific values for these parameters of the checking procedure are proposed to obtain a good balance. Finally, section 9 concludes the paper.

2. MODELING ISSUE DEPENDENCIES

The overall utility of a set of *independent* issues can be computed as a weighted sum of the values of each of the issues by associating an evaluation function with each issue variable (see e.g. [6, 10]). The properties of the utility function are derived from these evaluation functions which map issue values on a closed interval [0; 1]. This model, represented in equation (1), can be used for issue values that are numeric (e.g. price, time) as well as for issue values that are discrete (e.g. colors, brands).

$$u(x_1,..., x_n) = \sum_{i=1}^{n} w_i ev_i(x_i) \qquad (1)$$

Equation (1) cannot be used, however, for modeling dependencies between issues and equation (1) needs to be generalized to equation (2) (cf. also [1]). Of course, the value of an issue does not need to depend on all other issues and subsets of dependent issues will have to be considered to model individual examples.

$$u(x_1,..., x_n) = \sum_{i=1}^{n} w_i ev_i(x_1,..., x_n) \qquad (2)$$

The representation of a utility space with non-linear issue dependencies as in equation (2) is similar to the model proposed in [7]. The main difference is that instead of considering only binary issue values, we allow multi-valued, discrete, as well as continuous issue ranges.

The complexity of a utility function determines the computational complexity of the negotiation process. One of the main problems in dependent multi-issue negotiation is the computational complexity associated with searching for appropriate bids in the corresponding utility spaces. In case a utility function of multiple issues is non-linear in these issues, i.e. there are issue dependencies, finding a particular bid in the utility space is intractable.

3. APPROXIMATING UTILITY SPACES

To experimentally determine the need for, and later, to assess the effectiveness of including our checking procedure in negotiations in which approximated utility spaces are used, we need a functioning approximation method and an implemented negotiation strategy.

This section provides a brief overview of the WAID-approximation technique of [5]. The WAID-method transforms a utility space with issue dependencies into a space without such dependencies to meet the input requirements of efficient multi-issue negotiation algorithms, see Figure 1. The WAID-method is explained only to the detail necessary to understand the problem of approximations and in order to understand that using a checker in the negotiation algorithm would diminish the risk of erroneous bids. More details can be found in [5].

The main idea of the WAID-method is that structural features of the negotiation domain and utility functions with issue dependencies can be exploited to approximate a preference profile and eliminate issue dependencies. It also seems that humans tend to simplify the structure of their preferences and prefer to negotiate one issue at a time [13].

Formally, the objective of the WAID-method is to transform a utility space $u(x_1,\ldots,x_n)$ based on dependent issues as represented by equation (2) to a utility space $u'(x_1,\ldots,x_n)$ without such dependencies that can be represented by equation (1). The

transformation consists of approximating each of the evaluation functions $evi(x_1,\ldots,x_n)$ by a function $ev'_i(x_i)$ in which the influence of the values of other issues x_j, $j \neq i$, on the associated value $ev_i(x_1,\ldots,x_n)$ have been eliminated.

The heart of the WAID-method is a weighted averaging technique. The dedication of WAID to utility spaces for negotiation shows in it's exploitation of some general and, if available, additional domain specific insight into negotiation. These insights concern the relative importance of bids and what utility can reasonably be expected of an outcome of a negotiation.

The WAID-method consists of 4 steps. The first step is to estimate an expected outcome utility, called *m-point*. The m-point later serves as a focus point for the approximation. Secondly, an evaluation of the type of approximation that best fits the case at hand is made. This second step is not elaborated here. Third, the actual approximation is computed. In the last step the difference of the original and approximated utility space is determined. Depending on this analysis, negotiators can decide to use the approximation or not in their negotiation algorithm.

Estimate an Expected Outcome Utility

In the first step, the expected utility of the outcome is estimated. This estimate is called the *m-point* and is used to define a region in the utility space where the actual outcome is expected to be. The m-point is used to feed information about the final goal of negotiation, i.e., the utility of the outcome, into the approximation technique used to transform the utility space.

For multi-issue negotiation in general we may assume that the expected outcome of the negotiation is located somewhere in the open utility interval (0.5; 1), say 0.75. Lower than 0.5 would not be accepted by the agent, and 1 is the maximal utility. The approximation should be most accurate in that interval, and especially around the m-point, because those points are most important for getting a good negotiation outcome. An experienced agent or one with additional knowledge about the domain can narrow the interval of the m-point.

Choice of Weighting Function

The next step is to define a weighting function ψ. An agent may be more or less uncertain, about its estimate of the m-point and therefore, also of the corresponding interval. The weighting function ψ is chosen such that the approximation is most accurate in the region(s) of the utility space corresponding to that interval.

Computing the Approximation

The third step is to calculate an approximation of the original utility space based on non-linear issue dependencies using the m-point and the weighting function as defined in the previous steps. The result of this step is a utility space that can be defined as a weighted sum of evaluations of independent issues, i.e., of the form of equation (1). The WAID-method multiplies each evaluation value with its corresponding weight and then averages the resulting space by integration. Additionally the weighting is normalized over the interval of integration, see equation (3). V denotes the range of integration and is a volume of n-1 dimensionality build from the issue dimensions $\{x_1,x_2,\ldots,x_{i-1},x_{i+1},\ldots,x_n\}$. Of course, not all issues have to depend on all others and some issue variables may be dropped from the equation in that case. The approximation technique can be applied sequentially for each issue variable which involves dependencies between issues.

$$ev'_i(x_i) = \frac{\int_V \psi(x_1, x_2, .., x_n) ev_i(x_1, .., x_n) dV}{\int_V \psi(x_1, ..., x_n) dV} \qquad (3)$$

The WIAD method allows arbitrary utility spaces with issue dependencies to be approximated by weighted approximation. The approximation of the utility space modeling an agent's may be more or less differ from the original utility space, but always will introduce errors in the associated utility of a bid. As a result, in a negotiation, bids selected by using the approximated space may deviate from the actual utility in the original space. Such bids in turn may become a final deal. If the deviation is quite big, however, the outcome may not be acceptable in the end to the agent. But how big is that risk? To understand the risks, experimentation is necessary. To be able to experiment, comparable negotiations must be performed with the original and with the approximated utility space. To be comparable, a particular negotiation strategy must be applied in both cases. The next section discusses such a strategy and presents the results of its application in a thorough experiment to understand the risks of using an approximated space. Since the risk is real, we then investigate in Section 5 and after whether a controlled trade-off between computational complexity and accuracy of the negotiation strategy can be incorporated into a negotiation algorithm.

Negotiation Algorithm Used by Agent B

Initialization: set initial utility to maximum of U'_B.

1. **Evaluate bid $bid_A(i)$ received from opponent A:**
   Accept and end negotiation if $U_B(bid_A(i)) > U_B(bid_B(i))$
2. **Compute concession and target utility:**
   Concession $\gamma = \beta*(1-\mu/\ U_B(bid_B(i)))*(\ U_B(bid_A(i))-U_B(bid_B(i)))$
   Target Utility $\tau = U_B(bid_B(i))+\gamma$
3. **Determine a next bid:**
   Find a bid $bid_B(i+1)$ such that $U'_B(bid_B(i+1)) = \tau$
4. **Send bid to opponent.**

Table 2. ABMP negotiation algorithm for approximations

4. NEGOTIATION ALGORITHM

The negotiation algorithm that is used plays a key role in obtaining a good negotiation outcome. The approximation of a preference profile allows an agent to more efficiently compute good bids during negotiation, but does not in itself provide a guarantee that against arbitrary opponents a good negotiation outcome will be reached. More insight is required to assess the effects of using approximations of real preference profiles.

As a first step, therefore, we analyze the effects of using approximated utility spaces as substitutes for the original spaces. To perform such an analysis, we use a negotiation algorithm that corresponds to the ABMP-strategy of [6], but other strategies could have been used as well. The algorithm is outlined in Table 1. It is assumed that negotiation proceeds between agents A and B. In Table 1, the perspective of agent B, that uses the

approximated space $U'_{B,}$ is provided. The original space of agent B is indicated by U_B.

The negotiation strategy can be outlined in the following way. In step 1, any previous bid of the opponent A is evaluated and accepted if it exceeds the bid of agent B in the last round. If an agreement cannot yet be reached, the ABMP-strategy determines a next bid to offer in two steps: the strategy first (step 2) determines the target utility for the next bid based on a computation of a concession step, and then (step 3) determines a bid that has that target utility. Step 3 of the strategy is very efficient for utility spaces without issue dependencies. It is in this step that the approximated utility space U'_B has to be used. Note that the approximated utility space U' is only used in the initialization and in step 3 since its purpose is to speed up the negotiation. In the other steps the computations that involve the original space are computationally cheap. In step 4, finally, the computed bid is sent to the opponent for evaluation.

The ABMP negotiation algorithm is used to assess the outcome deviation that may occur when an approximated space is used instead of the original space during a negotiation. In the experiments that were performed agent A also uses a variant of the ABMP strategy but does not approximate any issue dependencies in its utility space. Instead it uses exhaustive search through its utility space in step 3 to determine a next bid given a suitable discretization of this space (i.e. using small enough steps).To compare outcomes for utility spaces of medium size, the same negotiation is performed again with agent B using exhaustive search in step 3. Of course, exhaustive search can only be used for utility spaces of medium size due to exponential time costs and memory limitations. It is, however, imperative to use it if we want to calculate outcome deviation. In the experiments, spaces with up to a number of 5 issues and a number of discretization steps of at most 25 have been used (see also Section 6 and 7). Agent A always begins the negotiation by proposing an initial bid.

To analyze the impact of the weighted averaging method on the negotiation outcome a probabilistic experimental setup has been used. The negotiation outcomes obtained by using the approximation method are compared with those obtained using the original utility space. The experimental results are obtained from utility spaces modeled by multivariate quadratic polynomials. These polynomials may have multiplicative terms $x_i x_j$ which represent issues. It is well-known that solving such quadratic programming problems is NP-hard, see e.g. [3]. In the experiments utility spaces have been randomly generated. The *m-point* parameter that has to be fixed in order to apply the WAID-method is determined for each utility space by a Monte-Carlo method.

The main result of the experiments performed shows that the distribution of negotiation outcome deviations is similar to a normal distribution with a mean value close to zero. Figure 3 presents the distribution of outcome deviations for a negotiation about 4 issues. The deviation is a result of using the approximated space in the negotiation strategy instead of performing an exhaustive search to find a good bid in the original space. As can be seen in figure 3, the bell-shaped distribution (average = -0.02; standard deviation = 0.09) means that the negotiation over the approximated space tends to produce the same result as the negotiation over the original space using exhaustive search. This demonstrates that one may expect to obtain reasonable outcomes when negotiating with approximated spaces instead of non-approximated spaces.

Even though this result shows that approximating the original utility space to remove issue dependencies may result in quite reasonable outcomes compared to those obtained otherwise, it also shows that there is quite a high chance of deviating significantly. In fact, for the 4 issue case figure 3 shows that there is a quite high probability of obtaining outcomes that are worse by up to 33%. Additionally, the curve is not really symmetrical and shows a tendency towards negative deviations. As an illustration, the probability of obtaining a result that is worse than 10% equals 0.196. It is clear that in many domains such a high risk will be unacceptable.

The main conclusion thus is that additional measures need to be taken to reduce this risk. The benefit of using approximated spaces is clear: issues can be negotiated independently which makes the negotiation tractable. But a balance has to be found between the computational costs and the risk of significantly deviating negotiation outcomes. Ideally, we would like to be able to make a tradeoff between costs and outcome deviation to obtain the right balance and control the risk of bad outcomes.

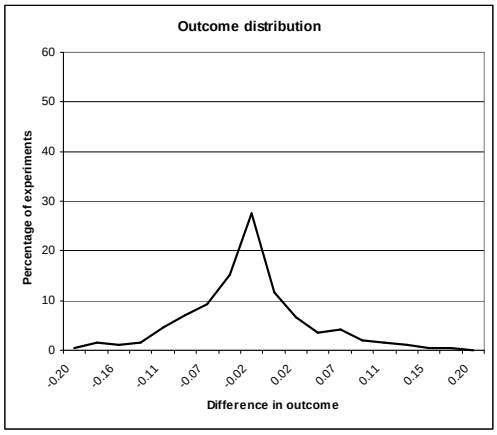

Figure 3 – Distribution of negotiation outcome deviation for approximated spaces vs. original spaces for 4 issues (k=15)

5. PROCEDURE FOR CONTROLLING NEGOTIATION OUTCOME DEVIATION

In this section we propose a parameterized procedure that can be used to control the probability of large outcome deviations. The parameters of this procedure can moreover be used to influence the tradeoff between the accuracy of the negotiation outcome and the computational efficiency of the negotiation strategy. In the next sections, experimental results are presented that allow the tuning of these parameters.

In the negotiation algorithm the bid selection procedure is the source of the deviation of the negotiation outcome. In particular, in step 3 of the algorithm in Table 2 the approximated space is used instead of the original space which gives rise to outcome deviations. To avoid approximation errors that are too big, we propose to add a checking procedure in this step which compares

the utility of a bid in the approximated space with the utility in the original space.

The absolute error as a result of the approximation can be computed simply by subtracting the utility in the approximated space from the utility in the original space as in equation (4).

$$\Delta(x_1, x_2, ..., x_n) = |u(x_1, x_2, ..., x_n) - u'(x_1, x_2, ..., x_n)| \quad (4)$$

This equation gives the error associated with the utility of any bid that is proposed during the negotiation. In order to get as close as possible to a negotiation outcome that would result if the original space would have been used, one approach is to minimize this error for each bid that is offered to the opponent during the negotiation.

The proposed procedure can be found in Table 4. The step to determine a next bid is refined and an iterative procedure is incorporated to check whether the difference in utility stays below a certain threshold δ. As before, in step 3a a bid is computed that matches a certain target utility. In step 3b, however, now a check has been incorporated that checks whether $\Delta(bid) < \delta$, that is, whether the absolute approximation error stays below a threshold δ. This additional check itself is computationally cheap, since it involves only a simple calculation using equation (4). If $\Delta(bid) > \delta$, a bid bid', which utility differs minimally from the previously computed bid, is searched for, until $\Delta(bid') < \delta$. This iterative procedure for finding an appropriate bid is called *δ-checking*.

The additional check is used to avoid the risk of proposing bids with (very) low utilities in the original space that have (much) higher utilities in the approximated space. The concessions made in step 3 thus are controlled by a parameter δ to ensure that they are not too big.

Negotiation Algorithm Used by Agent B

Initialization: set initial utility to maximum of U'_B.

1	**Evaluate bid $bid_A(i)$ received from opponent A:** Accept and end negotiation if $U_B(bid_A(i)) > U_B(bid_B(i))$
2	**Compute concession and target utility:** Concession $\gamma = \beta*(1-\mu/\ U_B(bid_B(i)))*(\ U_B(bid_A(i))-U_B(bid_B(i)))$ Target Utility $\tau = U_B(bid_B(i))+\gamma$
3	**Determine a next bid:**
3a	**Find a bid with target utility** Find a bid $bid_B(i+1)$ such that $U'_B(bid_B(i+1)) \approx \tau$
3b	**Compare bid utility in approximated and original space** Check whether $\lvert U_B(bid_B(i+1)) - U'_B(bid_B(i+1))\rvert < \delta$
3c	**If not, find next candidate for the bid and repeat step (3b):** Find next candidate bid $bid_B(i+1)$ such that $U'(bid_B(i+1)) \approx \tau$ and utility with previous bid only differs minimally.
4	**Else, send bid to opponent.**

Table 4. Negotiation algorithm with δ-checking procedure

A few remarks about implementing step 3c are in order. Currently, we use a simple approach and a discretization of the approximated evaluation functions is used. Using these discretized evaluation functions, a distance between the target evaluation value and each issue value can be calculated as follows:

$$d_i(ev(x_i), tev(x_i)) = |ev(x_i) - tev(x_i)| \quad (5)$$

The impact of adding the δ-checking procedure to the negotiation algorithm on the outcome distribution is significant, as is shown by figure 5. The experimental setup is exactly the same as that for figure 3 but the negotiation algorithm used by agent B now includes the checking procedure. It shows the outcome distribution for a threshold of δ=0.01.

Clearly, the outcome distribution curve in figure 5 is more symmetrical than in figure 3 and more clustered around the mean; it has a mean=-0.00016 and a standard deviation of 0.045. A more detailed analysis of the relation between δ and the outcome deviation is presented in the next section.

The δ-checking procedure introduces additional search again into the computation of a bid. Various heuristics could be applied again, however, to minimize the amount of search. For example, a limit on the number of iterations could be introduced for spaces of high dimensionality to ensure a bid would be found within a reasonable amount of time. (The probability of finding an appropriate bid is high in high-dimensional spaces close to the m-point.) The relation of the value of the δ-parameter and the computational cost is analyzed in more detail using experimental results in Section 7.

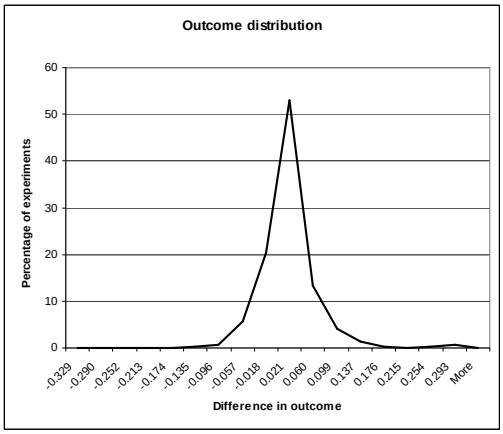

Figure 5 – Outcome distribution with checking procedure for approximated spaces vs. original spaces for 4 issues (k=15)

6. IMPACT ON OUTCOME DEVIATION

In this section, we present experimental results that show how the value of the δ-parameter in the checking procedure relates to the distribution of the outcome deviation. These results show that there is a direct relation between the size of δ and outcome distribution.

Additionally, we investigated the influence of the discretization per issue under consideration on the outcome distribution. In the experiments we performed, the possible values for each issue were reduced by discretizing the space to 10, 15, 20, and 25 values. In the results below, the discretization parameter is indicated by *k*. Maybe somewhat surprisingly the different values for *k* used in the experiments do not have such a big impact on the outcome distribution.

In order to assess the impact of adding the checking procedure to the negotiation algorithm, we performed experiments with 3, 4, 5, and 6 issues. Finally, for the δ-parameter of the checking procedure we used the values 0.001, 0.005, 0.01, 0.02, 0.03, and 0.05. In total, we performed over 44.000 experiments in which the

outcomes were compared with the original space: 12.000 for 3 issues, 12.000 for 4 issues, 12.000 for 5 issues, and 6.000 for 6 issues. Comparisons of negotiation outcome for spaces of higher dimensionality were not feasible.

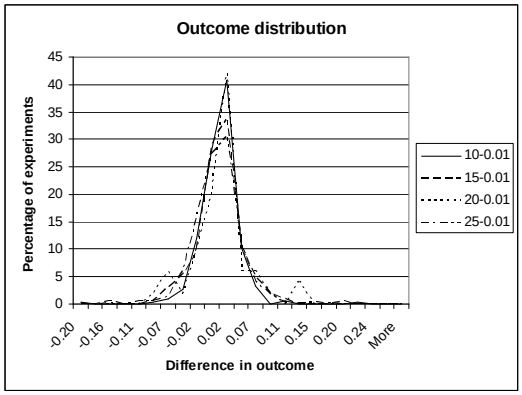

Figure 6. The distribution of outcome deviations for 5 issues and δ = 0.01. The various lines relate to different *k*-values.

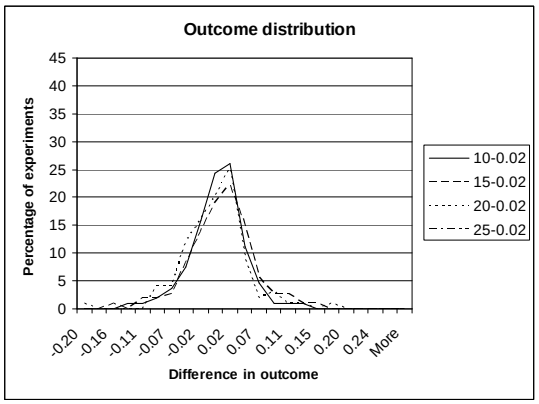

Figure 7. The distribution of outcome deviations for 5 issues and δ = 0.02. The various lines relate to different *k*-values.

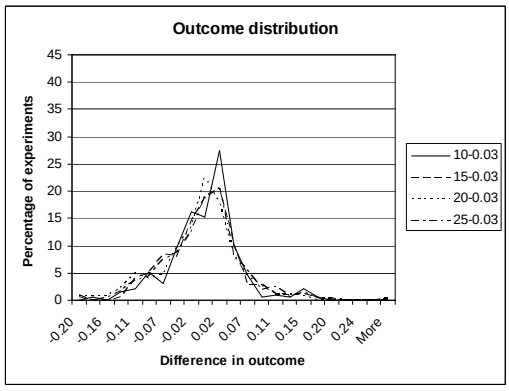

Figure 8. The distribution of outcome deviations for 5 issues and δ = 0.03. The various lines refer to different *k*-values.

The higher the number of issues *n* and the higher the discretization parameter *k*, the longer it takes to do the exhaustive search (it takes k^n steps). Also, for spaces with more that 5 issues and high discretization factors, the memory requirements become unmanageable. As a result, the number of experiments with 6 issues was lower than those with 3, 4 and 5 issues. To investigate the scalability of the proposed approach, we ran 500 experiments with 10 issues for δ=0.02 and each k-value, so 2000 experiments in total. The results for 10 and 50 issues are presented in Section 8.

The experimental results relating the value of δ to the outcome distribution are depicted in Figures 6 to 8. We do not show all results but only those for δ-values of 0.01, 0.02, and 0.03 which most clearly demonstrate the impact of different values on the distribution and also define the turning points where decreasing this parameter further does not have a very big impact anymore (see also Figure 12) and decreasing it results in significantly worse outcomes. In Figures 6 to 8, on the x-axis the outcome difference is set out. The outcome deviation may be bigger than the value of the δ-parameter since errors may accumulate over multiple rounds in the negotiation. The y-axis refers to the percentage of experiments having particular outcome differences. The different lines correspond with different values of the discretization parameter *k*. For each combination of a particular number of issues, δ-value, and k-value, 500 experiments were run.

In general, as is to be expected since δ is supposed to control the error introduced by the approximation, the experimental findings show that smaller values for δ result in negotiation outcomes that are closer to the outcomes in the original space. A positive value with respect to difference in outcome means that the negotiation outcome was improved compared to the outcome obtained when using the original space.

The findings illustrated in Figures 6 to 8 are as follows. For δ = 0.01 (see Figure 6) the standard deviation ranges from 0,0327 to 0,0442, and the average outcome difference ranges from -0,0066 to 0,0015. For δ = 0.02 (see Figure 7) the standard deviation ranges from 0.0350 to 0.05806 and the average outcome difference ranges from -0.0142 to 0.0010. Finally, for δ = 0.03 (see Figure 8) the standard deviation ranges from 0,0499 to 0,0717, and the average outcome difference ranges from -0,0199 to -0,0151.

7. IMPACT ON COMPUTATIONAL COST

Including the checking procedure implies that the bid determination part might need iterations to find an appropriate bid. The previous section shows that smaller δ–values lead to better outcome deviations, and it stands to reason that the smaller the value, the higher the number of iterations needed. To get more insights into the frequency with which the need for iterations causes high computational costs, a series of experiments have been performed. The algorithm was tested for 4, 5, 6, and 10 issues, with the discretization value k varying over {10, 15, 20, 25} and δ varying over {0.005, 0.001, 0.03, 0.02, 0.01}. Each test was performed 500 times with randomly generated original utility spaces.

Figures 9, 10, and 11 show the results for 5 issues, the results for other values are not shown, since they do not provide additional insights. In these pictures, on the x-axis the logarithmic costs are set out. The y-axis refers to the frequency with which an experiment had such a logarithmic cost, with respect to the total number of experiments. The different lines refer to different k-

values. In Figure 13, Section 8 the same analysis is presented for 10 issues with $\delta = 0.02$.

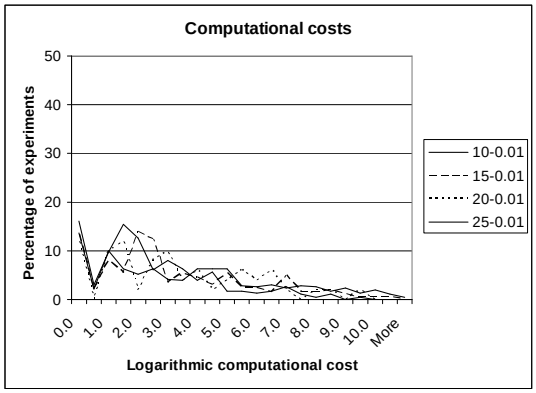

Figure 9. Computational costs for 5 issues and $\delta = 0.01$. The different lines refer to different k-values.

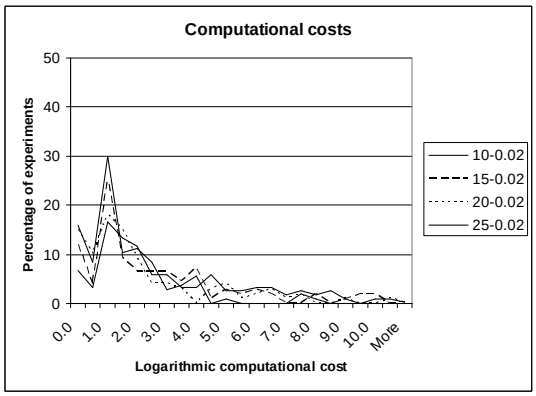

Figure 10. Computational costs for 5 issues and $\delta = 0.02$. The different lines refer to different k-values.

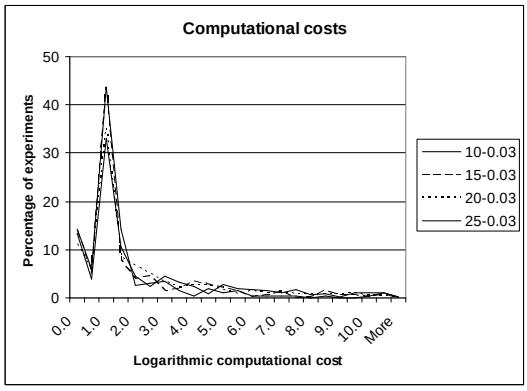

Figure 11. Computational costs for 5 issues and $\delta = 0.03$. The different lines refer to different k-values.

The results clearly show the expected increase of high computational costs for higher δ–values: higher percentages for higher computational values. However, when looking at the areas underneath the lines, another interesting observation can be made. In Figure 9, for $\delta = 0.01$, the bulk of the area underneath the lines ends approximately at $\ln(x) = 6$. In Figure 10, for $\delta = 0.02$ the bulk ends at $\ln(x) = 4$, and in Figure 11, for $\delta = 0.03$ at $\ln(x) = 2$. Evidently, the number of iterations needed is bounded.

8. TRADE OFF

Combining the results of the outcome analysis of Section 6 and the computational cost analysis of Section 7 shows that the need for a small outcome difference has to be balanced against computational costs. In this a setting for the k, and δ parameters is chosen that balances accuracy against efficiency. The approach with these parameter settings is shown to be still efficient for a large numbers of issues.

Sections 6 and 7 show that accuracy and computational cost increase as δ decreases. To find a good balance between accuracy and cost, an integrated analysis has been performed for the usual combination of parameters: the number of issues ranging over {4, 5, 6, 10}, k ranging over {10, 15, 20, 25} and δ ranging over {0.001, 0.005, 0.01, 0.02, 0.03, 0.05, 1}. Note that $\delta=1$ corresponds to a setting in all checks are successful and, therefore, no iterations are necessary.

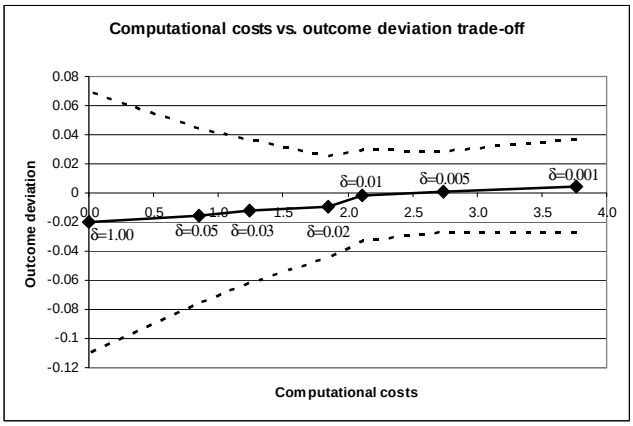

Figure 12. Computational cost and outcome deviation for 5 issues and k=10

Figure 12 presents the trade-off between negotiation outcome accuracy and the computational costs. Each point on the solid line of the chart represents the average of a series of experiments where δ varies over {0.001, 0.005, 0.01, 0.02, 0.03, 0.05, 1}. The dashed lines represent the spread of the negotiation outcome deviation. The top line is an average + standard deviation and bottom line is the average - standard deviation.

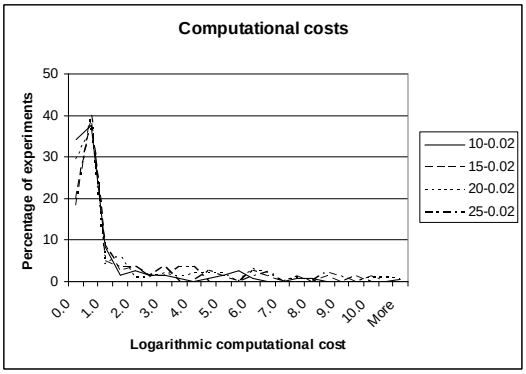

Figure 13. Computational costs for 10 issues and $\delta = 0.02$. The different lines refer to different k-values

The results show that a good compromise is a δ–value of 0.02: for $\delta < 0.02$ the costs increase, for $\delta > 0.02$ the outcome approximation gets worse. Furthermore, the standard deviation drops off at this value, but does not decrease further for even smaller δ-values.

To analyse the scalability of the modified negotiation algorithm we performed a series of negotiations with 10-issues. Unfortunately, it is no longer possible to use exhaustive search as a benchmark for the negotiation outcome efficiency due to the extremely large utility space (11^{10} to 26^{10}). Figure 13 shows computational cost for 10-issues negotiation for $\delta = 0.02$ and various k-values. The figure suggests that the most of the randomly generated utility spaces remain tractable for the negotiation algorithm with the δ-checking procedure.

9. CONCLUSION

The paper proposes a δ-checking procedure that handles the short comings of multi-issue negotiation systems that base their operations on approximations of utility spaces with issue dependencies. In case the issues are interdependent, no efficient method exists to compute bids during a negotiation, even if the agent tries to guess the profile of the opponent [7]. To mitigate this problem, either mediators may be used, or the utility space corresponding to the interdependent issues can be approximated so that issues are no longer interdependent. The WAID-method presented in [5] is such an approximation method.

However, using an approximation always comes with a risk. In the case of multi-issue negotiation, the risk is that a bid is proposed (and accepted by the other party) that seems to have a good utility, but in fact, in the original utility space has a much lower utility. The δ-checking procedure proposed in this paper offers a way to avoid this risk at the cost of additional computations. Experimental results show, however, that a tradeoff can be made between the accuracy of the bids and the computational overhead this entails. If the δ-parameter in the checking procedure is set to 0.02, the utility of the bids made is at most 0.02 away from the real utility, on a scale from 0 to 1. Moreover, using this value for the δ-parameter, the negotiation algorithm including the δ-checking procedure can handle high-dimensional utility spaces. As experimental results show, the negotiation outcome obtained in this manner only slightly deviates from the outcome obtained without approximation.

To conclude, in this paper an effective balance is found of accuracy versus efficiency for multi-issue negotiation with issue dependencies in which the dependencies are removed by approximation.

10. REFERENCES

[1] Bar-Yam, Y., 1997. Dynamics of complex systems, Addison-Wesley (Reading).

[2] Davies, R., and Smith, R.G., 1983, Negotiation as a metaphor for distributed problem solving, in Artificial Intelligence, 20, 1, pp. 63 – 109.

[3] S. S. Fatima, M. Wooldridge and N. R. Jennings, 2006, On efficient procedures for multi-issue negotiation", in: Proc. 8th Int Workshop on Agent-Mediated Electronic Commerce, Hakodate, Japan, 71-84.

[4] Favati, P., Lotti, G., Romani, F., 1994. Theoretical and Practical Efficiency Measures for Symmetric Interpolatory Quadrature Fromulas, in BIT Numerical Mathematics, Volume 34(4).

[5] Hindriks, K., Jonker, C.M., Tykhonov, D., 2006, Eliminating Interdependencies between Issues for Multi-Issue Negotiation, In: Cooperative Information Agents X, Lecture Notes in Computer Science, Volume 4149, pp. 301-316.

[6] Jonker, C.M., and Treur, J., 2001, An Agent Architecture for Multi-Attribute Negotiation, in Proceedings of the 17th International Joint Conference on AI, IJCAI'01, ed-ited by B. Nebel, pp. 1195 – 1201.

[7] Klein, M., Faratin, P., Sayama, H., and Bar-Yam, Y., 2002, Negotiating Complex Contracts, in Autonomous Agents and Multi-Agent Systems, AAAI Press (Bologna).

[8] Lai, G., Li, C., Sycara, K., and Giampapa, J., 2004, Literature Review on Multi-attribute Negotiations, Technical Report CMU-RI-TR-04-66, Carnegie Mellon University, Robotics Institute.

[9] Pardalos, P.M., and Vavasis, S.A, 1991, Quadratic Programming with One Negative Eigenvalue Is NP-Hard, in: Journal of Global Optimization, 1:15 – 22.

[10] Raiffa, H., 1996, Lectures on Negotiation Analysis, PON Books, Program on Negotiation at Harvard Law School, 513 Pound Hall, Harvard Law School (Cambridge).

[11] Robu, V., Somefun, D.J.A., La Poutre, J.A., 2005, Complex Multi-Issue Negotiations Using Utility Graphs, in Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05), Utrecht, pp. 280-287.

[12] Rosenschein, J.S., and Zlotkin, G., 1994, Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers, MIT Press.

[13] Thompson, Leigh, 2000, The Mind and Heart of the Negotiator, Prentice-Hall.

[14] Wang, I-J., Chong, E. K. P., and Kulkarni, S. R., 1996, Weighted Averaging and Stochastic Approximation, in Proceeding of the 35th Conference on Decision and Control, Kobe, Japan, December 1996, pp. 1071-1076.