An Approach to Implementing A Threshold Adjusting Mechanism in Very Complex Negotiations: A Preliminary Result

Katsuhide Fujita[†] **Takayuki Ito**[‡] **Hiromitsu Hattori**[*] **Mark Klein**[**]
[†]Department of Computer Science, Nagoya Institute of Technology, Japan
fujita@longwood.mta.nitech.ac.jp
[‡]Techno-Business School / Department of Computer Science
Nagoya Institute of Technology, Japan
ito.takayuki@nitech.ac.jp
[*]Graduate School of Informatics, Kyoto University, Japan
hatto@i.kyoto-u.ac.jp
[**]Center for Collective Intelligence, Massachusetts Institute of Technology, USA
m_klein@mit.edu

Abstract

In this paper, we propose a threshold adjusting mechanism in very complex negotiations among software agents. The proposed mechanism can facilitate agents to reach an agreement while keeping their private information as much as possible. Multi-issue negotiation protocols have been studied widely and represent a promising field since most negotiation problems in the real world involve **interdependent multiple issues**. We have proposed negotiation protocols where a bidding-based mechanism is used to find social-welfare maximizing deals. The existing works have not yet concerned about agents' private information. Such private information should be kept as much as possible in their negotiation. Thus, in this paper, we propose a new threshold adjusting mechanism in which agents who open their local information more than the others can persuade the others. The preliminary experimental results demonstrate that the threshold adjusting mechanism can reduce the amount of private information that is required for an agreement among agents.

Keywords: Multi-issue Negotiation, Threshold Adjusting, Multi-agent Systems

1 Introduction

Multi-issue negotiation protocols represent an important field of study since negotiation problems in the real world are often complex ones involving multiple issues. While there has been a lot of previous work in this area ([1; 2; 3; 4]) these efforts have, to date, dealt almost exclusively with simple negotiations involving **independent multiple** issues, and therefore linear (single optimum) utility functions. Many real-world negotiation problems, however, involve **interdependent multiple** issues. When designers work together to design a car, for example, the value of a given carburetor is highly dependent on which engine is chosen. The addition of such interdependencies greatly complicates the agent's utility functions, making them nonlinear, with multiple optima. Negotiation mechanisms that are well suited for linear utility functions, unfortunately, fare poorly when applied to nonlinear problems ([5]).

We have proposed a bidding-based **multiple-issue** negotiation protocol suited for agents with such nonlinear utility functions. Agents generate bids by sampling their own utility functions to find local optima, and then using constraint-based bids to compactly describe regions that have large utility values for that agent. These techniques make bid generation computationally tractable even in large (*e.g.*, 10^{10} contracts) utility spaces. A mediator then finds a combination of bids that maximizes social welfare.

The existing works have not yet concerned about agents' private information. Such private information should be kept as much as possible in their negotiation. In this paper, we propose a threshold adjusting mechanism. First agents make bids that produce more utility than the common threshold value based on the bidding-based protocol we proposed in [6]. Then the mediator asks each agent to reduce its threshold based on how much each agent opens its private information to the others. Each agent makes bids again above the threshold. This process con-

tinues iteratively until agreement is reached or no solution. Our experimental results show that our method substantially outperforms the existing negotiation methods on the point of how much agents have to open their own utility space.

The remainder of the paper is organized as follows. First we describe a model of non-linear multi-issue negotiation. Second, we describe a bidding-based negotiation protocol designed for such contexts. Third we propose a threshold adjusting mechanism that helps agents to keep their private information secret as much as possible. Forth, we present experimental assessment of this protocol. Finally, we compare our work with previous efforts, and conclude with a discussion of possible avenues for future work.

2 Negotiation with Complex Utilities

2.1 Complex Utility Model

We consider the situation where n agents want to reach an agreement. There are m issues, $s_j \in S$, to be negotiated. The number of issues represents the number of dimensions of the utility space. For example, if there are 3 issues[1], the utility space has 3 dimensions. An issue s_j has a value drawn from the domain of integers $[0, X]$, *i.e.*, $s_j \in [0, X]$[2].

A contract is represented by a vector of issue values $\vec{s} = (s_1, ..., s_m)$.

An agent's utility function is described in terms of constraints. There are l constraints, $c_k \in C$. Each constraint represents a region with one or more dimensions, and has an associated utility value. A constraint c_k has value $w_i(c_k, \vec{s})$ if and only if it is satisfied by contract $\vec{s}$. Figure 1 shows an example of a binary constraint between issues 1 and 2. This constraint has a value of 55, and holds if the value for issue 1 is in the range $[3, 7]$ and the value for issue 2 is in the range $[4, 6]$. Every agent has its' own, typically

[1]The issues are not "distributed" over agents. The agents are all negotiating over a contract that has N (e.g. 10) issues in it. All agents are potentially interested in the values for all N issues.

[2]A discrete domain can come arbitrarily close to a real domain by increasing the domain size. As a practical matter, very many real- world issues that are theoretically real (delivery date, cost) are discretized during negotiations. Our approach, furthermore, is not theoretically limited to discrete domains. The deal determination part is unaffected, though the bid generation step will have to be modified to use a nonlinear optimization algorithm suited to real domains.

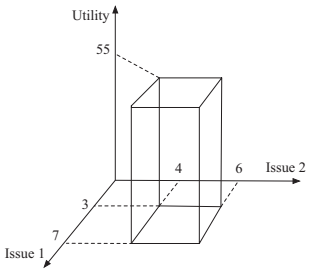

Figure 1. Example of A Constraint

unique, set of constraints.

An agent's utility for a contract $\vec{s}$ is defined as $u_i(\vec{s}) = \sum_{c_k \in C, \vec{s} \in x(c_k)} w_i(c_k, \vec{s})$, where $x(c_k)$ is a set of possible contracts (solutions) of c_k. This expression produces a "bumpy" nonlinear utility space, with high points where many constraints are satisfied, and lower regions where few or no constraints are satisfied. This represents a crucial departure from previous efforts on multi-issue negotiation, where contract utility is calculated as the weighted sum of the utilities for individual issues, producing utility functions shaped like flat hyper-planes with a single optimum. Figure 2 shows an example of a nonlinear utility space. There are 2 issues, *i.e.*, 2 dimensions, with domains $[0, 99]$. There are 50 unary constraints (*i.e.*, that relate to 1 issue) as well as 100 binary constraints (*i.e.*, that inter-relate 2 issues). The utility space is, as we can see, highly nonlinear, with many hills and valleys.

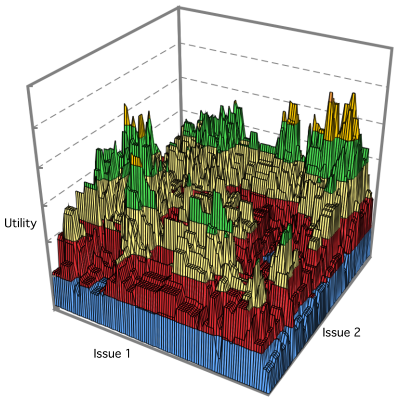

Figure 2. A Complex Utility Space for a Single Agent

We assume, as is common in negotiation contexts, which agents do not share their utility functions with each other, in order to preserve a competitive edge. It will generally be the case,

in fact, that agents do not fully know their desirable contracts in advance, because each own utility functions are simply too large. If we have 10 issues with 10 possible values per issue, for example, this produces a space of 10^{10} (10 billion) possible contracts, too many to evaluate exhaustively. Agents must thus operate in a highly uncertain environment.

Finding an optimal contract for individual agents with such utility spaces can be handled using well-known nonlinear optimization techniques such a simulated annealing or evolutionary algorithms. We cannot employ such methods for negotiation purposes, however, because they require that agents fully reveal their utility functions to a third party, which is generally unrealistic in negotiation contexts.

The objective function for our protocol can be described as follows:

$$\arg\max_{\vec{s}} \sum_{i \in N} u_i(\vec{s}) \qquad (1)$$

Our protocol, in other words, tries to find contracts that maximize social welfare, *i.e.*, the total utilities for all agents. Such contracts, by definition, will also be Pareto-optimal.

2.2 Bidding-based Consenting Protocol

The bidding-based negotiation protocol consists of the following four steps:

[Step 1: Sampling] Each agent samples its utility space in order to find high-utility contract regions. A fixed number of samples are taken from a range of random points, drawing from a uniform distribution. Note that, if the number of samples is too low, the agent may miss some high utility regions in its contract space, and thereby potentially end up with a sub-optimal contract.

[Step 2: Adjusting] There is no guarantee, of course, that a given sample will lie on a locally optimal contract. Each agent, therefore, uses a nonlinear optimizer based on simulated annealing to try to find the local optimum in its neighborhood. Figure 3 exemplifies this concept. In this figure, a black dot is a sampling point and a white dot is a locally optimal contract point.

[Step 3: Bidding] For each contract $\vec{s}$ found by adjusted sampling, an agent evaluates its utility by summation of values of satisfied constraints. If that utility is larger than the reservation value δ, then the agent defines a bid that

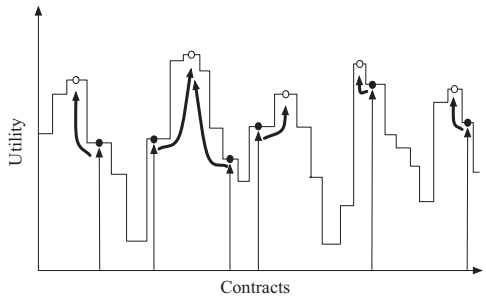

Figure 3. Adjusting the Sampled Contract Points

covers all the contracts in the region that has that utility value. This is easy to do: the agent need merely find the intersection of all the constraints satisfied by that $\vec{s}$.

Steps 1, 2 and 3 can be captured as follows:
SN: The number of samples
T: Temperature for Simulated Annealing
V: A set of values for each issue, V_m is for an issue m

1: **procedure** bid-generation with SA(*Th*, *SN*, *V*, *T*, *B*)
2: $P_{smpl} := \emptyset$
3: **while** $|P_{smpl}| < SN$
4: $P_{smpl} := P_{smpl} \cup \{p_i\}$ (randomly selected from P)
5: $P := \Pi_{m=0}^{|I|} V_m$, $P_{sa} := \emptyset$
6: **for each** $p \in P_{smpl}$ **do**
7: $p' :=$ simulatedAnnealing(p, T)
8: $P_{sa} := P_{sa} \cup \{p'\}$
9: **for each** $p \in P_{sa}$ **do**
10: $u := 0$, $B := \emptyset$, $BC := \emptyset$
11: **for each** $c \in C$ **do**
12: **if** c contains p as a contract and p satisfies c **then**
13: $BC := BC \cup c$
14: $u := u + v_c$
15: **if** $u >= Th$ **then**
16: $B := B \cup (u, BC)$

[Step 4: Deal identification] The mediator identifies the final contract by finding all the combinations of bids, one from each agent, that are mutually consistent, *i.e.*, that specify overlapping contract regions[3]. If there is more than one such overlap, the mediator selects the one with

[3]A bid has an acceptable region. For example, if a bid has a region, such as [0,2] for issue1, [3,5] for issue2, the bid is accepted by a contract point (1,4), which means issue1 takes 1, issue2 takes 4. If a combination of bids, i.e. a solution, is consistent, there are definitely overlapping re-

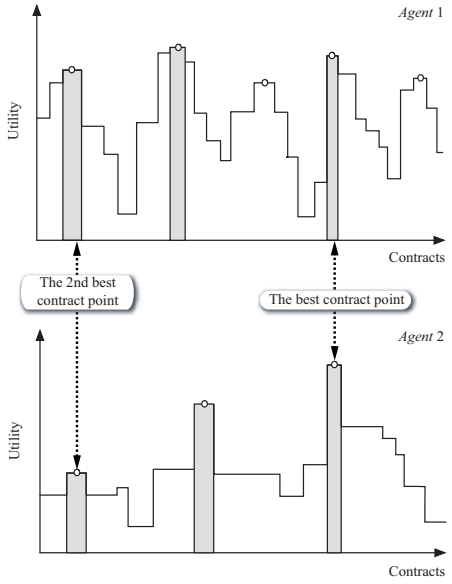

Figure 4. Deal Identification

the highest summed bid value (and thus, assuming truthful bidding, the highest social welfare) (see Figure 4). Each bidder pays the value of its winning bid to the mediator.

The mediator employs breadth-first search with branch cutting to find social-welfare-maximizing overlaps:
Ag: A set of agents
B: A set of Bid-set of each agent ($B = \{B_0, B_1, ..., B_n\}$,
A set of bids from agent i is $B_i = \{b_{i,0}, b_{i,1}, ..., b_{i,m}\}$)

1: **procedure** search_solution(B)
2: $SC := \bigcup_{j \in B_0} \{b_{0,j}\}, i := 1$
3: **while** $i < |Ag|$ **do**
4: $SC' := \emptyset$
5: **for each** $s \in SC$ **do**
6: **for each** $b_{i,j} \in B_i$ **do**
7: $s' := s \cup b_{i,j}$
8: **if** s' is consistent **then** $SC' := SC' \cup s'$
9: $SC := SC', i := i + 1$
10: $maxSolution$ = getMaxSolution(SC)
11: **return** $maxSolution$

It is easy to show that, in theory, this approach can be guaranteed to find optimal contracts. If every agent exhaustively samples every contract in its utility space, and has a reservation value of zero, then it will generate bids that represent the agent's complete utility function. The mediator, with the complete utility functions for all agents in hand, can use exhaustive search over all bid combinations to find the social welfare maximizing negotiation outcome. But this approach is only practical for very small contract spaces. The computational cost of generating bids and finding winning combinations grows rapidly as the size of the contract space increases. As a practical matter, we introduce the threshold to limit the number of bids the agents can generate. Thus, deal identification can terminate in a reasonable amount of time.

In the previous work [6], the threshold for each agent is commonly defined by the mediator. Agents could not change it by their selves. The threshold adjusting mechanism proposed in this paper allows agents to change their threshold values.

gion. For instance, a bid with regions (Issue1,Issue2) = ([0,2],[3,5]), and another bid with ([0,1],[2,4]) is consistent.

3 Threshold Adjusting Mechanism

3.1 The Outline of the Threshold Adjusting Mechanism

The main idea of the threshold adjusting mechanism is that if an agent reveals the larger area of his utility space, then he can persuade the other agents. On the other hand, an agent who reveals the small area of his utility space, he should adjust his threshold to agree with if no agreement is achieved. The revealed area can be defined how the agent reveals his utility space on his threshold value. The threshold value is defined at the same value beforehand. Then the threshold values are changed by each agent based on the amount of the revealed area afterwards. Figure 5 shows the concept of the revealed area of agent's utility space. If the agent decreases the threshold value, then this means that he reveals his utility space more.

Figure 6 shows an example of the threshold adjusting process among 3 agents. The upper figure shows the thresholds and the revealed areas before adjusting the threshold. The bottom figure shows the thresholds and the revealed areas after adjusting the threshold. In particular, in this case, agent 3 revealed the small amount of his utility space. The amount of agent 3's revealing utility space in this threshold adjusting is largest among these 3 agents. In the protocol, this process is repeated until an agreement is achieved or until they could not find any agreement. The

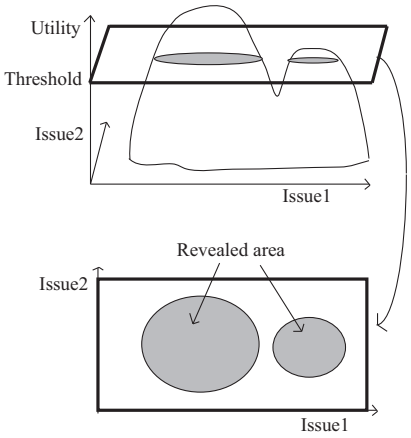

Figure 5. Revealed Area

exact rate of the amount of revealed utility space and the amount of decreasing the threshold is defined by the mediator or the mechanism designer.

The details of the threshold adjusting mechanism is shown as follows:
Ar: Area of each agent ($Ar = \{Ar_0, Ar_1, ..., Ar_n\}$)

1: **procedure** threshold_adjustment( )
2: **loop:**
3: $i := 1, B := \emptyset$
4: **while** $i < |Ag|$ **do**
5: bid_generation_with_SA(Th_i, V, SN, T, B_i)
6: $SC := \emptyset$
7: $maxSolution$:= search_solution(B)
8: **if** $maxSolution$ is not empty
9: $maxSolution$:= getMaxSolution(SC)
10: **break loop**
11: **elseif** all agent can lower the threshold
12: $i := 1$
13: $SumAr := \Sigma_{i \in |Ag|}\ Ar_i$
14: **while** $i < |Ag|$ **do**
15: $Th_i := Th_i - C * (\sigma_i Ar - Ar_i)/\sigma_i Ar$
16: $i := i + 1$
17: **end while**
18: **else**
19: **break loop**
20: **return** $maxSolution$

The above algorithm utilizes step 1, step 2, step 3, and step 4 in the previous section. In the former paper [6], we did not define any external loop of these steps. This paper is the first that proposed the external loop for an effective consenting mechanism.

3.2 Incremental Deal Identification

The threshold adjusting process shown in the previous section could reduce the computational cost of deal identification in step 4. The original step 4 requires an exponential computational cost because the computation is actually combinatorial optimization. In the new threshold adjusting process, agents incrementally reveal their utility spaces as bids. Thus, for each round, the mediator only computes the new combinations of bids that submitted newly in that round. This process actually reduces the computational cost. We just observed this fact in the preliminary experiments, and did not investigate this deeply. Thus future work includes the investigation of this good feature of incremental deal identification.

4 A Preliminary Experimental Result

We conducted several experiments to evaluate the effectiveness of our approach. In each experiment, we ran 100 negotiations between agents with randomly generated utility functions. We compare our new threshold adjusting protocol and the existing protocol without adjusting the threshold in terms of optimality and privacy.

In the experiments on optimality, for each run, we applied an optimizer to the sum of all the agents' utility functions to find the contract with the highest possible social welfare. This value was used to assess the efficiency (*i.e.*, how closely optimal social welfare was approached) of the negotiation protocols. To find the optimum contract, we used simulated annealing (SA) because exhaustive search became intractable as the number of issues grew too large. The SA initial temperature was 50.0 and decreased linearly to 0 over the course of 2500 iterations. The initial contract for each SA run was randomly selected.

In terms of privacy, the measure is the range of revealed area. Namely, if an agent reveals one point of the gird of utility space, this means he lost 1 privacy unit. If he reveals 1000 points, the he lost 1000 privacy.

The parameters for our experiments were as follows:

Number of agents is $N = 3$. Number of issues is 2 to 10. Domain for issue values is $[0, 9]$.

Constraints : 10 unary constraints, 5 binary constraints, 5 trinary constraints, etc. (a unary

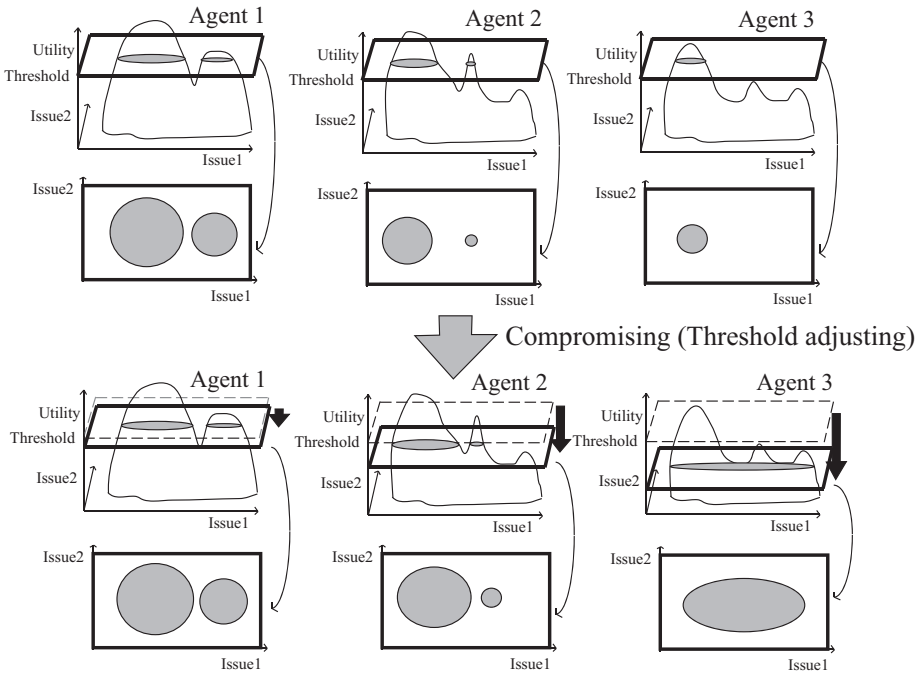

Figure 6. The Threshold Adjusting Process

constraint relates to one issue, an binary constraint relates to two issues, and so on).

The maximum value for a constraint : $100 \times (Number\ of\ Issues)$. Constraints that satisfy many issues thus have, on average, larger weights. This seems reasonable for many domains. In meeting scheduling, for example, higher order constraints concern more people than lower order constraints, so they are more important for that reason.

The maximum width for a constraint : 7. The following constraints, therefore, would all be valid: issue 1 = $[2, 6]$, issue 3 = $[2, 9]$ and issue 7 = $[1, 3]$.

The number of samples taken during random sampling: $(Number\ of\ Issues) \times 200$.

Annealing schedule for sample adjustment: initial temperature 30, 30 iterations. Note that it is important that the annealer not run too long or too 'hot', because then each sample will tend to find the global optimum instead of the peak of the optimum nearest the sampling point.

The threshold agents used to select which bids to make in starts with 900 and decreases until 200 in the threshold adjusting mechanism. The protocol without the threshold adjusting process defines the threshold as 200. The threshold is used to cut out contract points that have low utility.

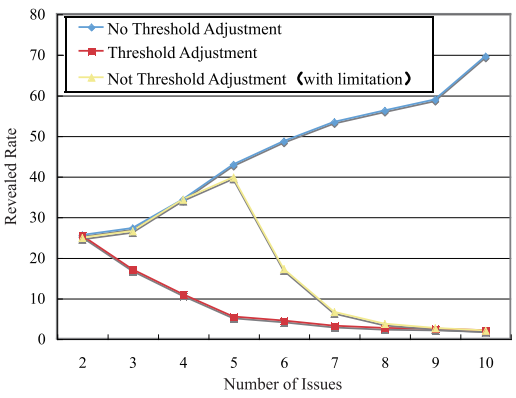

Figure 7. Revealed Rate

The limitation on the number of bids per agent: $\sqrt[n]{6400000}$ for N agents. It was only practical to run the deal identification algorithm if it explored no more than about 6400,000 bid combinations, which implies a limit of $\sqrt[n]{6400000}$ bids per agent, for N agents.

In our experiments, we ran 100 negotiations in every condition. Our code was implemented in Java 2 (1.5) and run on a core 2 duo processor iMac with 1.0GB memory under Mac OS X 10.4.

Figure 7 shows the optimality of 3 comparable

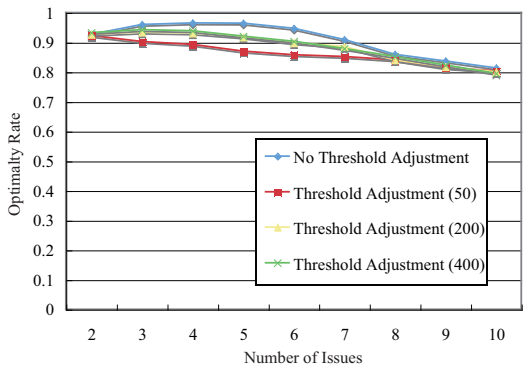

Figure 8. Optimality

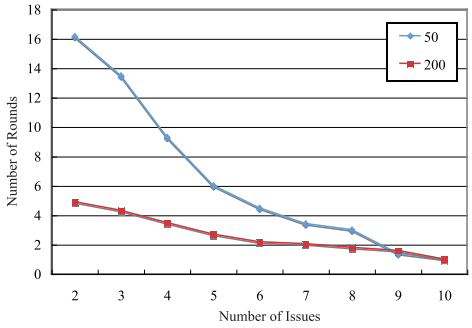

Figure 9. Number of rounds

mechanisms, one with the threshold adjustment (with bid limitation), one without the threshold adjustment and bid limitation, and one without the threshold adjustment with bid limitation. The revealed rate is defined by $(Revealedrate) = (Revealedarea)/(Wholeareaofutilityspace)$.

The mechanism without both of the threshold adjustment and bid limitation increasing the revealed rate. This means that if we do not use the threshold adjustment and bid limitation, then agents need to reveal their utility space much more than the other mechanisms.

We found that bid limitation can show the nice effects to keep the increasing amount of revealed rate small. The mechanism with bid limitation but without the threshold adjustment shown by triangles starts decreasing when the number of issue is 5, namely bid limitation starts being active.

Compared with the above two mechanisms, the mechanism with the threshold adjustment proposed inn this paper drastically decreases the amount of the revealed rate.

As we show in the previous paragraph, our proposed threshold adjustment mechanism can effectively reduce the revealed rates. We then show the optimality of our proposed mechanism is quite competitive compared with the other mechanisms in Figure 8.

Each line in Figure 8 means the followings: No threshold adjustment means a mechanism without the threshold adjustment. Threshold adjustment (50), Threshold adjustment (200), and Threshold adjustment (400) mean mechanisms with the threshold adjustment. Each mechanism determines the decreasing amount of the threshold by $50 \times (SumAr - Ar_i)/SumAr$, $200 \times (SumAr - Ar_i)/SumAr$, and $400 \times (SumAr - Ar_i)/SumAr$, respectively. $SumAr$ means the sum of all agents' revealed areas. Ar_i means $agent_i$'s revealed area.

As we can see in Figure 8, in terms of the optimality, the difference between "no threshold adjustment" and "threshold adjustments" is small. At most the difference is around 0.1 around 3 issues to 7 issues. When the threshold decreasing amount is not large, say 50, agents could miss the agreement points that have larger total utilities. This occurs when some agents have higher utility on the agreement point but other agents have very lower utility on the agreement point. "No threshold adjustment" mechanism makes agents to submit all agreement points that have larger utility than the minimum threshold. Thus, "No threshold adjustment" can find such cases. But "threshold adjustment" mechanisms fail to capture such cases when the decreasing amount is smaller.

Figure 9 and Figure 10 compare the required rounds and the revealed rates for different decreasing amounts, 50 and 200. Figure 9 demonstrates that the decreasing amount is small, 50, then the number of rounds could be larger. On the other hand, in Figure 10, if the decreasing amount is small, then the revealed rate is relatively small.

5 Related Work

Most previous work on multi-issue negotiation ([7; 1; 2]) has addressed only linear utilities. A handful of efforts have, however, considered nonlinear utilities. [8] has explored a range of protocols based on mutation and selection on binary contracts. This paper does not describe

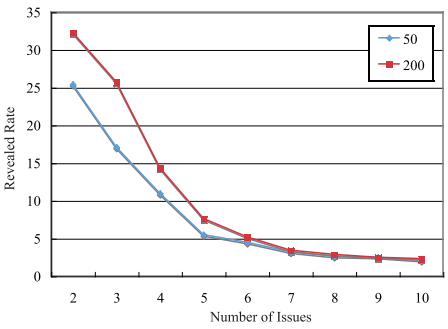

Figure 10. Revealed Rates

what kind of utility functions is used, nor does it present any experimental analyses. It is therefore unclear whether this strategy enables sufficient exploration of the utility space to find win-win solutions with multi-optima utility functions. [9] presents an approach based on constraint relaxation. In the proposed approach, a contract is defined as a goal tree, with a set of on/off labels for each goal, which represents the desire that an attribute value is within a given range. There are constraints that describe what patterns of on/off labels are allowable. This approach may face serious scalability limitations. However, there is no experimental analysis and this paper presents only a small toy problem with 27 contracts. [10] also presents constraint based approach. In this paper, a negotiation problem is modeled as a distributed constraint optimization problem. During exchanging proposals, agents relax their constraints, which express preferences over multiple attributes, over time to reach an agreement. This paper claims the proposed algorithm is optimal, but do not discuss computational complexity and provides only a single small-scale example. [5] presented a protocol, based on a simulated-annealing mediator, that was applied with near-optimal results to medium-sized bilateral negotiations with binary dependencies. The work presented here is distinguished by demonstrating both scalability, and high optimality values, for multilateral negotiations and higher order dependencies.

6 Conclusions

In this paper, we proposed a threshold adjusting mechanism in very complex negotiations among software agents. In very complex negotiations, we assume agents have to do interdependent multi-issue negotiation. The threshold adjusting mechanism can facilitate agents to reach an agreement while keeping their private information as much as possible. The preliminary experimental results demonstrate that the threshold adjusting mechanism can reduce the amount of private information that is required for an agreement among agents. One of the interesting future works includes more autonomous threshold adjustment mechanism for agents in very complex negotiations.

References

[1] P. Faratin, C. Sierra, and N.R. Jenning. Using similarity criteria to make issue tradeoffs in automated negotiations. In *Artificial Intelligence*, pp. 142:205–237, 2002.

[2] S. Fatima, M. Wooldridge, and N.R. Jennings. Optimal negotiation of multiple issues in incomplete information settings. In *AAMAS-2004*, pp. 1080–1087, 2004.

[3] R. Y. K. Lau. Towards genetically optimised multi-agent multi-issue negotiations. In *of HICSS-2005*, 2005.

[4] L.K. Soh and X. Li. Adaptive, confidence-based multiagent negotiation strategy. In *AAMAS-2004*, pp. 1048–1055, 2004.

[5] M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam. Negotiating complex contracts. *Group Decision and Negotiation*, 12(2):58–73, 2003.

[6] T. Ito, H. Hattori, and M. Klein. Multi-issue negotiation protocol for agents : Exploring nonlinear utility spaces. In *JCAI-2007*, pp. 1347–1352, 2007.

[7] T. Bosse and C. M. Jonker. Human vs. computer behaviour in multi-issue negotiation. In *RRS-2005*, pp. 11–24, 2005.

[8] R. J. Lin and S. T Chou. Bilateral multi-issue negotiations in a dynamic environment. In *AMEC-2003*, 2003.

[9] M. Barbuceanu and W. K. Lo. Multi-attribute utility theoretic negotiation for electronic commerce. In *AMEC-2000*, pp. 15–30, 2000.

[10] X. Luo, N. R. Jennings, N. Shadbolt, H. Leung, and J.H. Lee. A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. *Artificial Intelligence*, 148:53–102, 2003.