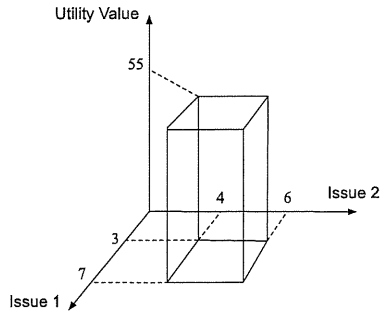


Fig. 1. Example of a cube-constraint.

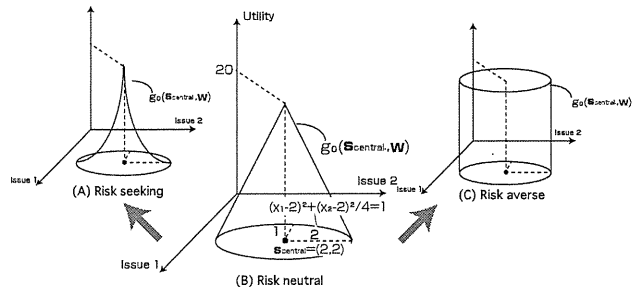


Fig. 2. Example of cone-constraints.

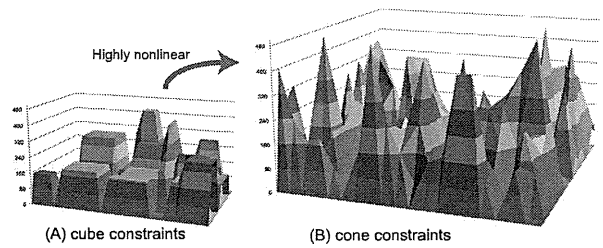


Fig. 3. Example of utility space with cone constraints.

utility space dimensions. For example, if there are three issues, the utility space has three dimensions. The issues are not "distributed" over agents, who are all negotiating a contract with N (e.g., 10) issues in it. All agents are potentially interested in the values for all N issues. Issue s_j has a value drawn from the domain of integers $[0,X]$, i.e., $s_j \in [0,X](1 \leq j \leq M)$. A contract is represented by a vector of issue values $\vec{s} = (s_1, ..., s_m)$. The objective function for agreement search protocols can be described as follows:

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

The proposed protocols in the literature try to find contracts that maximize social welfare, i.e., the total utilities for all agents. Such contracts, by definition, will also be Pareto-optimal.

In this paper, we deal with cube-constraints and cone-constraints as the utility function. Every agent has its own, typically unique, set of constraints.

[Cube-constraints] An agent's utility function is described in terms of constraints [5]. There are l constraints, $c_k \in C$. Each constraint represents a region with one or more dimensions and has an associated utility value. Constraint c_k has value $w_i(c_k, \vec{s})$ if and only if it is satisfied by contract $\vec{s}(1 \leq k \leq l)$. We call this type of constraint a "cube-constraint." **Fig. 1** shows an example of a binary constraint between Issues 1 and 2. This constraint, which has a value of 55, holds if the value for Issue 1 is in the range $[3,7]$ and the value for Issue 2 is in the range $[4,6]$.

In recent works (e.g., [7]), several types of cube-constraints were proposed. We also include a variety of cube-constraints in our testbed.

[Cone-constraints] An agent's utility function can be described in terms of cone-constraints. By formalizing risk attitude in terms of the cone-constraints, utility function of agents captures the utility information in real world. **Fig. 2** shows an example of a binary cone-constraint between Issues 1 and 2. This cone-constraint has a value of 20, which is maximum if the situation is $\vec{s}_{central} = [2,2]$. The impact region is $\vec{w} = [1,2]$. The expression for a segment of the base is $(x_1 - 2)^2 + (x_2 - 2)^2/4 = 1$.[1]

Suppose there are l cone-constraints, $C = \{c_k | \ 1 \leq k \leq l\}$. Cone-constraint c_k has gradient function $g_k(\vec{s}_{central}, \vec{w})$, which is defined by two values: central value $\vec{s}_{central}$, which is the highest utility in c_k, and impact region $\vec{w}$, which represents the region where c_k is affected. We assume not only circle-based but also ellipse-based cones. Thus constraint c_k has value $u_i(c_k, \vec{s})$ if and only if it is satisfied by contract $\vec{s}$. In this paper, impact region $\vec{w}$ is not a value but a vector. These formulas can represent utility spaces if they are in a n-dimensional space.

In addition, cone-constraints can include the risk attitude for constraints by configuring gradient function $g_k(\vec{s}_{central}, \vec{w})$. If the agent usually has a risk neutral attitude for c_k, g_k is defined as **Fig. 2(B)** (e.g., proportion). However, the attitudes (types) of agent can change from risk-seeking to risk-averse for making agreements. For example, if agents have a risk-seeking attitude for constraint c_k, g_k is defined as **Fig. 2(A)** (e.g., exponent). If an agent has a risk-averse attitude for c_k, g_k is defined as **Fig. 2(C)**. If agents have the most risk-averse attitude for c_k, g_k stays constant. Therefore, c_k is shaped like a column if the agents have the most risk-averse attitude. In real world, there are at least two kinds of risk: 1) the risk of getting a bad deal, 2) the risk of failing to get a deal. In this paper, we assume "2) risk of failing to get a deal."

An agent's utility for 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.

Figure 3 shows an example of a nonlinear utility space with two issues. This utility space is highly nonlin-

1. The general expression is $\sum_{i=1}^{m} x_i^2/w_i^2 = 1$

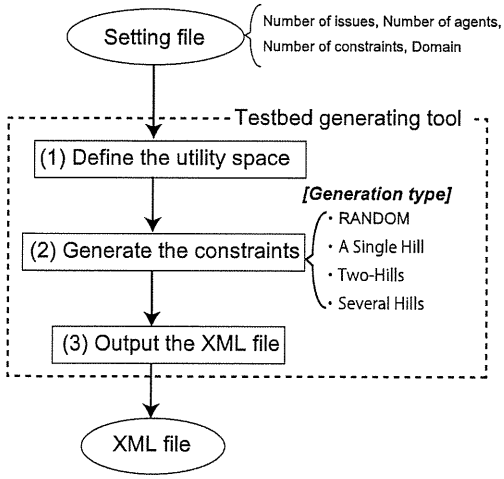


Fig. 4. Flow of testbed generating tool.

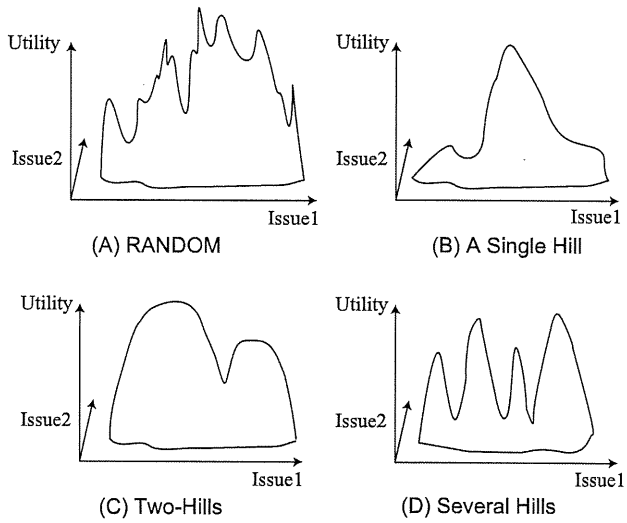


Fig. 5. Generation type.

ear with many hills and valleys. [5] proposed a utility function based on "cube"-constraints. Compared with cube-constraints, highest point in the utility space is narrower. Therefore, the protocols for making agreements must search in highly nonlinear utility space. A simple simulated annealing method to directly find optimal contracts is especially insufficient in a utility function based on cone-constraints.

We assume, as is common in negotiation contexts, that agents do not share their utility functions with each other to preserve a competitive edge. Generally, in fact, agents do not completely know their desirable contracts in advance, because their 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, which is too many to evaluate exhaustively. Agents must thus operate in a highly uncertain environment.

3. Common Testbed Based on XML for Negotiation Protocols

3.1. Testbed Generating Tool

We have been implementing a common testbed generating tool for multi-issue negotiation protocols based on XML. The input of a testbed generating tool is a configuration file that includes the number of issues and the number of agents. The output is an XML file that defines the agents' utility spaces. The source code for this tool is downloadable from: *http://www-itolab.mta.nitech.ac.jp/MultiIssueNegotiations.*

Figure 4 shows the program flow of our testbed generating tool. First, the utility space is defined based on the configuration file. Second, constraints are generated based on the specified type of utility spaces. Finally, an XML file is outputted. The details of the testbed generating tool are shown as follows:

(1) Defining utility space: The testbed generating tool defines the utility space information based on the configuration file. The configuration file includes the number of issues, agents, and constraints as well as the value domain per issue. Constraints are classified by the number of related constraints. For example, a unary constraint is related to one issue, a binary constraint is related to two issues, etc. In the configuration file, we write the number of constraints for each related constraint like "unary constraints include 10, binary constraints include 5, etc."

(2) Generating utility spaces: In the current implementation, the testbed generating tool generates utility spaces based on four different types of utility spaces: Random, A Single Hill, Two Hills, and Several Hills. Statements about the details of each type are shown as follows.

Random: in this type, constraints are generated randomly. Such generation is used in the experiments in several works [5]. **Fig. 5** shows an example of utility space plotted by all statements as agent constraints. This utility space plotted is highly nonlinear, as **Fig. 5(A)** shows.

A Single Hill: an example of this type is a collaborative negotiation among the same type of agents. The utility space plotted by all agents has one higher point, as **Fig. 5(B)** shows. In such utility spaces, reaching an agreement is usually easy.

Two Hills: an example of this type is a bilateral negotiation between two types of agents. In particular, such negotiation between buyers and sellers is popular. The utility space plotted by all agents has two higher points, as **Fig. 5(C)** shows. In such utility spaces, making agreements is hard because the agents are likely in a hostile relation.

Several Hills: an example of this type is collaborative negotiation among more than three other types of agents. Collaborative design for a car among designers, engineers, and business managers is a concrete example. The utility space plotted by all agents' constraints has more

```
<?xml version="1.0" encoding="Shift_JIS" standalone="no"?>
<UtilitySpace>
<Dimension>4</Dimension>
<Domain>0-9</Domain>
<Agent no=0 name="Alice">
<ReservationValue>11</ReservationValue>
<Constraint no=0 name="0">
<Cardinality>2</Cardinality>
<Utility>69</Utility>
<Minimum>
<Issue no=2 name="size"> 4 </Issue>
</Minimum>
<Maximum>
<Issue no=2 name="size"> 8 </Issue>
</Maximum>
</Constraint>
<Constraint no=1 name="1">
...
</Constraint>
</Agent>
</UtilitySpace>
<Agent no=1 name="Bob">
<ReservationValue>15</ReservationValue>
<Constraint no=0 name="0">
<Cardinality>1</Cardinality>
...
</Constraint>
</Agent>
</UtilitySpace>
```

Fig. 6. Example XML for cube-constraints.

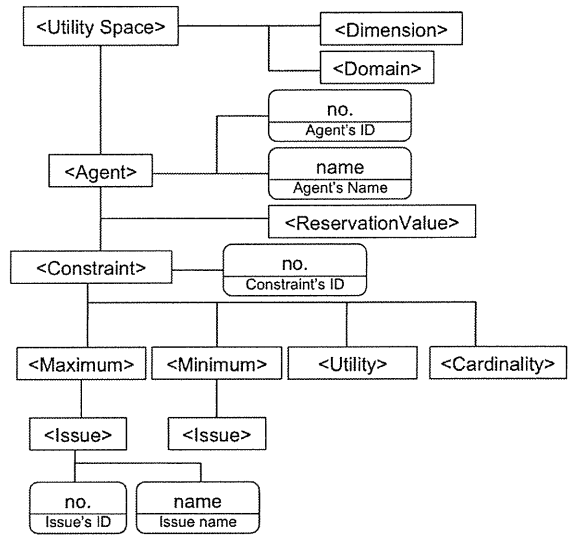


Fig. 7. Tree-structured XML chart for cube-constraints.

than three higher points, as **Fig. 5(D)** shows. In such utility spaces, finding agreement points is hard because there are too many hills. Thus search algorithms usually try to find the highest points.

(3) Output XML file: The testbed generating tool outputs the XML file on the testbed for negotiation. By outputting these files, users can easily understand the information. Additionally, users can modify, change, and update the data, and XML data are not dependent on a certain environment. Users like research communities can also easily exchange data with each other. The details of the XML formats are described in the next subsection.

3.2. XML Format for Testbeds

We propose the XML format for expressing the agent's utility function. In XML, this information is defined by tags. The specification of XML formats in cube-constraints and cone-constraints is described as follows.

XML format for cube-constraints: **Fig. 6** shows an example of the XML format for cube-constraints. **Fig. 7** shows a tree-structured chart for cube-constraints. The tree-structured chart enables us to understand the parent-child relation between elements. A detailed description of the elements is described as follows.

<UtilitySpace>: UtilitySpace element shows the specification information about the entire utility space. This element has the subelements of "Dimension," "ValueNumber," and "Agent."

<Dimension>: This element specifies the number of issues. In **Fig. 6**, the number of issues is four.

<Domain>: This element specifies the value domain for each issue. In **Fig. 6**, the domain of all issues is 0.9.

<Agent>: This element, which specifies the agents, has attributes of agent's ID and name. In **Fig. 6**, the agent's ID is 0 and its name is Alice. There could be multiple agent elements in UtilitySpace element. This element has the subelements of ReservationValue and many Constraint elements.

<ReservationValue>: This element specifies the reservation utility value for determining whether to "agree" or "disagree" with the contract alternatives in a negotiation. In **Fig. 6**, the reservation value is 21.

<Constraint>: This element, which defines the constraints, has the ID of the constraint as an attribute. This element has the subelements of Issue, Utility, and Cardinality. In **Fig. 6**, the ID of the constraints is 0.

<Minimum>: This element defines the possible minimum values for each issue. In **Fig. 6**, the possible minimum value of Issue 2 is 4. This means that the value for the issue should have more than 4.

<Maximum>: This element defines the possible maximum values for each issue. In **Fig. 6**, the possible maximum value of Issue 2 is 8. This means that the value for the issue should have less than 8.

<Utility>: This element defines the utility value in this constraint. The constraints have this utility value if the value for each issue is in the range defined by Issue elements. In **Fig. 6**, constraint 0 has a value of 69, and it holds if the value for Issue 1 is 0, the value for issue 2 is 8, the value for Issue 3 is in the range $[4,8]$, and the value for Issue 4 is 4.

<Cardinality>: This element shows the number of issues related to this constraint. In **Fig. 6**, the cardinality is one. This is because this constraint is related to issue 2. In the other words, this constraint is constrained by a issues. In our definition, the contract has a value if only the issues related to the constraints satisfy the possible values. In other words, all values are permitted in other issues not related to the constraint.

XML formats for cone-constraints: **Fig. 8** shows an example of an XML for cone-constraints. **Fig. 9** shows a

```
<?xml version="1.0" encoding="Shift_JIS" standalone="no"?>
<UtilitySpace>
<Dimension>5</Dimension>
<Domain>0-10</Domain>
<Agent name="Alice" no="0">
<ReservationValue>21</ReservationValue>
<Constraint no="0">
<Cardinality>1</Cardinality>
<MaxUtility>122</MaxUtility>
<RiskAttitude>1</RiskAttitude>
<CenterPoint>
<Issue name="4" no="4">0</Issue>
</CenterPoint>
<Width>
<Issue name="4" no="4">2</Issue>
</Width>
</Constraint>
<Constraint no="1">
<Cardinality>1</Cardinality>
...
</Constraint>
</Agent>
</UtilitySpace>
```

Fig. 8. Cone-constraints XML.

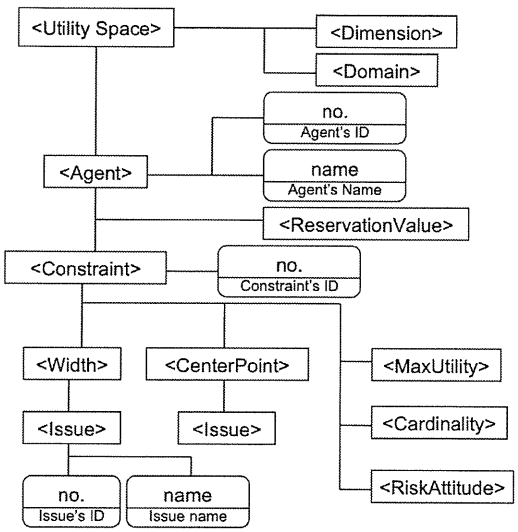


Fig. 9. Tree-structured chart for cone-constraints XML.

tree-structured chart for cone-constraints. The XML elements in the "UtilitySpace" and "Agents" elements are almost the same as the XML elements for cube-constraints. A detailed description of the elements in the cone-based constraints is described as follows.

<*MaxUtility*>: This element shows the central value, which is the highest utility in the constraint. In **Fig. 8**, the central value is 122, which is the maximum utility in the constraint.

<*RiskAttitude*>: This element shows a gradient function that represents the risk attitude for making agreements. In our testbed generating tool, we defined a gradient function for each number. For example, one is defined that a gradient function constant is constant. In **Fig. 8**, the risk attitude for making agreements is one. Future work includes an extension that enables users to simply define the gradient function.

<*Width*>: This element shows the impact region, which

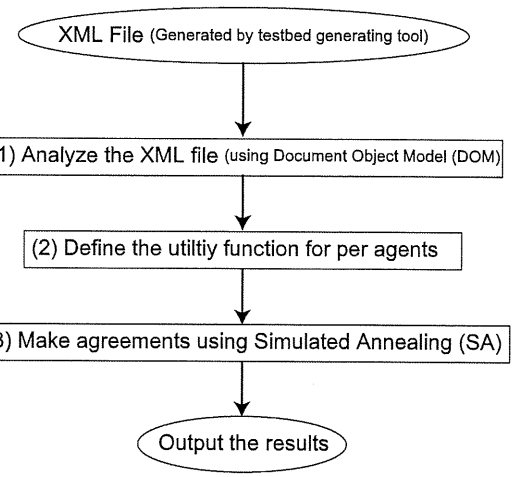


Fig. 10. Program flow using testbeds.

represents the region affected by the constraint. The impact region is defined in each Issue element. In **Fig. 8**, the impact region in Issue 4 is two.

<*CenterPoint*>: This element shows the central point, where the utility is maximum. In the CenterPoint element, the central point is defined by Issue elements. In **Fig. 8**, the central point is 0 in Issue 4 and all values are permitted in other issues (Issues 0–3).

4. Java Program Using the Testbed

In this subsection, we describe the Java program using the testbeds proposed in the previous section. Our code was implemented in Java 2 (1.5). The program source codes are downloadable from: *http://www-itolab.mta.nitech.ac.jp/MultiIssueNegotiations/*.

Figure 10 shows the flow of the JAVA program using testbeds. This program inputs XML files generated by the tool. The following are the details of this program behavior:

Analyzing XML files: in this program, an XML file is analyzed by a Document Object Model (DOM) [6], which is a platform and a language-independent standard object model for representing HTML or XML documents as well as an Application Programming Interface (API) for querying, traversing, and manipulating such documents. The information of the structure of the utility space and the agent's utility function are read from XML files.

Defining the utility function for each agent: the structure of the utility space and the agent's utility function are defined based on the XML analyzed in the previous step.

Searching agreements using SA: in this program, we provide a simple agreement algorithm that gathers and aggregates all individual agent's utility spaces into one central place and then finds the most optimal contract using Simulated Annealing (SA) [8]. In simulated annealing, the mediator moves randomly if the temperature is high, but he/she moves to the highest neighbor if the temper-

ature is low. A simulated-annealing method of making agreements was employed in previous works [5] because this search method is superior to other search methods, such as hill climbing search in multi interdependent issue negotiation.

In future work, we will generate this program using other programming languages such as C++, Ruby, Python, and Perl so that this testbed can be used by many users.

5. Related Works

As far as the authors know, this is the first attempt to create a testbed for multiple interdependent issue negotiation protocols. The following is a literature review of multi-issue negotiation problems. All of these protocols are evaluated on the original testbed. Our testbed might provide opportunities to compare these algorithms based on the same criteria.

Most previous work on multi-issue negotiation ([1, 2, 9]) has only addressed linear utilities. Some researchers have been focusing on more complex and nonlinear utilities. [10] explored a range of protocols based on mutation and selection on binary contracts. This paper does not describe what kind of utility function is used, nor does it present any experimental analyses, so it remains unclear whether this strategy enables sufficient exploration of utility space. [11] presents an approach based on constraint relaxation. [12] presented a protocol that was applied with near-optimal results on 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. [13] and [14] also presented a protocol for multi-issue problems for bilateral negotiations. [15] and [16] presented a multi-item and multi-issue negotiation protocol for bilateral negotiations in electronic commerce situations.

[17] proposed bilateral multi-issue negotiations with time constraints, and [18] proposed multi-issue negotiations that employ a third party to act as a mediator to guide agents toward equitable solutions. This framework also employs an agenda that serves as a schedule for the ordering of issue negotiations. Agendas are very interesting because agents only need to focus on a few issues. [19] proposed a checking procedure to mitigate this risk and showed that by tuning this procedure's parameters, outcome deviation can be controlled. These studies reflect interesting viewpoints, but they focused on just bilateral trading or negotiations.

6. Conclusion

In this paper, we proposed a testbed generating tool based on XML for multi-issue negotiation. Our tool provides a common testbed to evaluate the effectiveness of multi-issue negotiation protocols. Moreover, users can easily understand the meaning of data because it is based on a simple XML format. In this testbed, four types of utility spaces were provided that corresponded to real negotiation cases. Finally, we demonstrated examples of experiments using our testbed in which we analyzed the differences among types of utility spaces.

References:

[1] T. Bosse and C. M. Jonker, "Human vs. Computer Behaviour in Multi-Issue Negotiation," In Proc. of 1st Int. Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems (RRS-2005), pp. 11-24, 2005.

[2] P. Faratin, C. Sierra, and N. R. Jennings, "Using Similarity Criteria to Make Issue Trade-offs in Automated Negotiations," In Artificial Intelligence, Vol.142, pp. 205-237, 2002.

[3] K. Fujita, T. Ito, and M. Klein, "Preliminary Result on Secure Protocols for Multiple Issue Negotiation Problems," In Proc. of The 11th Pacific Rim Int. Conf. on Multi-Agents (PRIMA-2008), 2008.

[4] K. Fujita, T. Ito, and M. Klein, "A representative-based multi-round protocol for multi-issue negotiations," In Proc. of The 7th Int. Joint Conf. on Autonomous Agents and Multi-agent Systems (AAMAS-2008), 2008.

[5] T. Ito, H. Hattori, and M. Klein, "Multi-issue Negotiation Protocol for Agents: Exploring Nonlinear Utility Spaces," In Proc. of 20th Int. Joint Conference on Artificial Intelligence (IJCAI-2007), pp. 1347-1352, 2007.

[6] W3C. "Document Object Model (DOM) Technical Reports." http://www.w3.org/DOM/DOMTR

[7] J. R. V. I. Marsa-Maestre, M. A. Lopez-Carmona, and E. d. l. Hoz, "Effective bidding and deal identification for negotiations in highly nonlinear scenarios," In Proc. of The 8th Int. Joint Conf. on Autonomous Agents and Multi-agent Systems (AAMAS-2009), 2009.

[8] S. J. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," Prentice Hall, 2002.

[9] S. S. Fatima, M. Wooldridge, and N. R. Jennings, "Optimal Negotiation of Multiple Issues in Incomplete Information Settings," In Proc. of Third Int. Joint Conf. on Autonomous Agent and Multi-Agent Systems, (AAMAS-2004), pp. 1080-1087, 2004.

[10] R. J. Lin and S. T. Chou, "Bilateral Multi-Issue Negotiations in a Dynamic Environment," In Proc. of AMEC-2003, 2003.

[11] M. Barbuceanu and W.-K. Lo, "Multi-attribute Utility Theoretic Negotiation for Electronic Commerce," In Proc. of AMEC-2000, pp. 15-30, 2000.

[12] M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam, "Negotiating Complex Contracts," Group Decision and Negotiation, Vol.12, No.2, pp. 58-73, 2003.

[13] G. Lai, C. Li, and K. Sycara, "A general model for pareto optimal multi-attribute negotiations," In Proc. of The 2nd Int. Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems (RRS-2006), 2006.

[14] G. Lai, K. Sycara, and C. Li, "A decentralized model for multi-attribute negotiations with incomplete information and general utility functions," In Proc. of The 2nd Int. Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems (RRS-2006), 2006.

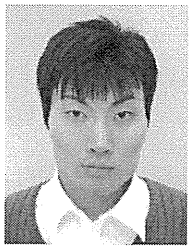
[15] V. Robu, D. J. A. Somefun, and J. L. Poutre, "Modeling complex multi-issue negotiations using utility graphs," In Proc. of the 4th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), 2005.

[16] V. Robu and H. L. Poutre, "Retrieving the Structure of Utility Graphs Used in Multi-Item Negotiation through Collaborative Filtering of Aggregate Buyer Preferences," In Proc. of The 2nd Int. Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems (RRS-2006), 2006.

[17] S. S. Fatima, M. Wooldridge, and N. R. Jennings, "Approximate and online multi-issue negotiation," In Proc. of th 6th Int. Joint Conf. on Autonomous Agents and Multi-agent Systems (AAMAS-2007), pp. 947-954, 2007.

[18] J. Shew and K. Larson, "The Blind Leading the Blind: A Third-Party Model for Bilateral Multi-issue Negotiations under Incomplete Information," In Proc. of The 1st Int. Workshop on Agent-based Complex Automated Negotiations (ACAN-2008), 2008.

[19] K. Hindriks, C. Jonker, and D. Tykhonov, "Avoiding Approximation Errors in Multi-Issue Negotiation with Issue Dependencies," In Proc. of The 1st Int. Workshop on Agent-based Complex Automated Negotiations (ACAN-2008), 2008.



Name:
Katsuhide Fujita

Affiliation:
Nagoya Institute of Technology
Massachusetts Institute of Technology

Address:
Gokiso-cho, Showa-ku, Nagoya, Aichi 466-8555, Japan

Brief Biographical History:
2008 Bachelor's degree of Engineering from the Nagoya Institute of Technology
2010 Master of Engineering from the Nagoya Institute of Technology
2010- Visiting Student at Sloan School of Management, Massachusetts Institute of Technology

Main Works:
- K. Fujita, T. Ito, and M. Klein, "Secure and efficient protocols for multiple interdependent issues negotiation," J. of Intelligent and Fuzzy Systems Vol.21, No.3, pp. 175-185, 2010.
- K. Fujita, T. Ito, and M. Klein, "A Secure and Fair Protocol that Addresses Weaknesses of the Nash Bargaining Solution in Nonlinear Negotiation," Group Decision and Negotiation, Springer, 10.1007/s10726-010-9194-6, 2010.

Membership in Academic Societies:
- The Institute of Electrical and Electronics Engineers (IEEE)
- Information Processing Society of Japan (IPSJ)
- The Institute of Electronics, Information, and Communicate Engineering (IEICE)
- The Japanese Society for Artificial Intelligence (JSAI)

Name:
Takayuki Ito

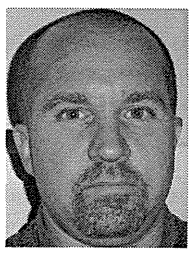
Affiliation:
Nagoya Institute of Technology
Massachusetts Institute of Technology

Address:
Gokiso-cho, Showa-ku, Nagoya, Aichi 466-8555, Japan

Brief Biographical History:
1995 B.E. of Engineering from the Nagoya Institute of Technology
1997 M.E of Engineering from the Nagoya Institute of Technology
2000 Doctor of Engineering from the Nagoya Institute of Technology
2000-2001 Visiting Researcher, USC/ISI
2001-2003 Associate Professor, Japan Advanced Institute of Science and Technology (JAIST)
2003- Associate Professor, Graduate School of Engineering, Nagoya Institute of Technology
2005-2006 Visiting Researcher, Division of Engineering and Applied Science, Harvard University
2005-2006 Visiting Researcher, Sloan School of Management, Massachusetts Institute of Technology
2008-2009 Visiting Researcher, Sloan School of Management, Massachusetts Institute of Technology

Main Works:
- K. Fujita, T. Ito, and M. Klein, "A Secure and Fair Protocol that Addresses Weaknesses of the Nash Bargaining Solution in Nonlinear Negotiation," Group Decision and Negotiation, Springer, 10.1007/s10726-010-9194-6, 2010.

Membership in Academic Societies:
- The Association for Computing Machinery (ACM)
- The Institute of Electrical and Electronics Engineers (IEEE)
- Association for the Advancement of Artificial Intelligence (AAAI)
- Information Processing Society of Japan (IPSJ)
- The Institute of Electronics, Information, and Communicate Engineering (IEICE)
- The Japanese Society for Artificial Intelligence (JSAI)



Name:
Mark Klein

Affiliation:
Massachusetts Institute of Technology

Address:
5 Cambridge Center, NE25-754, Cambridge, MA 02139, USA

Brief Biographical History:
1989 Ph.D. in Computer Science, University of Illinois
1995- Research Faculty, the Applied Research Lab Information Systems Department, Pennsylvania State University
1997- Research Associate, the MIT Sloan School of Management
2000- Principal Research Scientist, the MIT Sloan School of Management

Main Works:
- M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam, "Negotiating Complex Contracts," Group Decision and Negotiation J., Vol.12, No.2, 2003.
- K. Fujita, T. Ito, and M. Klein, "A Secure and Fair Protocol that Addresses Weaknesses of the Nash Bargaining Solution in Nonlinear Negotiation," Group Decision and Negotiation, Springer, 10.1007/s10726-010-9194-6, 2010.

Membership in Academic Societies:
- The Institute of Electrical and Electronics Engineers (IEEE)
- The Association for Computing Machinery (ACM)
- Association for the Advancement of Artificial Intelligence (AAAI)

Representative based Multi-Round Protocol based on Revealed Private Information for Multi-issue Negotiations

Katsuhide Fujita* Takayuki Ito† Mark Klein‡

Abstract

Multi-issue negotiation protocols represent a promising field since most negotiation problems in the real world involve multiple issues. Our work focuses on negotiation with interdependent issues in which agent utility functions are nonlinear. Existing works have not yet focused on the private information of agents. In addition, they were not scalable in the sense that they have shown a high failure rate for making agreements among five or more agents. In this paper, we focus on a novel multi-round representative based protocol that utilizes the amount of revealed agents' private information. Experimental results demonstrate that our mechanism reduces the failure rate in making agreements and is scalable for the number of agents compared with existing approaches.

Keywords: Multi-issue Negotiation, Nonlinear Function

1 Introduction

We envision a future in which large numbers of participants collaborate, negotiate, and reach consensuses through computer-supported negotiation support systems for global problems. Collaboratorium [1, 2], one such pioneering work that enables many people to participate in an argument on such worldwide problems as global warming, provides a platform for coordinating large-scale arguments through web based collaboration tools. In our research, we consider a tool that supports large-scale negotiations among the world's people. In such

*School of Techno-Business Administration Nagoya Institute of Technology/ Visiting Student, Center for Collective Intelligence, Sloan School of Management, Massachusetts Institute of Technology, Gokiso, Showa-ku, Nagoya, Aichi, 466-8555 JAPAN, Tel:052-735-7968, Fax:052-735-7404, E-mail:fujita@itolab.mta.nitech.ac.jp

†Visiting Scholar, Center for Collective Intelligence, Sloan School of Management, Massachusetts Institute of Technology/ School of Techno-Business Administration, Department of Computer Science and Engineering, Nagoya Institute of Technology/ Researcher, PREST, Japan Science and Technology Agency (JST).

‡Center for Collective Intelligence MIT Sloan School of Management

a situation, eliciting people's utility spaces and automatically finding and suggesting possible agreements would be valuable. People could reach agreements based on such system suggestions.

Multi-issue negotiation protocols represent an important field of study. Even though much previous work exists in this area [3, 4, 5], most of it deals exclusively with simple negotiations involving independent multiple issues. These studies of negotiations mainly assume that agents have an incentive to cooperate to achieve win-win agreements because the situation is not a zero-sum game. Many real-world negotiation problems, however, are complex and involve interdependent multiple issues. Thus, we focus on complex negotiation with interdependent multiple issues.

The bidding based negotiation protocol offers high performance on multi interdependent issues negotiation. However, it has two main issues. **1) Privacy:** Existing works have not yet addressed agents' private information, which should not be revealed excessively because agents who reveal too much utility information suffer a disadvantage. For example, suppose that several companies are collaboratively designing and developing a new car model. If one company reveals more utility information than the other companies, those other companies can learn more of that company's utility information. As a result, the company will face a disadvantage in subsequent negotiations. Furthermore, explicitly revealing utility information is dangerous from a security standpoint. **2) Scalability for the number of agents:** The bidding based negotiation protocol does not have high scalability for the number of agents. In the bidding based negotiation protocol, the mediator needs to find the optimum combination of submitted bids from the agents. However, the computational complexity for finding solutions is too large. The number of agent bids was limited in existing work [6]. Limiting bids causes low optimality and high failure rate for agreements.

To resolve privacy issues, we define an agent's **revealed area** that represents the amount of his/her revealed utility space. This revealed area numerically defines which agents are cooperative and which are not. Additionally, the mediator can understand how much of the agent's private information has been revealed in the negotiation.

Moreover, we propose a **representative based protocol** that has high scalability for the number of agents and considers the agent's private information. In our protocol, we first select representatives who revealed more of their utility space than the others. These representatives reached an agreement on alternatives and proposed them to the other agents. Finally, the other agents can express their own intentions concerning agreement or disagreement. In this protocol, agents who revealed more private utility information can have a greater chance to be representatives who will attend to reach an agreement on behalf of the other agents. Although agents tend to avoid revealing their own private information, they have an incentive to reveal it to be representatives.

The representative based protocol has been inspired by the parliamentary systems in England, Canada, Australia, Japan, etc. in which representatives are making an agreement on behalf of other people. In a situation in which many

people have to reach an agreement, directly reflecting all members' opinions is quite difficult. Doing so requires much time and energy and is not scalable. Although voting is one option, voting might have paradoxical results [7].

We expand our mechanism to be multi-round by using the Threshold Adjustment Protocol [8]. The multi-round mechanism improves the failure rates and achieves fairness in terms of the revealed area. This means that the amounts of the revealed areas are almost the same among agents. Further, a representative mechanism can prevent the unfair solutions that can exist in the original Threshold Adjustment Protocol.

The representative based protocol drastically reduces computational complexity because only representative agents try to reach a consensus. The experimental results demonstrate that our protocol reduces the failure rate in making agreements and that it is scalable on the number of agents compared with existing approaches. We also demonstrate that our protocol reduces the revealed area compared with existing works. Furthermore, we investigate the detailed effect of the representative selection method in our protocol and call the selection method RAS in which agents who reveal a larger utility area are selected as representatives. In the experiments, we compare RAS with a selection method in which representative agents are randomly selected (RANDOM).

The remainder of this paper is organized as follows. First, we describe a model of non-linear multi-issue negotiation and an existing work's [6] problems. Second, we define the revealed area and proposed our new negotiation mechanism. Third, we describe the multi-round negotiation protocol. Fourth, we present an experimental assessment of this protocol. Finally, we describe related work and draw conclusions.

2 Negotiation Using Complex Utility Space

2.1 Complex Utility Model

We consider the situation where n agents want to reach an agreement. m issues, $s_j \in S$ must be negotiated. The number of issues represents the number of dimensions of the utility space. For example, if there are three issues, the utility space has three dimensions. The issues are not "distributed" over agents who are all negotiating a contract that has N (e.g., 10) issues. All agents are potentially interested in the values for all N issues. Issue s_j has a value drawn from the domain of integers $[0, X]$, *i.e.*, $s_j \in [0, X]$. A discrete domain can come arbitrarily close to a real domain by increasing the domain size. As a practical matter, 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, although the bid generation step must be modified to use a nonlinear optimization algorithm suited to real domains. 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

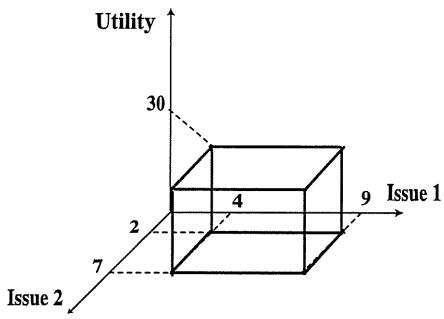


Figure 1: Example of a constraint

l constraints, $c_k \in C$. Each constraint represents a region with one or more dimensions and has an associated utility value. 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 30 and holds if the value for issue 1 is in the range $[4, 9]$ and the value for issue 2 is in the range $[2, 7]$. Every agent has its own, typically unique, set of constraints.

An agent's utility for 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 hyperplanes with a single optimum. Figure 2 shows an example of a nonlinear utility space. There are two issues, *i.e.*, two dimensions, with domains $[0, 99]$. There are 50 unary constraints (*i.e.*, that relate to one issue) as well as 100 binary constraints (*i.e.*, that interrelate to two issues). The utility space is highly nonlinear with many hills and valleys.

In our utility function, we assume interdependency between the issues. For example, since an agent has a binary constraint between issues 1 and 2, as Figure 1 shows, they are interdependent for the agent. Therefore, our utility space is highly interdependent.

As is common in negotiation contexts, we assume that agents do not share their utility functions with each other to preserve a competitive edge. In fact, generally agents do not completely know their desired contracts in advance because their 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, which is too many to evaluate exhaustively. Agents must thus operate in highly uncertain environments.

Finding an optimal contract for individual agents with such utility spaces can be handled using such well-known nonlinear optimization techniques as sim-

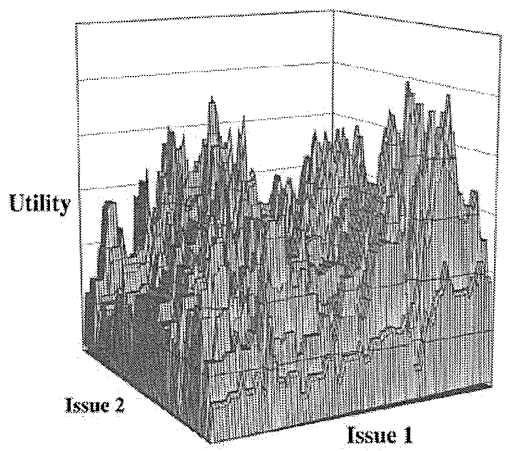


Figure 2: Complex utility space for single agent

ulated 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:

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

In other words, our protocol 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 Existing Bidding based Protocol

In a previous work [6], agents reach an agreement based on the following steps. This is called a **basic bidding based mechanism**.

[Generate bids] Each agent samples its utility space to find high-utility contract regions. A fixed number of samples are taken from a range of random points, drawn 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.

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 [9] to find the local optimum in its neighborhood. Figure 3 exemplifies this concept. Black dots are sampling points and white dots are locally optimal contract points.

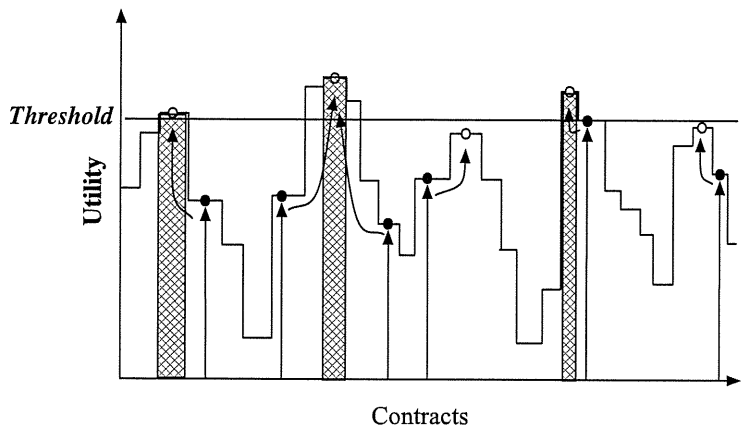


Figure 3: Generating bids

For each contract $\vec{s}$ found by adjusted sampling, an agent evaluates its utility by a summation of the values of satisfied constraints. If that utility is larger than reservation value δ **(threshold)**, then the agent defines a bid that covers all the contracts in the region with that utility value. This is easy: the agent merely finds the intersection of all the constraints satisfied by that $\vec{s}$.

[Find the Solutions] In negotiation, the mediator takes the middle position and 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 (Figure 4) [1]. If there is more than one such overlap, the mediator selects the one with the highest summed bid value (and assuming truthful bidding, the highest social welfare).

2.3 Scalability and Privacy Problems

Since it is a combinatorial optimization calculation, computational complexity for finding solutions exponentially increases based on the number of bids. For example, if there are 10 agents and each agent has 20 bids, the number of bids is 20^{10}. To make our negotiation mechanism scalable, the computational complexity must be reduced to find solutions.

We limited the number of bids for each agent to handle the computational complexity in the basic bidding based protocol [6]. The concrete number of bids in this limitation was $\sqrt[N]{6,400,000}$, a number that reflects our experimental calibration in 2005. But even though CPUs are faster now, the limitation

[1] A bid has an acceptable region. For example, if a bid has regions [0,2] for issue 1 and [3,5] for issue 2, the bid is accepted by a contract point (1,4), which means that issue 1 takes 1 and issue 2 takes 4. If a combination of bids, i.e., a solution, is consistent, definitely overlapping regions exist. For instance, a bid with regions (issue 1, issue 2) = ([0,2],[3,5]), and another bid with ([0,1],[2,4]) is consistent.

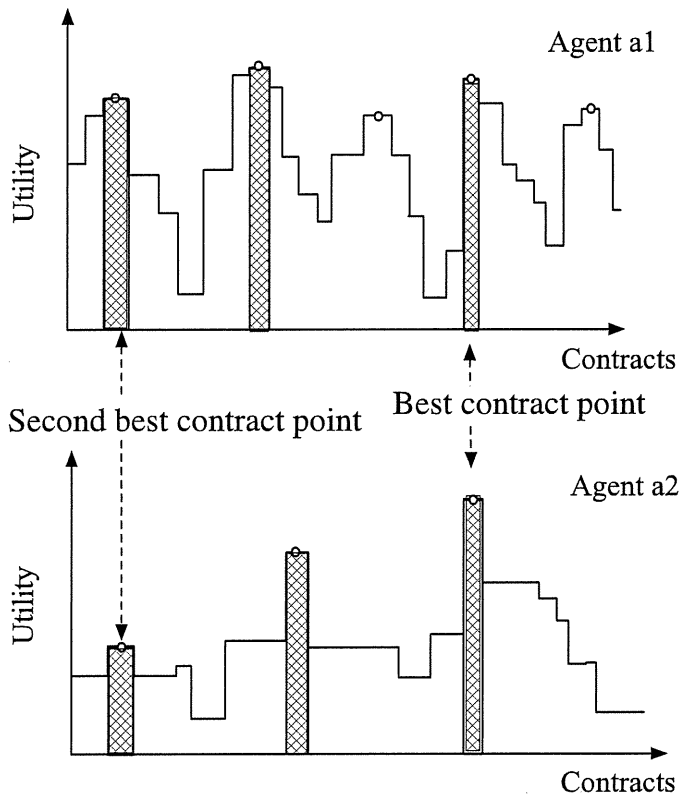


Figure 4: Find solutions

number does not differ so much because this is an exponential problem. Table 1 shows the limitation numbers of bids in one agent. This number quickly drops by increasing the total number of agents. Because of the limitation of bids, the failure rate for finding agreements quickly increases along with increasing the number of agents. When the number of agents is five and the number of issues is seven, we experimentally observed that the failure rate is around 40%. In fact, a strong trade-off exists between increasing the number of total bids and finding good quality solutions. Increasing the number of total bids is not an effective approach for finding good quality agreements.

Thus, it is necessary to build another mechanism that will find higher quality solutions without limiting the bids. Our mechanism proposed in this paper is highly scalable. The other issue with existing protocols is that they are not concerned with privacy in utility spaces. Even in a collaborative situation among people, it is normal to keep one's own utility space closed as long as one is not asked to do otherwise. Our new mechanism achieves such a situation by defining

Num. of agents	Limit of bids	Num. of agents	Limit of bids
2	2530	7	9
3	186	8	7
4	50	9	6
5	23	10	5
6	13		

Table 1: Limitation of the bids

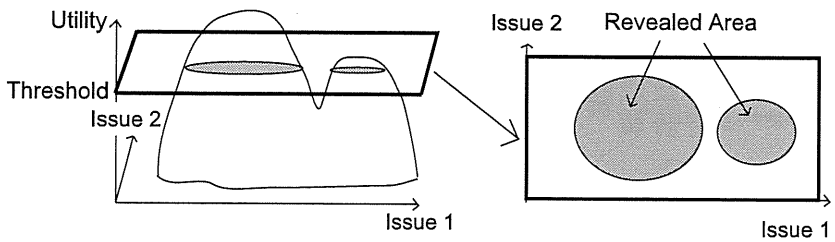


Figure 5: Revealed area

the **revealed area** in utility spaces.

3 Multi-Round Representative based Protocol based on Revealed Private Information

3.1 Revealed Area for Agent

We focus on the amount of private information agents revealed in the negotiation. We employ **revealed area** as a measure of the amount of revealed utility space. Figure 5 shows an intuitive example of a revealed area, defined as an agent's possible contract points that are revealed in his utility space on his threshold.

For an agent, it is important for him/her to know how much his/her private information is revealed compared with the other agents. The mediator can judge whether an agent is cooperative based on the amount of revealed private information.

We use a **threshold** that is employed in generating bids as a measure of adjusting agents' revealed areas. Since directly adjusting the revealed area is difficult because agents have complex utility spaces, we consider adjusting their threshold to adjust their revealed areas. The threshold is employed for an agent to generate his/her bids based on utility values above the threshold. The threshold was originally adopted to adjust the number of bids. However, in this paper, we also utilize it for determining an agent's revealed area while handling complex utility space.

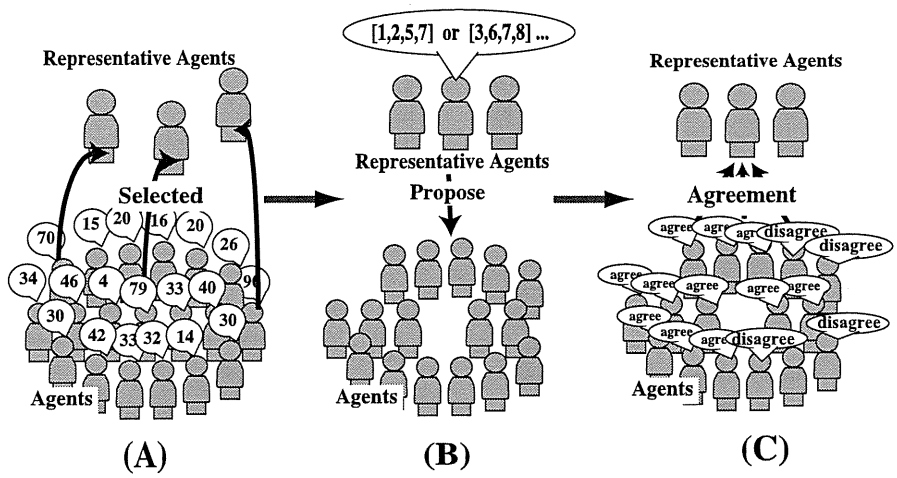


Figure 6: Representative based protocol

3.2 Representative based Protocol

Representative-based protocol consists of three steps. The first step is to select the representative agents (Step1). The second step is to find solutions, and propose them to the other agents (Step2). The third step is to respond to the agreement by the other agents (Step3).

We assume each agent uses a reservation value to determine whether to "agree" or "disagree" with the representative agents. Actually, for practical applications, the reservation value can be determined by a human user. In addition, we assume that the number of representatives is static in representative based protocol. This protocol consists of the following steps.

[Step 1: Selection of Representative Agents] Representative agents are selected based on the amount of their revealed areas, as shown in Figure 6 (A). First, each agent submits how much he can reveal his utility space to the mediator. Namely, each agent submits the numeric value of the amount of his possible revealed area. The mediator selects the representative agents who revealed a large area. We call this selection method RAS. This step is main additional coordination processes by the use of representatives. By employing RAS, all agents are satisfied with the mediator's decision because it is the best method for all agents to find optimal solutions.

[Step 2: Proposing by Representatives] Representative agents find solutions and propose them to other agents, as shown in Figure 6 (B). First, representative agents find solutions by employing a breadth-first search with branch cutting to find solutions (from lines 3 to 14 in representative_protocol()).

Next, the representative agents ask the other agents whether they "agree" or "disagree." Step 2 is repeated until all the other agents agree or the solutions

found by the representatives are rejected by the other agents.
[Step 3: Respond to Agreement by other Agents] First, the other agents receive the solutions from the representative agents. Then they judge whether they "agree" or "disagree" by determining whether the solution's utility is higher than their own reservation value (Figure 6 (C).).

Steps 1, 2, and 3 are captured as Algorithms 1 and 2:

Algorithm 1 representative_protocol(B)

B: A set of bid-set of each agent
(B=$\{B_0, B_1, ..., B_n\}$, a set of bids from agent i is $B_i = \{b_{i,0}, b_{i,1}, ..., b_{i,m_i}\}$)
RB:A set of bid-set of each representative agent
(RB=$\{RB_0, RB_1, ..., RB_m\}$, a set of bids from representative agent i is $RB_i = \{rb_{i,0}, rb_{i,1}, ..., rb_{i,l_i}\}$)
SC: A set of solution-set of each representative agent
(SC=$\{SC_0, SC_1, ..., SC_n\}$, a set of bids from agent i is $SC_i = \{sc_{i,0}, sc_{i,1}, ..., sc_{i,m_i}\}$)

```
RB := select_representative(B)
SC := RB_0, i := 1
while i < the number of representative agents do
   SC' := ∅
   for s ∈ SC do
      for rb_{i,j} ∈ RB_i do
         s' := s ∪ rb_{i,j}
      end for
   end for
   if s' is consistent then
      SC' := SC' ∪ s'
   end if
   SC := SC', i := i + 1
end while
while i < |SC| do
   if ask_agent(SC_i) is true & SC_i Utility is maximum then
      return SC_i
   else
      return No Solution
   end if
end while
```

This protocol is scalable for the number of agents. In a representative protocol, combinatorial optimization only occurs in negotiation among representative agents. In fact, the computational complexity for proposing solutions to unrepresentative agents only increases linearly and is almost negligible. Thus, the computational complexity is drastically reduced compared with the existing mechanism.

Finally, we describe the trade-off for an agent between revealing a large amount of utility space and being a representative agent. Representative agents have advantages since they can propose alternatives to other agents and dis-

Algorithm 2 ask_agent(SC)

select_representative() is a method for performing Step 1
Th: A reservation value of each agent (Th=$\{Th_0, Th_1, ..., Th_n\}$)

```
1: while i < the number of agents do
2:   if SC'sUtility < Th_i then
3:     return false
4:   else
5:     i := i + 1
6:   end if
7: end while
8: return true
```

advantages because they need to reveal larger utility space. Unrepresentative agents have advantages in keeping their utility hidden and disadvantages in responding based on representatives' agreements.

3.3 Threshold Adjusting Mechanism

We extend our protocol to multi-round negotiation based on the threshold adjusting method [8] so that the number of times to be a representative agent is fair. The total amount of revealed utility space for each agent is almost the same by the threshold adjustment mechanism.

The main idea of the threshold adjusting mechanism is simple: if an agent reveals a larger area of his utility space, he should gain an advantage. On the other hand, an agent who reveals a smaller area of his utility space should adjust his threshold to agree with others. The threshold values are changed by each agent based on the amount of revealed area. If the agent decreases the threshold value, this means that he must reveal more of his utility space.

This mechanism is repeated until an agreement is achieved or all agents refuse to lower their thresholds. Agents can decide whether to lower the threshold based on their reservation value, i.e., the minimum threshold. This means that agents have the right to reject the request to decrease their threshold if the request decreases a threshold lower than the reservation value.

Figure 7 shows an example of the threshold adjusting process among three agents. The upper and bottom figures show the thresholds and the revealed areas before and after adjusting the threshold, respectively. In particular, in this case, agent 3 revealed a small amount of his utility space. The amount of agent 3's revealed utility space in this threshold adjustment is the largest among these three agents. The exact rate of the amount of revealed utility space and the amount of decreased threshold are defined by the mediator or the mechanism designer.

The threshold adjusting mechanism is shown as Algorithm 3:

In the threshold adjusting mechanism, agents can consider others' behaviors by adjusting the agent thresholds. In our definition, agents can reveal more

Algorithm 3 threshold_adjustment( )

Ar: Area Range of each agent ($Ar = \{Ar_0, Ar_1, ..., Ar_n\}$)
representative_protocol(): representative based protocol explained in previous section.

```
1: loop
2:   i := 1, B := ∅
3:   while i < |Ag| do
4:     bid_generation_with_SA(Th_i,V,SN,T,B_i)
5:   end while
6:   maxSolution := representative_protocol(B)
7:   if find maxSolution then
8:     break loop
9:   else if all agent can lower the threshold then
10:    i := 1
11:    SumAr := Σ_{i∈|Ag|} Ar_i
12:    while i < |Ag| do
13:      Th_i := Th_i − C ∗ (SumAr − Ar_i)/SumAr
14:      i := i + 1
15:    end while
16:  else
17:    break loop
18:  end if
19: end loop
20: return maxSolution
```

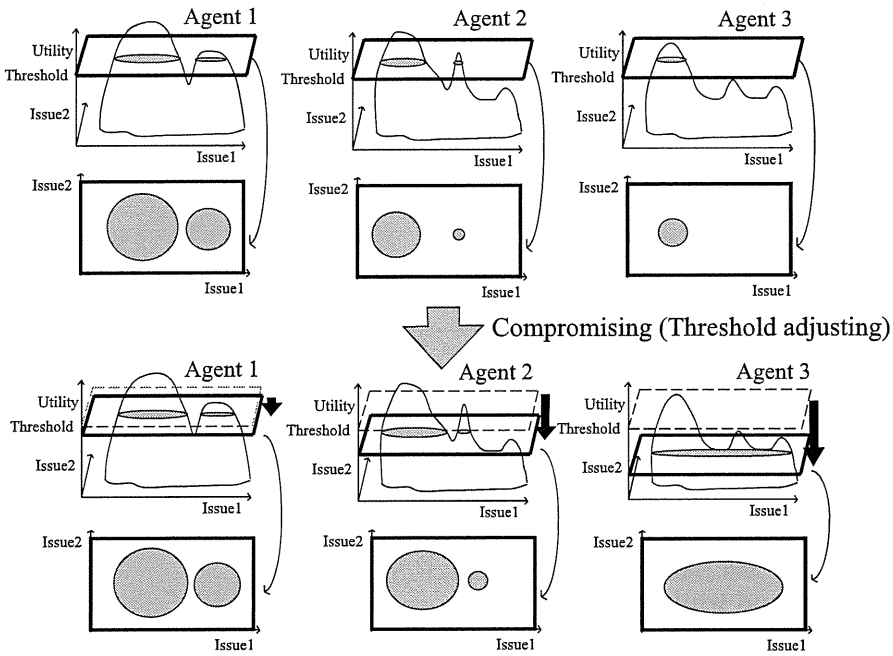


Figure 7: Threshold adjusting process

revealed area if they greatly lower their threshold. Additionally, the width of decreasing the threshold is decided based on a comparison of the others' revealed areas in the threshold adjusting mechanism. Therefore, they can take the behaviors of others into consideration in multi-round negotiation.

4 Experiment Results

4.1 Experiment Settings

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.

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 measurement is the range of the revealed areas. If an agent reveals one point on the grid of the utility space, he loses 1 privacy unit. If he reveals 1000 points, then he loses 1000 privacy units.

We also analyze the representative selection method in our protocol. The representative selection method remains an important research point. The selection method in which agents who reveal a larger utility area are selected as representatives is called (**RAS**), and the random selection method in which representatives are randomly selected is called (**RANDOM**). To investigate the detailed effects of RAS, we assume RANDOM is the general basis for comparison.

The following are the parameters for our experiments:

- Domain for issue values: $[0, 9]$.
- Constraints: 10 unary constraints, 5 binary constraints, 5 trinary constraints, etc. (a unary constraint relates to one issue, a binary constraint relates to two issues, and so on).
- Maximum constraint value: $100 \times (number\ of\ issues)$. Constraints that satisfy many issues have on average larger weights. This seems reasonable for many domains. To schedule meetings, for example, higher order constraints concern more people than lower order constraints, so they are more important.
- Maximum constraint width: 7. The following constraints, therefore, are all valid: issue $1 = [2, 6]$, issue $3 = [2, 9]$ and issue $7 = [1, 3]$.
- Number of samples taken during random sampling: $(number\ of\ issues) \times 200$.
- Annealing schedule for sample adjustment: initial temperature 30, 30 iterations. Note that the annealer must 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.
- Threshold used by agents to select what to bid 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 excise contract points with low utility.
- Amount of threshold is decreased by $100 \times (SumAr - Ar_i)/SumAr$. $SumAr$ means the sum of all agents' revealed areas. Ar_i means agent i's revealed area.
- Limitation on number of bids per agent: $\sqrt[N]{6,400,000}$ for N agents. It was only practical to run the deal identification algorithm if it explored no more than about 6,400,000 bid combinations, which implies a limit of $\sqrt[N]{6,400,000}$ bids per agent for N agents.