Journal of Information Processing Vol.23 No.2 176–184 (Mar. 2015)

[DOI: 10.2197/ipsjjip.23.176]

Regular Paper

Low-Complexity Exploration in Utility Hypergraphs

Rafik Hadfi[1,a)] Takayuki Ito[1,b)]

Received: June 30, 2014, Accepted: September 12, 2014

Abstract: A novel representation for nonlinear utility spaces is provided, by adopting a modular decomposition of the issues and the constraints. The idea is that constraint-based utility spaces are nonlinear with respect to issues, but linear with respect to the constraints. The result is a mapping from a utility space into an issue-constraint hypergraph. Exploring the utility space is therefore reduced to a message passing mechanism along the hyperedges by means of utility propagation. The optimal contracts are efficiently found using a variation of the Max-Sum algorithm. Particularly, we use a power-law heuristic that lowers the search cost when exploring the utility hypergraph. We experimentally evaluate the model using parameterized random nonlinear utility spaces, showing that it can handle a large family of complex utility spaces using several exploration strategies. The complexity of the generated utility spaces is evaluated using the information theoretic notion of entropy. The optimal search strategy allows a better scaling of the model for complex utility spaces.

Keywords: interdependent issues, nonlinear utility, hyper-graph, Max-Sum, entropy

1. Introduction

Automated negotiation is a mechanism for consensus building among distributed decision makers. In this sense, it can efficiently incorporate both human and artificial decision makers to solve a wide range of complex problems. Its applications range from coordination and cooperation [9], [13] to task allocation [10], [11] and decentralized information services [12]. Most of the realistic negotiation scenarios are characterized by multiple and interdependent issues, which yields complex preferential structures, or precisely, utility spaces [8]. As the search space and the complexity of the problem grow, finding an optimal solution becomes intractable for one single agent. Similarly, in the case where the problem is distributed, reaching an agreement between a group of agents becomes harder.

In this paper, we propose to tackle the complexity of utility spaces used in multi-issue negotiation by rethinking the way they are represented. We think that adopting the adequate representation gives a solid ground to tackle the scaling problem. We address this problem by adopting a representation that allows a modular decomposition of the issues-constraints given the intuition that constraint-based utility spaces are nonlinear with respect to issues, but linear with respect to the constraints. This allows us to rigorously map the utility space into an issues-constraints hypergraph with the underling interdependencies. Exploring the utility space reduces then to a message passing mechanism along the hyperedges by means of utility propagation.

Adopting a graphical representation while reasoning about utilities is not new in the multi-issue negotiation literature. In fact, the idea of utility graphs could potentially help decomposing highly nonlinear utility functions into sub-utilities of clusters of inter-related items, as in Refs. [2] or [1]. Similarly, [20] used utility graphs for preferences elicitation and negotiation over binary-valued issues. Reference [17] adopts a weighted undirected graph representation of the constraint-based utility space. Particularly, a message passing algorithm is used to find the highest utility bids by finding the set of unconnected nodes which maximizes the sum of the nodes' weight. However, restricting the graph and the message passing process to constraints' nodes does not allow the representation to be descriptive enough to exploit any potential hierarchical structure of the utility space through a quantitative evaluation of the interdependencies between both issues and constraints. In Ref. [4], issues' interdependency are captured by means of similar undirected weighted graphs where a node represents an issue. This representation is restricted to binary interdependencies while real negotiation scenarios involve "bundles" of interdependent issues under one or more specific constraints. In our approach, we do not restrict the interdependency to lower-order constraints but we allow $p-$ary interdependencies to be defined as an hyperedge connecting p issues.

Adopting such graphical representation with its underlying utility propagation mechanism comes from the intuition that negotiation, after all, is a cognitive process that involves concepts and associations, performed by supposedly bounded rational agents [15]. And while bearing in mind the fact that cognitive processes perform some form of Bayesian inference [14], we chose to adopt a graphical representation that serves more as an adequate framework for any preference-based space. The advantage of using this representation is its scalability in the sense that the problem becomes harder for a large number of issues and constraints. But if we can decompose the utility space, we can exploit it more efficiently. Another way to look at this "con-

1 Department of Computer Science and Engineering, Nagoya Institute of Technology. Gokiso, Showa-ku, Nagoya 466–8555, Japan
a) rafik@itolab.nitech.ac.jp
b) ito.takayuki@itolab.nitech.ac.jp

nectionist" representation is that it can be clustered in ways that can isolate interdependent components, thus, allowing them to be treated separately and even negotiated independently form the rest of the problem.

Another motivation behind the hypergraph representation is that it allows a layered, hierarchical view of any given negotiation scenario. Given such architecture, it is possible to recursively negotiate over the different layers of the problem according to a top-down approach. Even the idea of issue could be abstracted to include an encapsulation of sub-issues, located in sub-utility spaces and represented by cliques in the hypergraph. Consequently, search processes can help identify optimal contracts for improvement at each level. This combination of separating the system into layers, then using utility propagation to focus attention and search within a constrained region can be very powerful in the bidding process. A similar idea of recursion in the exploration of utility space was introduced by Ref. [18] although it is region-oriented and does not adopt a graphical representation of the utility space. We experimentally evaluated our model using parametrized and random nonlinear utility spaces, showing that it can handle large and complex spaces and outperforms previous sampling approaches. The adopted model was also evaluated in terms of the complexity of a family of utility spaces.

Overall, the contribution of the paper could be summarized as following.

- A better representation for nonlinear utility spaces to tackle the complexity problem. It has the merit of being modular and rich, which allows several search strategies to be tested as well as any graph-theoretic analysis.
- An efficient algorithm for optimal contracts search based on message passing. The algorithm outperforms all the other sampling-based methods and provides a better scaling.
- Quantitative assessment of the complexity of nonlinear utility spaces using entropy [5], and how it affects search.
- Identification of several search methods as well as the optimal strategy that minimizes the search cost. The optimal search strategy is a novel and efficient heuristic for *loopy utility propagation* inspired by the loopy belief propagation for probabilistic graphical models [19].

The paper is organized as following. In the next section, we propose the basics of our new nonlinear utility space representation. In Section 3, we describe the contracts search mechanisms. In Section 4, we provide the complexity study and the optimal strategy. In Section 5, we provide some experimental results. In Section 6, we conclude and outline the future work.

2. Nonlinear Utility Space Representation

2.1 Problem Formulation

We start from the formulation of nonlinear multi-issue utility spaces used in Ref. [7]. That is, an n−dimensional utility space is defined over a finite set of issues $\mathbb{I} = \{i_1, \ldots, i_k, \ldots, i_n\}$. The issue k, namely i_k, takes its values from a finite set $\mathbb{I}_k$ with $\mathbb{I}_k \subset \mathbb{Z}$. A contract $\vec{c}$ is a vector of issue-values with $\vec{c} \in \mathcal{I}$ and $\mathcal{I} = \times_{k=1}^{n} \mathbb{I}_k$.

An agent's utility function is defined in terms of m constraints, making the utility space a constraint-based utility space. That is, a constraint c_j is a region of the total n−dimensional utility space.

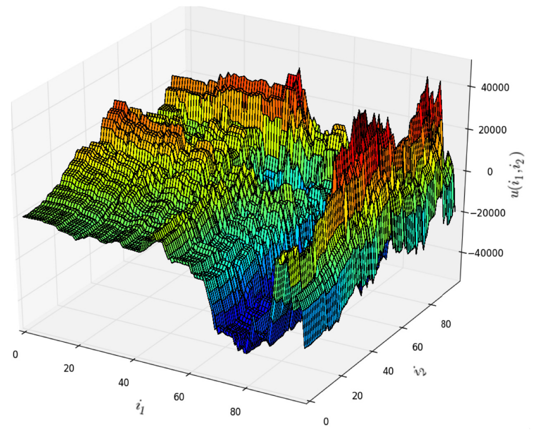


Fig. 1 2−dimensional nonlinear utility space.

We say that the constraint c_j has a weight, or value $w(c_j, \vec{c})$ for contract $\vec{c}$ if c_j is satisfied by $\vec{c}$. That is, when the contract point $\vec{c}$ falls within the hyper-volume defined by constraint c_j, namely $hyp(c_j)$. The utility of an agent for $\vec{c}$ is thus defined as in (1).

$$u(\vec{c}) = \sum_{c_{j\in[1,m]},\ \vec{c}\,\in\, hyp(c_j)} w(c_j, \vec{c}) \tag{1}$$

In the following, we distinguish three types of constraints: Cubic, Bell and Plane constraints. Constraint-based utility spaces are a practical way to represent non-monotonic and nonlinear utility functions [16], [17], [18]. The representation Eq. (1) produces a "bumpy" nonlinear utility space with high points whenever many constraints are satisfied and lower points where few or no constraints are satisfied. For instance, **Fig. 1** shows an example of nonlinear utility space for issues i_1 and i_2 taking values in $\mathbb{I}_1 = \mathbb{I}_2 = [0, 100]$, with $m = 600$ constraints having types in $\theta = \{Cube,\ Bell,\ Plane\}$.

2.2 New Representation

The utility function Eq. (1) is nonlinear in the sense that the utility does not have a linear expression against the contract [7]. This is true to the extent that the linearity is evaluated with regard to the contract $\vec{c}$. However, from the same expression Eq. (1) we can say that the utility is in fact linear, but in terms of the constraints $\{c_1, \ldots, c_j, \ldots, c_m\}$. The utility space is therefore decomposable according to these constraints. This yields a modular representation of the interactions between the issues and how they locally relate to each other. In fact, $hyp(c_j)$ reflects the idea that the underlying contracts are governed by the bounds defined by c_j once the contracts are projected according to their issues' components. In this case, the interdependence is not between issues but between constraints. For instance, two constraints c_1 and c_2 can have in common one issue i_k taking values respectively from an interval $\mathbb{I}_{k,c_1}$ if it is involved in c_1, and values in $\mathbb{I}_{k,c_2}$ if it is involved in c_2, with $\mathbb{I}_{k,c_1} \neq \mathbb{I}_{k,c_2}$. Finding the value that maximizes the utility of i_k while satisfying both constraints becomes harder due to fact that changing the value of i_k in c_1 changes the instance of i_k in c_2 in a cyclic manner. This nonlinearity gets worse with an increasing number of issues, domains' sizes, and the non-monotonicity of the constraints.

Next, we propose to transform Eq. (1) into a modular, graphical representation. Since one constraint can involve one or more multiple issues, we will adopt a hypergraph representation.

2.3 Utility Hypergraph

We assign to each constraint $c_{j\in[1,m]}$, a factor Φ_j, with $\Phi = \{\Phi_j\}_{j=1}^m$. We define the hypergraph G as in Eq. (2).

$$G = (\mathbb{I}, \Phi) \tag{2}$$

Nodes in $\mathbb{I}$ define the issues and the hyperedges in Φ are the factors (constraints). To each factor Φ_j we assign a neighbors' set $\mathcal{N}(\Phi_j) \subset \mathbb{I}$ containing the issues connected to Φ_j (involved in c_j), with $|\mathcal{N}(\Phi_j)| = \varphi_j$. In case $\varphi_j = 2\ \forall j \in [1, m]$, the whole problem collapses to a constraints satisfaction problem in a standard graph. To each factor Φ_j corresponds a φ_j−dimensional matrix, $\mathcal{M}_{\Phi_j}$, where the jth dimension is the discrete interval $[a_k, b_k] = \mathbb{I}_k$, the domain of issue i_k. This matrix contains all the values that could be taken by the issues in $\mathcal{N}(\Phi_j)$. Each factor Φ_j has a function ϕ_j defined as a sub-utility function of the issues in $\mathcal{N}(\Phi_j)$, as in Eq. (3).

$$\begin{aligned}\phi_j :\ & \mathcal{N}(\Phi_j)^{\varphi_j} \to \mathbb{R} \\ & \phi_j(i_1, \dots, i_k, \dots, i_{\varphi_j}) \mapsto w(c_j, \vec{c})\end{aligned} \tag{3}$$

As we are dealing with discrete issues, Φ_j is defined by the matrix $\mathcal{M}_{\Phi_j}$. That is, $\phi_j(i_1, \dots i_k, \dots, i_{\varphi_j})$ is simply the $(1, \dots, k, \dots, \varphi_j)^{th}$ entry in $\mathcal{M}_{\Phi_j}$ corresponding as well to the value $w(c_j, \vec{c})$ mentioned in Eq. (1). It is possible to extend the discrete case to the continuous one by allowing continuous issue-values and defining Φ_j as a continuous function. Next, we give few examples about the model usage.

2.3.1 Example 1.

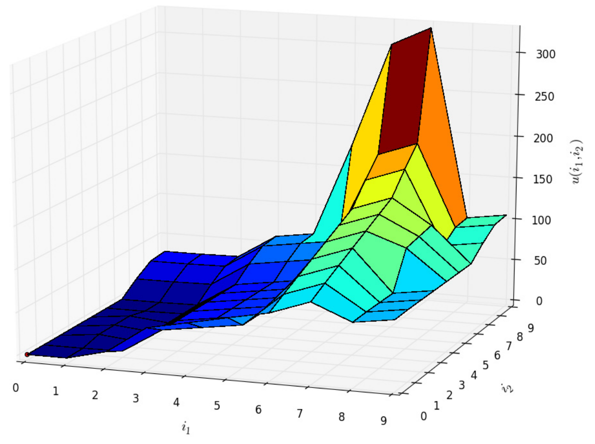
Figure 2 illustrates a 2−dimensional utility space with its hypergraph G_2. The issues' domains are $\mathbb{I}_1 = \mathbb{I}_2 = [0, 9]$. G_2 consists of $m = 10$ constraints (red squares) where each constraint involves at most 2 issues (white circles). We note 3 cubic constraints $\{C_j\}_{j=0}^2$ and 7 plane constraints $\{P_j\}_{j=0}^6$ with parameters $\beta_j, \alpha_j \in [-100, 100]$ and $j \in \{P_0, \dots, P_6\}$.

2.3.2 Example 2.

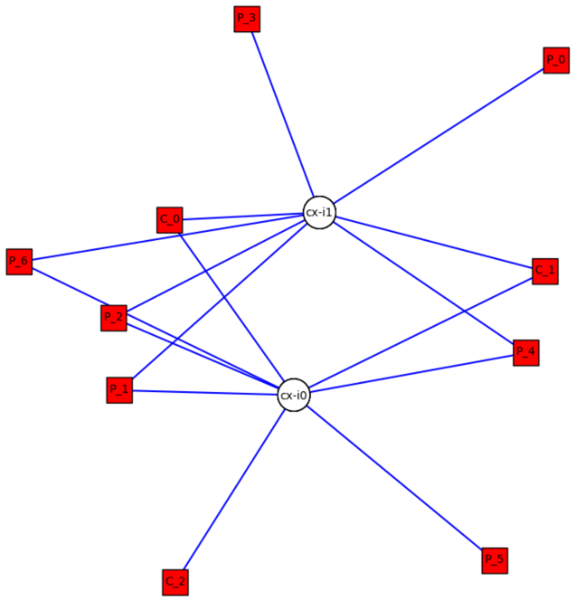
Consider the 10−dimensional utility space mapped into the hypergraph G_{10} defined as $G_{10} = (\mathbb{I}, \Phi)$ with $\mathbb{I} = \{i_k\}_{k=1}^9$ and $\Phi = \{\Phi_j\}_{j=1}^7$ as shown in **Fig. 3**. Each issue i_k has a set $\mathbb{I}_k = \bigcup_{v\in\mathcal{N}(k)} \mathbb{I}_{k,v}$ where $\mathbb{I}_{k,v}$ is an edge connecting i_k to its neighbor $v \subset \mathcal{N}(k) \in \Phi$. For example, $\mathbb{I}_1 = \bigcup_{v\in\{\Phi_1,\Phi_3,\Phi_6\}} \mathbb{I}_{1,v} = \{[5, 9], [3, 4], [3, 6]\}$. Constraints are defined by 3 types of geometrical shapes: cubic, plane or bell [16]. For instance, $\Phi_{1,2,3,4}$ could be cubic, $\Phi_{5,6}$ could be planar and Φ_7 could be a bell. Any combination is in fact possible, and depends only on the problem in hand and how it is being specified. To each constraint we assign a functional representation used to compute the utility of a contract if it satisfies the constraint by being located in the corresponding hyper-volume. For example, the utility function ϕ_j, defined in Eq. (3), corresponds to the functional definition of each constraint shown in Eq. (4).

$$\phi_j = \begin{cases} \mathit{Plane}: & \beta_j + \sum_{k=1}^{\varphi_j} \alpha_{j,k} \times v_k(i_k),\ (\beta_j, \alpha_{j,k}) \in \mathbb{Z}^2 \\ \mathit{Cube}: & v_j \\ \mathit{Bell}: & V_j \end{cases} \tag{4}$$

A plane constraint Φ_j will be defined using its φ_j−dimensional



(a) Utility space



(b) Utility hypergraph

Fig. 2 2−dimensional utility space and its hypergraph.

equation, while a cubic constraint will be assigned the value v_j in case the contract is in the cube. The computation of the utility V_j of a bell shaped constraint is performed as in Eq. (5). Herein, δ is the Euclidean distance from the center of the bell constraint to the contract point. Distances are normalized in $[-1, 1]$.

$$V_j = \begin{cases} \beta_j\,(1 - 2\delta^2) & \text{if} \quad \delta < 0.5 \quad \beta_j \in \mathbb{Z} \\ 2\,\beta_j\,(1 - \delta)^2 & \text{if} \quad \delta < 1 \quad \beta_j \in \mathbb{Z} \\ 0 & \text{else} \end{cases} \tag{5}$$

It is important to note that in our usage of the constraints, the agent is required to know the structure of the constraints through their functional definition. However, this is not always the case as the agent might face the situation where the only available assessment tool is a utility function deprived from its internal structure. In this case, we might think of a sampling method allowing us to construct an approximation of the agent's utility space. This could be done in a similar way to what *Markov Chain Monte Carlo* (*MCMC*) methods or a *Gibbs sampler* could perform in the case of Bayesian problems; especially, with the graphical nature of our representation. In the case were the agent knows her utility model, the hypergraph will be built through the composition

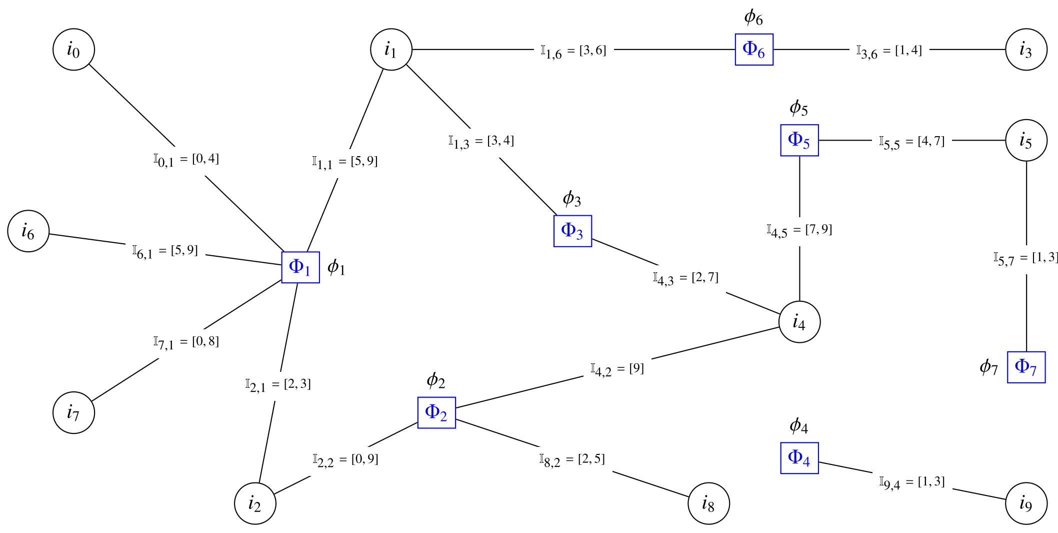

i_0
i_1
ϕ_6
Φ_6
i_3
$\mathbb{I}_{1,6} = [3,6]$
$\mathbb{I}_{3,6} = [1,4]$
ϕ_5
Φ_5
$\mathbb{I}_{5,5} = [4,7]$
i_5
$\mathbb{I}_{1,3} = [3,4]$
$\mathbb{I}_{0,1} = [0,4]$
$\mathbb{I}_{1,1} = [5,9]$
ϕ_3
Φ_3
$\mathbb{I}_{4,5} = [7,9]$
i_6
$\mathbb{I}_{6,1} = [5,9]$
Φ_1
ϕ_1
$\mathbb{I}_{4,3} = [2,7]$
$\mathbb{I}_{5,7} = [1,3]$
i_4
$\mathbb{I}_{7,1} = [0,8]$
$\mathbb{I}_{2,1} = [2,3]$
ϕ_2
Φ_2
$\mathbb{I}_{4,2} = [9]$
ϕ_7
Φ_7
i_7
ϕ_4
Φ_4
$\mathbb{I}_{2,2} = [0,9]$
$\mathbb{I}_{8,2} = [2,5]$
$\mathbb{I}_{9,4} = [1,3]$
i_2
i_8
i_9

Fig. 3 Issues-Constraints Hypergraph.

of several sub-utilities specified by the constraints. This could be done as to map different preferences' profiles the agent is expressing, like indifference, risk aversion, decreasing preference, and so on. Generally, and as it is the case in this paper, we are more interested in studying random parametric preferences, represented as Random Utility Hypergraph (RUH). Such representation allows any utility space model to be mapped into a hypergraphical representation that could make usage of the exploration strategies we are proposing. This includes linear, nonlinear and constraint-based utility models.

3. Optimal Contracts

The exploration of the utility hypergraph is inspired from the sum-product message passing algorithm for belief propagation [19]. However, the multiplicative algebra is changed into an additive algebra to support the utility accumulation. The messages circulating in the hypergraph are nothing other than the contracts we are attempting to optimize through utility maximization. Next, we develop the message passing (MP) mechanism operating on the issues and the constraints.

3.1 Message Passing

In the following, we consider the issues set $\mathbb{I}$ and a contract point $\vec{x} = (x_1, \ldots, x_i, \ldots, x_n) \in \mathcal{I}$. We want to find a contract $\vec{x^*}$ that maximizes the utility function defined in Eq. (1). Assuming that ϕ_j is the local utility of constraint Φ_j, we distinguish two types of messages: messages sent from issues towards constraints, and messages sent from constraints towards issues. The whole message passing process is an alternation of these two types of messages.

3.1.1 From Issue i_k to Constraint Φ_j.

As shown in Eq. (6), each message $\mu_{i_k \to \Phi_j}$ coming from i_k to Φ_j is the sum of the constraints' messages to i_k coming from constraints other than Φ_j.

$$\mu_{i_k \to \Phi_j}(i_k) = \sum_{\Phi_{j'} \in \mathcal{N}(i_k) \setminus \Phi_j} \mu_{\Phi_{j'} \to i_k}(i_k) \tag{6}$$

3.1.2 From Constraint Φ_j to Issue i_k.

Each constraint message is the sum of the messages coming from issues other than i_k, plus the constraint value $\phi_j(i_1, \ldots, i_k, \ldots, i_n)$, summed over all the possible values of the issues other than the issue i_k.

$$\mu_{\Phi_j \to i_k}(i_k) = \max_{i_1} \ldots \max_{i_{k'} \neq k} \ldots \max_{i_n} \Bigg[\phi_j(i_1, \ldots, i_k, \ldots, i_n) + \sum_{i_{k'} \in \mathcal{N}(\Phi_j) \setminus i_k} \mu_{i_{k'} \to \Phi_j}(i_k) \Bigg] \tag{7}$$

The mechanism starts from the leaves of the hypergraph, *i.e.*, the issues. At $t = 0$, the content of the initial messages is defined according to Eq. (8), $\phi'_j(i_k)$ being the partial evaluation of i_k in Φ_j.

$$\begin{aligned} \mu_{i_k \to \Phi_j}(i_k) &= 0 \\ \mu_{\Phi_j \to i_k}(i_k) &= \phi'_j(i_k) \end{aligned} \tag{8}$$

The partial evaluation $\phi'_j(i_k)$ of issue i_k in the factor Φ_j is the utility of i_k using Φ_j regardless of any other issue involved in Φ_j. For instance, for cubic and bell constraints, the evaluation is simply v_j and V_j $\forall k$ as in Eq. (4). If Φ_j is a plane constraint, the partial evaluation of i_k will be $\alpha_{j,k} \times v_k(i_k)$. In this manner, the factor Φ_j will get all the evaluations $\alpha_{j,k} \times v_k(i_k)$ from its surrounding issues (set $\mathcal{N}(\Phi_j)$) in order to yield the total utility Eq. (4) as a sum of the partial evaluations plus the plane constant β_j. Finally, the optimal contract $\vec{c^*}$ is found by collecting the optimal issues as in Eq. (9).

$$\vec{c^*} = \Bigg(\arg\max_{i_1} \sum_{\Phi_j \in \mathcal{N}(i_1)} \mu_{\Phi_j \to i_1}(i_1), \ldots, \arg\max_{i_k} \sum_{\Phi_j \in \mathcal{N}(i_k)} \mu_{\Phi_j \to i_k}(i_k), \ldots, \arg\max_{i_n} \sum_{\Phi_j \in \mathcal{N}(i_n)} \mu_{\Phi_j \to i_n}(i_n) \Bigg) \tag{9}$$

In a negotiation setting, it is more common that the agent requires a collection, or *bundle*, of the optimal contracts rather than one single optimum. In order to find such collection, we should endow Eq. (9) with a caching mechanism allowing each node in the hypergraph to store the messages that have been sent to it from the other nodes. That is, the cached messages will contain the summed-up utility values of the underlying node's instance. This is performed every time the operation *max* is called in Eq. (7) so that we can store the settings of the adjacent utility (and contract) that led to the maximum. Once ordered, such data structure allows us to generate an ordered bundle for the bidding process. In the next section, we algorithmically provide the MP mechanism.

3.2 Utility Propagation Algorithm

The main algorithm, Algorithm 1, operates on the hypergraph nodes by triggering the message passing process. Despite the fact that we have two types of nodes (issues and constraints), it is possible to treat them abstractly using $MsgPass$, in line 6. The resulting bundle is a collection of optimal contracts with utility greater or equal to the agent's reservation value rv. The message passing routine, $MsgPass$, is instantiated depending on the types of the source and destination nodes:

```
Algorithm: Utility Propagation
Input: $G = (\mathbb{I}, \Phi), rv, mode, \rho$
Output: Optimal contracts (bundle)
1  begin
2      for $i = 1 \to (\rho \times |\mathbb{I} \cup \Phi|)$ do
3          if $mode\ is\ Synchronous$ then
4              foreach $v_{src} \in \mathbb{I} \cup \Phi$ do
5                  foreach $v_{dest} \in v_{src}.Neighbors()$ do
6                      $v_{src}.MsgPass(v_{dest})$
7          else if $mode\ is\ Asynchronous$ then
8              $v_{src},\ v_{dest} \leftarrow rand_2([1, |V|]),\ v_{dest} \neq v_{src}$
9              $v_{src}.MsgPass(v_{dest})$
10     $bundle \leftarrow \emptyset$
11     foreach $i \in \mathbb{I}$ do
12         $bundle[i] \leftarrow \emptyset$
13         $\iota \leftarrow \cup_{j \in i.instances()}[j.min,\ j.max]$
14         $\mu^* \leftarrow k^* \leftarrow -\infty$
15         $\mu \leftarrow i.getmax()$
16         foreach $k = 1 \to |\mu|$ do
17             if $\mu^* < \mu[k]$ then
18                 $\mu^* \leftarrow \mu[k]$
19                 $k^* \leftarrow k$
20                 if $\mu^* \geq rv$ then
21                     $bundle[i] \leftarrow bundle[i] \cup \iota[k^*]$
22     return $bundle$
```

Algorithm 1: Main Algorithm.

From issue to constraint. The issue's message to a factor (or constraint) is the element-wise sum of all the incoming messages from other factors.

From constraint to issue. The factor's message to a targeted issue is done by recursively enumerating over all variables that the factor references Eq. (7), except the targeted issue. This needs to be performed for each value of the target variable in order to compute the message. If all issues are assigned, the values of the factor and of all other incoming messages are determined, so that their sum term is compared to the prior maximum. The resulting messages, stored in *bundle*, contain the values that maximize the factors' local utility functions. It is possible to avoid the systematic enumeration by adding a local randomization to the issue that the factor is referencing. Additionally, we can exploit the structure of the constraint though its function's monotonicity. That is, by optimizing the constraint locally and providing the optimal sub-contracts as messages. However, optimizing locally does not produce a global optimization due to the interdependence between constraints (example in Section 2.2).

3.3 Propagation Strategies

The propagation, or circulation of the messages in G could be defined according to a particular strategy with respect to the hypergraph topology. For example, lines 3 in Algorithm 1 refers to a systematic, deterministic or synchronous way of choosing the nodes. Line 7 corresponds to a randomized, non-deterministic, or asynchronous way of selecting the sources and the destinations. In the rest of the paper, we will adopt the asynchronous mode of messages transmission.

4. Exploration Complexity

4.1 Evaluation Criteria

Before the evaluation of the hypergraphical representation and the utility propagation algorithm, it is important to identify the criteria that could affect the complexity of the utility space and thus the probability of finding optimal contract(s). To this end, we start by defining the parameters that could have an impact on the complexity of the preference spaces. These parameters are also used for the generation of the scenarios.

- n: number of issues.
- m: number of constraints, hyperedges, or factors.
- π: for a constraint c_j and its factor Φ_j, $\pi(\Phi_j)$ refers to the number of issues involved in Φ_j, making it unary ($\pi(\Phi_j) = 1$), binary ($\pi(\Phi_j) = 2$), ternary ($\pi(\Phi_j) = 3$), or $\pi(\Phi_j)$-ary in the general case. π is defined as in Eq. (10),

$$\begin{aligned} \pi\ &: \Phi \to [1, n] \\ &\Phi_j \mapsto \underbrace{\mathcal{N}(\Phi_j)}_{\pi_j},\ j \in [1, m] \end{aligned} \tag{10}$$

with the resulting sequence Eq. (11).

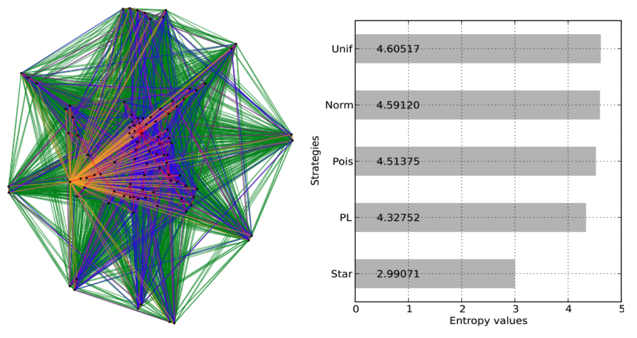
$$\pi = (\pi_1, \ldots, \pi_j, \ldots, \pi_m) \tag{11}$$

- Domain sizes of the issues $\prod_{k=1}^{n} |\mathbb{I}_k|$ as well as the domains' types (discrete and/or continue).

The most important criteria is the distribution π. In fact, it correlates with the computation time necessary for an algorithm to perform, as we will show in the next subsections. A practical way to evaluate different distributions is to define a utility space profile as a tuple (n, m, π). This parametrization will be used in the study of the complexity. It is important to note that a profile (n, m, π) must meet the consistency condition Eq. (12).

$$\pi_j \leq n \leq m \times \pi_j, \quad \forall j \in [1, m] \tag{12}$$

Such condition prevents cases like attempting to have 12−ary constraints in a 7−dimensional utility space. Next, we show how to quantitatively assess complexity using the entropy measure.



(a) General cognitive graph (b) Distributions' entropies

Fig. 4 Cognitive graph and degree distributions' entropies.

4.2 Complexity Evaluation Using Entropy

In the probabilistic sense, the sequence Eq. (11) could follow any distribution law. Furthermore, each distribution will have a complexity mirrored by the underlying utility hypergraph, which will certainly affect the performance of any search algorithms. In the general case, this is known as the degree distribution of a graph. The complexity of a particular profile (n, m, π) is assessed using the information theoretical notion of entropy Eq. (13).

$$H(\pi) = -\sum_{j=1}^{m} \pi_j \log(\pi_j) \tag{13}$$

Entropy could in fact be used to measure the complexity of cognitive graphical models [5], [6], including any representation that uses the idea of degree distribution Eq. (11). Herein, entropy is meant to reflect complexity from a temporal standpoint. As an example, suppose that π is taking different forms $\pi_{i\in[1,5]}$, shown in Eq. (14).

$$\pi \begin{cases} \pi_1 = Uniform(m-1) \\ \pi_2 = Normal(\mu \in [2,4], \sigma \in [0,2]) \\ \pi_3 = Poisson(\lambda \in [2,6]) \\ \pi_4 = Power-law(\alpha = 2.3) \\ \pi_5 = Star \end{cases} \tag{14}$$

The idea behind the distributions is that it is possible to traverse a graph according to different distributions $\pi_{i\in[1,5]}$. Exploring the graph corresponds to a specific way of moving from one node to another as to perform any type of optimization. For example let us take a graph with $m = 100$ nodes, represented in **Fig. 4** (a), with different distributions (π_1: green, π_2: purple, π_3: blue, π_4: red and π_5: orange). For instance, traversing the graph based on a star distribution (π_5, in orange) corresponds to moving from the central node to a peripheral node, then back to the central node, etc. The exploration based on a uniform distribution gives a complete graph (π_1, in green) where we explore based on all the nodes, uniformly. The same logic applies for other distributions.

These distributions differ in their topologies, but most importantly in terms of complexity, or as we refer to it here, entropy. As shown in Fig. 4 (b), uniform strategy has the highest entropy and generates a complete graph (K_{m-1}). On the contrary, the star (S_{m-1}) strategy has the lowest entropy, with nodes connected to one single node acting like a hub. If we increase the number of hubs to few hubs, we get the power law strategy which is ranked right above the star strategy. Normal and poisson distributions

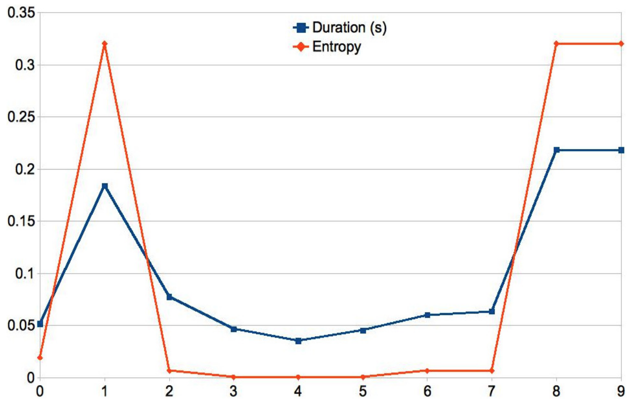


Fig. 5 $H(\pi_j)$ and $\Delta(\pi_j)$ for $\pi_{j\in[0,9]} \in \{\mathcal{U}, \mathcal{D}, \mathcal{PL}\}$.

fluctuate between π_1 and π_4 in terms of complexity, relatively to the values of μ, σ and λ. The uniform π_1 and the star π_5 act as complexity bounds. We will see in the next subsection how entropy, complexity and performance (time) relate to each other.

4.3 Complexity and Performance

Now, let us take a concrete example of exploration strategies and their underlying distributions, and let us evaluate the interplay between the entropy and the computation time needed by the search algorithm to find the optimal contract(s). We assume that we have 10 strategies $\pi_{k\in[0,9]}$ where each π_k is either a uniform distribution $\mathcal{U}$, a deterministic distribution $\mathcal{D}$ or a power-law distribution $\mathcal{PL}$. If the search algorithm is taken to be AsynchMP, then the computation time and the complexity of the underlying strategies are illustrated in **Fig. 5**. The first observation is that both entropy and computation time fluctuate similarly, describing the same topology of the underlying strategy. Secondly, $\mathcal{U}$ is the most complex structure, since it possess the highest entropy and computation time as opposed to $\mathcal{D}$ and $\mathcal{PL}$. It is possible to think about the complexity of strategy $\mathcal{U}$ from two standpoints. An analytical or cartesian view of the problem reduces the complexity to high-dimensionally [3]. From a graphical viewpoint, the distribution is perceived as representing a complete graph with one strong component having the highest number of possible connections. Both views reflect the difficulty of the search problem.

4.4 Optimal Strategy

Instead of the asynchronous mode of AsynchMP (randomly picking v_{src} and v_{dest}), we propose to use the distribution π as a prior, allowing us to optimize the message passing algorithm by taking into consideration certain topologies. For example, adopting a strategy $\pi \sim \mathcal{PL}$ allows us to focus on the hubs of the hypergpah, i.e., the factors with large numbers of issues. Let us call the new strategy AsynchMPi, which consists in performing the message passing within a set $\sigma_1 \subset (\Phi \bigcup \mathcal{I})$ of high degree nodes. For a specific profile (n, m, π), and for two strategies AsynchMP and AsynchMPi, the idea is to see which one converges to the optimium faster, while being certain that both will find this optimum. That is, the same profile will be traversed and explored with different distributions throughout time.

In the following, we show how the set σ_1 is constructed.

(1) Start by generating the sequence c_j, defined as in Eq. (15).

The sequence c_j has the property of having the majority of its points clustered in the upper portion of the domain. It will be used to map the set of high degree nodes.

$$c_j = \begin{cases} \frac{1}{2} & \text{if } j = 0 \\ \frac{1}{2} + \frac{1}{2} \times c_{j-1}^2 & \text{if } j \in [1, (n+m) \times 10] \end{cases} \quad (15)$$

(2) Uniformly sample r points from c_j. The result is $\mathcal{U}(c_j)$

(3) Generate the sequence s_π containing $\{\pi(v_i)\}_i$ by decreasing order Eq. (16). s_π approximates a power-law distribution.

$$s_\pi = \{i, \ldots, n' \mid \pi(v_i) \leq \cdots \leq \pi(v_{n'})\} \quad (16)$$

(4) Generate the set s_σ according to Algorithm 2.

```
Input: Sequences $\mathcal{U}(c_j)$ and $s_\pi$
Output: Set $s_\sigma$ of high connectivity nodes
begin
    $s_\sigma \leftarrow \emptyset$
    for $c \in \mathcal{U}(c_j)$ do
        if $c < \lceil c \rceil - \frac{1}{2}$ then
            $s_\sigma \leftarrow s_\sigma \cup \lceil c \rceil - 1$
        else
            $s_\sigma \leftarrow s_\sigma \cup \lceil c \rceil$
    return $s_\sigma$
```

Algorithm 2: Generation of high degree nodes s_σ.

5. Discussion

The generation of the hypergraph is performed using Algorithm 3. Depending on the nature of π, a particular topology will be generated. A uniform π generates a complete hypergraph, a power-law π generates a scale-free hypergraph and so on. In the following we evaluate AsynchMP and AsynchMPi for 6 profiles $(100, 100, \pi_i)$, $i \in \{5, 6, 7, 8, 9, 10\}$. The profiles have decreasing complexity defined as $\pi_i(\Phi_j) \leq i \; \forall j$.

It is important to make sure that both strategies give the same expected optimal utilities, as shown in **Fig. 7** (a). That is, for the different profiles specified by $i \in \{5, 6, 7, 8, 9, 10\}$ on the x axis, both strategies give the same optimal utility values, shown on the y axis. However, Fig. 7 (b) shows that restricting the message passing process to the high degree nodes (hubs) results in a drastic decrease in the computation time (in seconds) for the same profiles specified by $i \in \{5, 6, 7, 8, 9, 10\}$. The way we restrict the search to hubs is shown in **Fig. 6**. We sort the nodes $v_{i \in [0,59]}$ by decreasing degree ($\pi(v_i)$ is the degree of node v_i) and we randomly sample from this sequence according to the process described in Section 4.4. The resulting sequence of high degree nodes is shown as blue triangles in Fig. 6. We additionally observe that for small connectivity values ($\pi_1 = \pi_2$), the search process takes approximately the same amount of time despite the large number of issues and constraints $n = m = 100$. In fact, assessing the complexity of a utility space (respectively utility hypergrpah) must take into consideration the connectivity function π. In this sense, neither the dimension n nor the number of constraints m could objectively reflect this complexity. For instance, a profile $(100, 100, 1)$ is less complex than a profile $(10, 10, 2)$. In fact, $(100, 100, 1)$ is not a nonlinear utility space because $\pi(\Phi_j) = 1 \; \forall j$. That is, each constraint contains one unique issue, *i.e.*, the whole utility is reduced to a sum of the partial utilities of the individual issues with, $n = m$. Thus, the whole problem becomes linear with the utility function Eq. (17),

$$u(i_1, \ldots, i_j, \ldots, i_n) = \sum_{j=1}^{n} \phi_j(i_j) \quad (17)$$

```
Algorithm: ParamRandHGen
Input: $n, m, \pi$
Output: $G(\mathbb{I}, \Phi)$
begin
    $[\beta_{min}, \beta_{max}] \leftarrow [1, 100]$    // constants
    $[\alpha_{min}, \alpha_{max}] \leftarrow [0, 1]$    // slopes
    $[b_{min}, b_{max}] \leftarrow [0, 9]$    // bounds
    $\Phi \leftarrow [\emptyset] \times m$    // init constraints set
    for $k = 1 \rightarrow m$ do
        $\Phi[k].\theta \leftarrow rand(\{cube, plane, bell\})$
        if $\Phi[k].\theta = plane$ then
            $\alpha \leftarrow [0] \times n$
            for $j = 1 \rightarrow n$ do
                $\alpha[j] \leftarrow rand([\alpha_{min}, \alpha_{max}])$
            $\Phi[k].\alpha \leftarrow \alpha$
        if $\Phi[k].\theta \in \{bell, cube\}$ then
            // similar calculations; refer to (4) or (5)
            // ...
        $\Phi[k].\beta \leftarrow rand([\beta_{min}, \beta_{max}])$
        $\mu \leftarrow rand([1, n])$
        $\mathbb{I} \leftarrow \emptyset$
        while $|\mathbb{I}| \neq \mu$ do
            $\iota \leftarrow \pi(k)$
            if $\iota \notin \mathbb{I}$ then
                $\mathbb{I} \leftarrow \mathbb{I} \cup \iota$
        for $j = 1 \rightarrow \mu$ do
            $\mathbb{I}[j].a \leftarrow rand([b_{min}, b_{max}])$
            $\mathbb{I}[j].b \leftarrow rand([\mathbb{I}[j].a + \epsilon, b_{max}])$
        $\Phi[k].\mathbb{I} \leftarrow \mathbb{I}$
    return $\Phi$
```

Algorithm 3: Utility Hypergraph Generation.

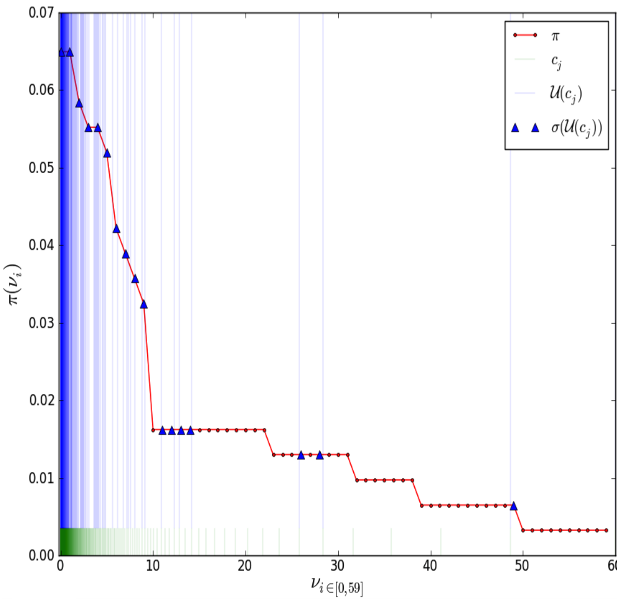
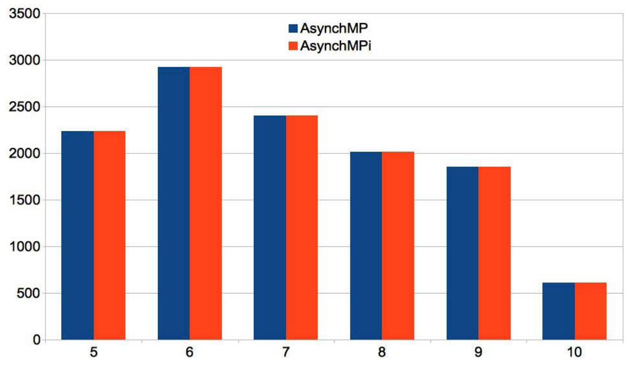
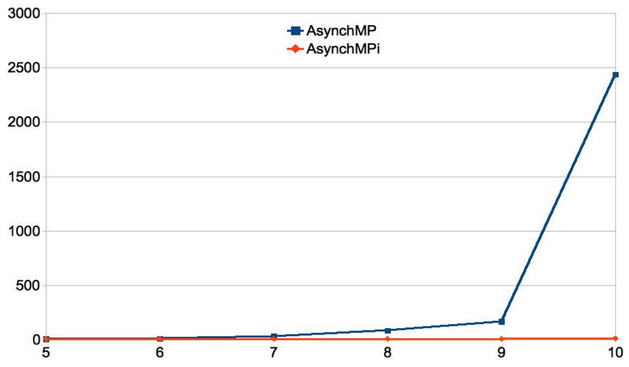


Fig. 6 Generation of s_σ for $n, m, p, r = (10, 50, 5, 17)$.



(a) Utility



(b) Computation time

Fig. 7 AsynchMP vs. AsynchMPi.

with ϕ_j being the utility of constraint c_j. We note that the previous case is a degenerate case, and that generally, we ought to generate constraints with cardinalities greater or equal to 2.

6. Conclusion

We introduced a new modular representation of utility spaces based on hypergraphs. The exploration and search for optimal contracts is performed based on a message passing mechanism outperforming the sampling-based optimizers. Additionally, the model was evaluated in terms of complexity assessment showing that power-law topologies have lower complexity. Consequently, we provided an exploration strategy that searches the hypergrpah based on a power-law topology. Results show that such strategy outperforms drastically the synchronous message passing strategy. As a future work, we intend to exploit the structure of the hypergraphs by proposing an hierarchical exploration scheme and evaluate it in a hierarchical negotiation scenario. Additionally, we intend to study the interdependence between the issues as to assess their importance and influence on the contracts optimality. Being able to assess the issues importance could in fact be used to simplify complex negotiation scenarios by focusing on the most important sub-contracts.

References

[1] Bacchus, F. and Grove, A.: Graphical Models for Preference and Utility, *Proc. 11th Conference on Uncertainty in Artificial Intelligence, UAI'95*, San Francisco, CA, USA, pp.3–10, Morgan Kaufmann Publishers Inc. (1995).

[2] Chajewska, U. and Koller, D.: Utilities as Random Variables: Density Estimation and Structure Discovery, *Proc. 16th Annual Conference on Uncertainty in Artificial Intelligence* (*UAI-00*), pp.63–71 (2000).

[3] Donoho, D.L.: High-dimensional data analysis: The curses and blessings of dimensionality, *American Mathematical Society Conf. Math Challenges of the 21st Century* (2000).

[4] Fujita, K., Ito, T. and Klein, M.: An Approach to Scalable Multi-issue Negotiation: Decomposing the Contract Space Based on Issue Interdependencies, *Proc. 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 02, WI-IAT '10*, Washington, DC, USA, pp.399–406, IEEE Computer Society (2010).

[5] Hadfi, R. and Ito, T.: Cognition as a Game of Complexity, *Proc. 12th International Conference on Cognitive Modeling* (*ICCM*) (2013).

[6] Hadfi, R. and Ito, T.: Uncertainty of Cognitive Processes with High-information Load, *Procedia - Social and Behavioral Sciences*, Vol.97, No.0, pp.612–619 (2013).

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

[8] Ito, T., Klein, M. and Hattori, H.: A multi-issue negotiation protocol among agents with nonlinear utility functions, *Multiagent and Grid Systems*, Vol.4, No.1, pp.67–83 (2008).

[9] Jennings, N.R.: An Agent-based Approach for Building Complex Software Systems, *Commun. ACM*, Vol.44, No.4, pp.35–41 (online), DOI: 10.1145/367211.367250 (2001).

[10] Ke, W., Peng, Z., Yuan, Q., Hong, B., Chen, K. and Cai, Z.: A method of task allocation and automated negotiation for multi robots, *Journal of Electronics* (*China*), Vol.29, No.6, pp.541–549 (online), available from ⟨http://dx.doi.org/10.1007/s11767-012-0868-x⟩ (2012).

[11] Krainin, M., An, B. and Lesser, V.R.: An Application of Automated Negotiation to Distributed Task Allocation, *IAT*, pp.138–145, IEEE Computer Society (2007).

[12] Kraus, S.: *Strategic Negotiation in Multiagent Environments*, MIT Press, Cambridge, MA, USA (2001).

[13] Kraus, S., Sycara, K. and Evenchik, A.: Reaching agreements through argumentation: A logical model and implementation, *Artificial Intelligence*, Vol.104, No.12, pp.1–69 (online), DOI: http://dx.doi.org/10.1016/S0004-3702(98)00078-2 (1998).

[14] Kwisthout, J. and van Rooij, I.: Bridging the gap between theory and practice of approximate Bayesian inference, *Cognitive Systems Research*, Vol.24, pp.2–8 (2013).

[15] Lin, R., Kraus, S., Wilkenfeld, J. and Barry, J.: Negotiating with bounded rational agents in environments with incomplete information using an automated agent, *Artif. Intell.*, Vol.172, No.6-7, pp.823–851 (2008).

[16] Lopez-Carmona, M.A., Marsa-Maestre, I., De La Hoz, E. and Velasco, J.R.: A Region-based Multi-issue Negotiation Protocol for Non-monotonic Utility Spaces, *Computational Intelligence*, Vol.27, No.2, pp.166–217 (2011).

[17] Marsa-Maestre, I., Lopez-Carmona, M.A., Velasco, J.R. and de la Hoz, E.: Effective bidding and deal identification for negotiations in highly nonlinear scenarios, *Proc. 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS '09*, Richland, SC, pp.1057–1064, International Foundation for Autonomous Agents and Multiagent Systems (2009).

[18] Marsa-Maestre, I., Lopez-Carmona, M., Carral, J. and Ibanez, G.: A Recursive Protocol for Negotiating Contracts Under Non-monotonic Preference Structures, *Group Decision and Negotiation*, Vol.22, No.1, pp.1–43 (2013).

[19] Pearl, J.: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988).

[20] Robu, V., Somefun, D.J.A. and Poutre, J.L.: Modeling complex multi-issue negotiations using utility graphs, *Proc. 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems* (*AAMAS 2005*), pp.280–287 (2005).

Rafik Hadfi was born in 1984. He received the M.Eng. from the Nagoya Institute of Technology (Japan). He is currently a Ph.D. student at the same institution. His research interests include Multiagent Systems, Decision Making, Preferences Elicitation and Cognitive modeling. He is a member of the Association for the Advancement of Artificial Intelligence (AAAI), the Information Processing Society of Japan (IPSJ), and the Institute of Electrical and Electronics Engineers (IEEE).

Takayuki Ito was born in 1972. He is currently Professor of the Nagoya Institute of Technology (Japan). He received the B.E., M.E., and Dr. Eng. from the Nagoya Institute of Technology in 1995, 1997, and 2000, respectively. From 1999 to 2001, he was a research fellow of the Japan Society for the Promotion of Science (JSPS). From 2000 to 2001, he was a visiting researcher at USC/ISI (University of Southern California/Information Sciences Institute). From Apr. 2001 to Mar. 2003, he was an associate professor of Japan Advanced Institute of Science and Technology (JAIST). From 2005 to 2006, he is a visiting researcher at Division of Engineering and Applied Science, Harvard University and a visiting researcher at the Center for Coordination Science, MIT Sloan School of Management. From 2008 to 2010, he was a visiting researcher at the Center for Collective Intelligence, MIT Sloan School of Management. He is a senior member of the Association for Computational Machinery (ACM), a member of the American Association for Artificial Intelligence (AAAI), the Japanese Society for Artificial Intelligence (JSAI), the Information Processing Society of Japan (IPSJ), the Institute of Electronics, Information and Communication Engineers (IEICE), the Japan Society for Software Science and Technology (JSSST), the Society of Instrument and Control Engineers (SICE), and the Japanese Economic Association (JEA).