https://doi.org/10.20965/jaciii.2015.p0514

Paper:

Complex Multi-Issue Negotiation Using Utility Hyper-Graphs

Rafik Hadfi and Takayuki Ito

Department of Computer Science and Engineering, Nagoya Institute of Technology
Gokiso, Showa-ku, Nagoya 466-8555, Japan
E-mail: {rafik@itolab., ito.takayuki@}nitech.ac.jp

[Received September 27, 2014; accepted April 10, 2015]

We propose to handle the complexity of utility spaces used in multi-issue negotiation by adopting a new representation that allows a modular decomposition of the issues and the constraints. This is based on the idea that a constraint-based utility space is nonlinear with respect to issues, but linear with respect to the constraints. This allows us to rigorously map the utility space into an issue-constraint hyper-graph. Exploring the utility space reduces then to a message passing mechanism along the hyper-edges of the hyper-graph by means of utility propagation. Optimal contracts are found efficiently using a variation of the Max-Sum algorithm. We evaluate the model experimentally using parameterized nonlinear utility spaces, showing that it can handle a large family of complex utility spaces by finding optimal contracts, outperforming previous sampling-based approaches. We also evaluate the model in a negotiation setting. We show that under high complexity, social welfare could be greater than the sum of the individual agents' best utilities.

Keywords: multi-agent systems, multi-issue negotiation, nonlinear utility spaces, hyper-graph, max-sum

1. Introduction

Realistic negotiation involves multiple interdependent issues, yielding complex and nonlinear utility spaces. Reaching a consensus among agents becomes more difficult as the search space and problem complexity grow. In this paper, we propose to tackle the complexity of utility spaces used in multi-issue negotiation by rethinking the way in which they are represented. We hold that adopting adequate representation provides the scaling problem with a solid ground to tackle. We do so by using a representation that enables a modular decomposition of the issues and constraints. This is based on the idea that a constraint-based utility space is nonlinear with respect to issues, but linear with respect to the constraints. This enables us to map the utility space rigorously into an issue-constraint hyper-graph with the underlying interdependencies. Exploring the utility space reduces to a message passing mechanism along the hyper-edges using utility propagation.

Adopting a graphical representation while reasoning about preferences is not new in the multi-issue negotiation literature. In fact, the idea of utility graphs could potentially help breaking down highly nonlinear utility functions into sub-utilities of clusters of inter-related items, as in [1, 2]. Utility graphs have been used for preferences elicitation and negotiation over binary-valued issues [3]. A weighted undirected graph has been used to represent constraint-based utility spaces [4]. Specifically, a message passing algorithm has been used to find the highest utility bids by finding the set of unconnected nodes that maximize the sum of nodes' weight. Restricting graph and message passing to nodes of constraints does not however enable 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. Furthermore, the interdependency of issues is captured using similar undirected weighted graphs for when a node represents an issue [5]. This representation is restricted to binary interdependencies whereas real negotiation scenarios involve "bundles" of interdependent issues defined under one or more constraints. In our approach, we do not restrict interdependency to lower-order constraints. Instead, we allow $p-$ary interdependencies to be defined as hyper-edges connecting p issues. Using such graphical representation with its underlying utility propagation mechanism is based on the idea that negotiation is, after all, a cognitive process that involves concepts and associations performed by bounded rational agents. Bearing in mind that cognitive processes perform some form of Bayesian inference [6], we used a graphical representation that serves as an adequate framework for any preference-based space.

Using this representation has the advantage of scalability in the sense that the problem becomes more difficult due to its large number of issues and constraints. Decomposing the utility space enables us to exploit it more efficiently. Another way of looking at this "connectionist" representation is that it can be clustered in ways that isolate interdependent components, thus, enabling them to be treated separately and even negotiated independently form the rest of the problem. Another motivation behind hyper-graph representation is that it enables a layered, hierarchical view of any given negotiation scenario [7, 8]. Such an architecture, lets us recursively negotiate over different layers of the problem according to a top-down approach. Even the idea of an issue can be abstracted to include an encapsulation of sub-issues, located in sub-utility spaces and represented by cliques in the hyper-graph. Search processes can then be used to help identify optimal contracts for improvement at individual leves. This combination of separating the system into layers, and then using utility propagation to focus attention on and search within a constrained region can be very powerful in the bidding process. A similar idea of recursion the exploring of utility space was introduced by [9], although it is region-oriented and does not graphically represent the utility space. We evaluated our model in experiments using parametrized and random nonlinear utility spaces. We showed that the model can handle large complex spaces by finding optimal bids while outperforming previous sampling approaches. We also evaluated our model in a mediated multi-lateral negotiation scenario to assess social welfare.

The paper is organized as follows. Section 2 proposes the basics of our new nonlinear utility space representation. Section 3 describes the search mechanism. Section 4 details the experimental results. Section 5 lists the conclusions and outlines the future work.

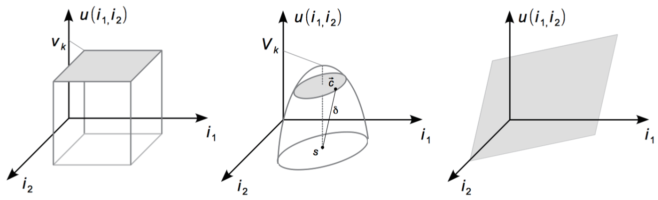


Fig. 1. Cubic, bell and plane constraints.

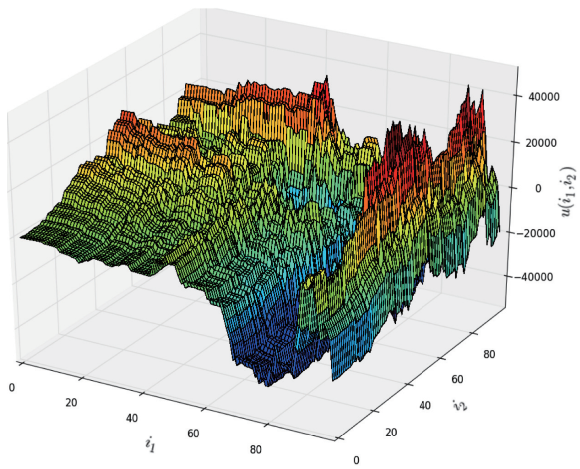


Fig. 2. 2−D nonlinear utility space.

2. Nonlinear Utility Space Representation

2.1. Formulation

We start by formulating the nonlinear multi-issue utility space used in [10], meaning that an n-dimensional utility space is defined over a finite set of issues $\mathbb{I} = \{i_1,\dots,i_k,\dots,i_n\}$. Issue k, namely i_k, takes its values from finite set $\mathbb{I}_k$ with $\mathbb{I}_k \subset \mathbb{Z}$. Contract $\vec{c}$ is a vector of issue values with $\vec{c} \in \mathscr{I}$ and $\mathscr{I} = \times_{k=1}^{n} \mathbb{I}_k$.

An agent's utility function is defined in terms of m constraints, making the utility space constraint-based. That is, constraint c_j is a region of the total n-dimensional utility space. We say that constraint c_j has value $w(c_j,\vec{c})$ for contract $\vec{c}$ if c_j is satisfied by $\vec{c}$, i.e., when contract point $\vec{c}$ falls within the hyper-volume defined by constraint c_j, namely $hyp(c_j)$. The utility of an agent for contract $\vec{c}$ is thus defined as in Eq. (1).

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

Below, we distinguish among three types of constraints – cubic, bell and plane – as shown in **Fig. 1**. Constraint-based utility formalism is a practical way of reasoning about preferences that have restrictions, as detailed elsewhere [4, 9, 11].

Such representation (Eq. (1)) produces "bumpy" nonlinear utility spaces with high points where many constraints are satisfied and low points where few or no constraints are satisfied. **Fig. 2** 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 $m = 500$ constraints.

2.2. New Representation

The utility function (1) is nonlinear in that the utility cannot be expressed as a linear function of the contracts [10]. This is true to the extent that linearity is evaluated with regard to contract $\vec{c}$. From the same Eq. (1), however, we can say that the utility is in fact linear, but in terms of constraints $\{c_1,\dots,c_j,\dots,c_m\}$. The utility space is therefore decomposable based on these constraints. This yields a modular representation of the interactions between the issues and how they relate locally 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 contracts are projected according to their issues' components. In this case, interdependence is between constraints, not between issues. Two constraints c_1 and c_2 can have one issue i_k in common taking values from 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 difficult because changing the value of i_k in c_1 changes the instance of i_k in c_2 cyclically. This nonlinearity gets worse as the number of issues, domain sizes, and the non-monotonicity of constraints increases.

We propose to transform Eq. (1) into a modular graphical representation. Since one constraint may involve one or more issues, we use hyper-graph representation.

2.3. From Utility Space to Utility Hyper-graph

To each constraint c_j, we assign factor Φ_j, yielding factor set $\Phi = \{\Phi_1,\dots,\Phi_j,\dots,\Phi_m\}$. Utility hyper-graph G

is defined as $G = (\mathbb{I}, \Phi)$. Nodes in $\mathbb{I}$ define issues and hyper-edges in Φ are factors (constraints). Neighbors set $\mathscr{N}(\Phi_j) \subset \mathbb{I}$ of factor Φ_j represents the issues connected to Φ_j (involved in constraint c_j), with $|\mathscr{N}(\Phi_j)| = \varphi_j$. In case $\varphi_j = 2\ \forall j$ the problem collapses into a constraints satisfaction problem in a standard graph.

φ_j-dimensional matrix $\mathscr{M}_{\Phi_j}$ corresponds to each factor Φ_j, where the kth dimension is $\mathbb{I}_k = [a_k, b_k]$, known as the domain of issue i_k. This matrix contains all values that could be taken by the issues in $\mathscr{N}(\Phi_j)$. Each factor Φ_j has function u_j defined as a sub-utility function of the issues in $\mathscr{N}(\Phi_j)$, defined as in Eq. (2)

$$\begin{aligned} u_j :\ & \mathscr{N}(\Phi_j)^{\varphi_j} \to \mathbb{R} \\ & u_j(\vec{x}) \mapsto w(c_j, \vec{x}),\ \vec{x} = (i_1, \ldots, i_j, \ldots, i_{\varphi_j}) \end{aligned} \tag{2}$$

We are dealing with discrete issues, so u_j is the mapping defined by matrix $\mathscr{M}_{\Phi_j}$. That is, $u_j(i_1, \ldots, i_k, \ldots, i_{\varphi_j})$ is simply the $(1, \ldots, k, \ldots, \varphi_j)^{th}$ entry in $\mathscr{M}_{\Phi_j}$ corresponding to value $w(c_j, \vec{x})$ described in Eq. (2). The discrete case can be extended to the continuous case by enabling continuous issue values and defining u_k as a continuous function. We provide examples of representation in the sections that follow.

2.3.1. Example 1

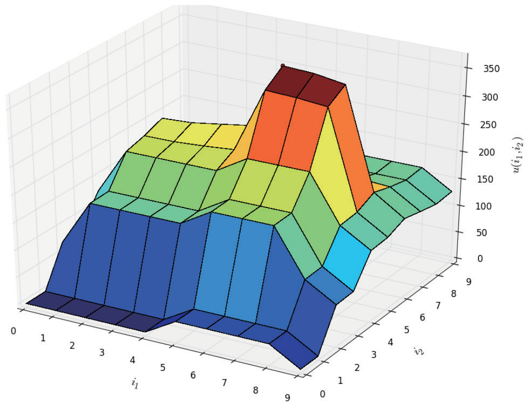
To show the mapping between 2−D utility space and its corresponding hyper-graph G_2 (**Fig. 3(b)**), the issue domains are $\mathbb{I}_1 = \mathbb{I}_2 = [0, 9]$. G_2 consists of $m = 10$ constraints (red squares) where each constraint involves at most 2 issues (white circles). Note the 6 cubic constraints, 3 plane constraints and 1 bell constraint.

2.3.2. Example 2

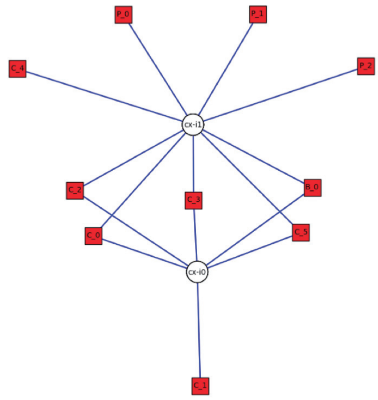
We consider 10−D utility space mapped onto hyper-graph $G_{10} = (\mathbb{I}, \Phi)$ with $\mathbb{I} = \{i_0, \ldots, i_9\}$ and $\Phi = \{\Phi_1, \ldots, \Phi_7\}$ on **Fig. 4**. Each issue i_k has set $\mathbb{I}_k = \bigcup_{v \in \mathscr{N}(i_k)} \mathbb{I}_{k,v}$ where $\mathbb{I}_{k,v}$ is an edge connecting i_k to its neighbor $v \subset \mathscr{N}(i_k) \in \Phi$, 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 cubic $(\Phi_1, \Phi_2, \Phi_3, \Phi_4)$, plane (Φ_5, Φ_6) and bell (Φ_7). Each constraint c_j (respectively factor Φ_j) is assigned a sub-utility function u_j to find the utility of a contract if it satisfies c_j by being located in $hyp(c_j)$. Depending on its type, each constraint's sub-utility is defined as in Eq. (3).

$$u_j \begin{cases} \beta_j + \sum_{k=1}^{\varphi_j} \alpha_{j,k} i_k,\ \beta_j, \alpha_{j,k} \in \mathbb{Z} & \text{if Plane} \\ v_j & \text{if Cube} \\ V_j & \text{if Bell} \end{cases} \tag{3}$$

That is, plane constraint Φ_j is defined using a φ_j−dimensional equation, while a cubic constraint is assigned value v_j. Sub-utility V_j of bell-shaped constraints is defined in Eq. (4). Here, δ is the Euclidean distance from the center of the bell constraint to the contract point.



(a) Utility space



(b) Utility hyper-graph

Fig. 3. 2−D utility space and its hyper-graph.

Distances are normalized in $[-1, 1]$.

$$V_j \begin{cases} \beta_j\,(1 - 2\delta^2) & \text{if } \delta < \frac{1}{2} \quad \beta_j \in \mathbb{Z} \\ 2\,\beta_j\,(1 - \delta)^2 & \text{if } \delta < 1 \quad \beta_j \in \mathbb{Z} \\ 0 & \text{else} \end{cases} . \tag{4}$$

3. Optimal Contracts

The exploration of the utility hyper-graph was inspired by the sum-product message-passing algorithm for loopy belief propagation [12]. The multiplicative algebra is changed to an additive algebra, however, to support utility accumulation necessary for assessing contracts. Messages circulating in the hyper-graph are simply the contracts we are attempting to optimize through utility maximization.

3.1. Message Passing (MP) Mechanism

We consider issue set $\mathbb{I}$ and contract point $\vec{c} = (i_1, \ldots, i_k, \ldots, i_n) \in \mathscr{I}$. We want to find contract $\vec{c}^*$ that maximizes the utility function (1). Assuming that u_j is a local sub-utility of constraint Φ_j, we distinguish between two types of messages, those sent from issues to constraints and those sent from constraints to issues.

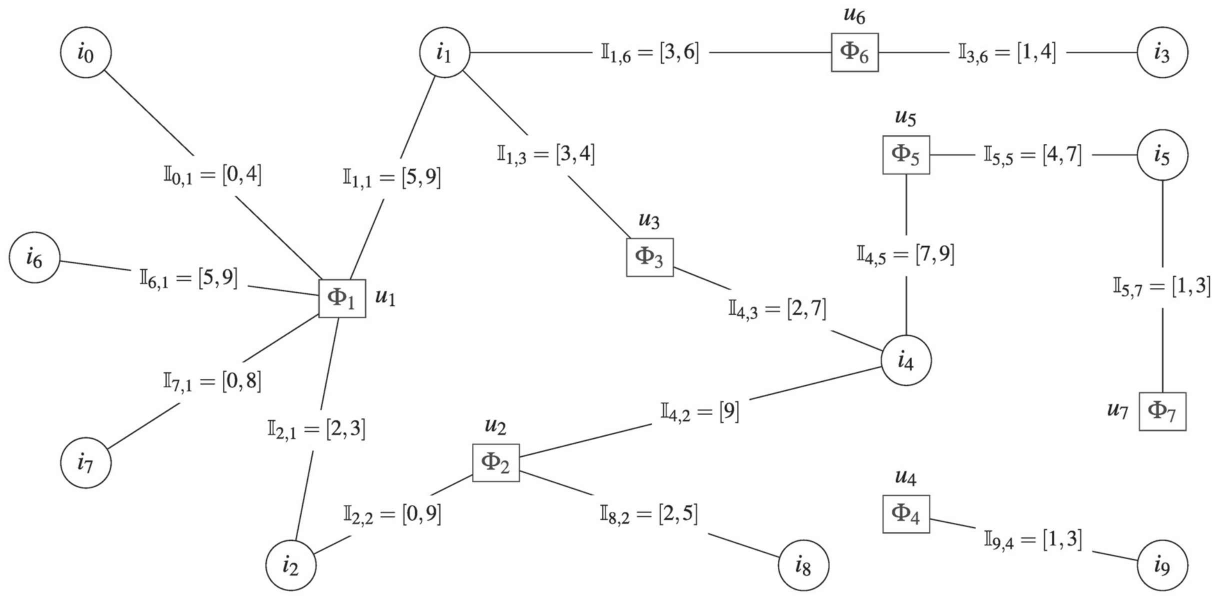


Fig. 4. Issues-constraints hyper-graph.

3.1.1. From Issue i_k to Constraint Φ_j

In Eq. (5), each message $\mu_{i_k \to \Phi_j}$ from i_k to Φ_j is the sum of constraint messages to i_k from constraints other than Φ_j.

$$\mu_{i_k \to \Phi_j}(i_k) = \sum_{\Phi_{j'} \in \mathscr{N}(i_k) \backslash \Phi_j} \mu_{\Phi_{j'} \to i_k}(i_k) \quad . \quad . \quad . \quad . \quad (5)$$

3.1.2. From Constraint Φ_j to Issue i_k

Each constraint message is the sum of messages from issues other than i_k, plus sub-utility $u_j(i_1, \ldots, i_k, \ldots, i_n)$, summed over all possible values of issues connected to constraint Φ_j other than issue i_k.

$$\mu_{\Phi_j \to i_k}(i_k) = \max_{i_1} \ldots \max_{i_{k'} \neq k} \ldots \max_{i_n} \left[u_j(i_1, \ldots, i_k, \ldots, i_n) + \sum_{i_{k'} \in \mathscr{N}(\Phi_j) \backslash i_k} \mu_{i_{k'} \to \Phi_j}(i_k) \right] \quad . \quad . \quad . \quad . \quad (6)$$

The MP mechanism starts from the leaves of the hyper-graph, *i.e.*, the issues. At $t = 0$, an initial message content is defined based on Eq. (7), with $u'_j(i_k)$ being the partial evaluation of i_k in factor Φ_j.

$$\mu_{i_k \to \Phi_j}(i_k) = 0 \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (7)$$
$$\mu_{\Phi_j \to i_k}(i_k) = u'_j(i_k)$$

An optimal contract $\vec{c}^*$ is found at any time by collecting optimal issue-values as in Eq. (8a).

$$\vec{c}^* = (i_1^*, \ldots, i_k^*, \ldots, i_n^*) \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (8a)$$
$$i_k^* = \arg\max_{i_k} \sum_{\Phi_j \in \mathscr{N}(i_k)} \mu_{\Phi_j \to i_k}(i_k) \quad . \quad . \quad . \quad . \quad . \quad (8b)$$

In a strategic encounter between agents, an agent more commonly requires a collection, or *bundle*, of optimal contracts rather than a single optimal contract. To find such a collection, we must endow Eq. (8b) with a caching mechanism enabling individual nodes in the hyper-graph to store messages sent to it from other nodes. That is, cached messages will contain summed-up utility values of an underlying node's instance. This is done each time *max* is called in Eq. (6), so that we can store the settings of the adjacent utility and contract that led to a maximum. Once ordered, such a data structure could be used, for instance in a bidding process.

3.2. Utility Propagation Algorithm

3.2.1. Main Algorithm

Algorithm 1 operates on hyper-graph nodes by triggering the MP process. Despite the fact that we have two types of nodes – issues and constraints –, it is possible to treat them abstractly using *MsgPass* (**Algorithms 2** and **3**). The resulting bundle is a collection of optimal contracts with utility equal to or greater than the agent's reservation value *rv*.

3.2.2. Issue's Messages to Constraints

The issue's message to a factor, or constraint, is the element-wise sum of all incoming messages from other factors, as shown in **Algorithm 2**.

3.2.3. Constraint's Messages to Issues

In **Algorithm 3**, the factor's message to a targeted issue is sent by recursively enumerating over all variables that the factor references, except the targeted issue Eq. (6). This must be performed for each value of the target variable in order to compute the message. If all issues are assigned ($\nexists i : \alpha[i] = -1$), the values of the factor and of all other incoming messages are determined so that their

Algorithm 1: Main Algorithm

```
Algorithm: Utility Propagation
Input: G = (𝕀, Φ), rv, mode, ρ
Output: Optimal contracts (bundle)
begin
    for i = 1 → (ρ × |𝕀 ∪ Φ|) do
        if mode is Synchronous then
            foreach v_src ∈ 𝕀 ∪ Φ do
                foreach v_dest ∈ v_src.Neighbors() do
                    v_src.MsgPass(v_dest)
                end
            end
        end
    end
    bundle ← ∅
    foreach i ∈ 𝕀 do
        bundle[i] ← ∅
        ι ← ∪_{j∈i.instances()} [j.min, j.max]
        μ* ← k* ← −∞
        μ ← i.getmax()
        foreach k = 1 → |μ| do
            if μ* < μ[k] then
                μ* ← μ[k]
                k* ← k
                if μ* ≥ rv then
                    bundle[i] ← bundle[i] ∪ ι[k*]
                end
            end
        end
    end
    return bundle
end
```

Algorithm 2: MsgPass: Issue's Messages to Constraints

```
Algorithm: MsgPass
Input: G(𝕀, Φ), Φ_j
Output: Updated message μ
begin
    μ ← [0] × |𝕀_k|
    for ν ∈ 𝒩(i_k)\Φ_j do
        μ ← μ + ν.GetMsg()
    end
    return μ
end
```

Algorithm 3: MsgPass: Constraint's Messages to Issues

```
Algorithm: MsgPass
Input: G(𝕀, Φ), i_k
Output: Updated message μ
begin
    α ← [−1] × φ_j
    ι ← π_{i_k}(Φ_j)
    if ι = ∅ then
        μ ← i_k.GetMsg()
        for i = 1 → len(μ) do
            α[ι] ← i
            μ[i] ← Sum(α, i_k)
        end
        return μ
    end
end
```

sum term is compared to the prior maximum, as in **Algorithm 4**. Resulting messages, stored in variable *bundle*, contain the values that maximize the factors' local utility functions.

3.2.4. Optimal Issue Values

It is possible at any time in the utility propagation process to collect the current optimal contract(s) by individually concatenating all of the optimal issue-values i_k^*, defined in Eq. (8b). Specifically, the summation in Eq. (8b) is performed so as to include only overlapping evaluations, depending on how issue domains are defined for different factors. **Fig. 5**, for instance, shows how issue i_k has three possible evaluations depending on intervals $\mathbb{I}_{k,1}$, $\mathbb{I}_{k,j}$ and $\mathbb{I}_{k,m}$ with $\mathbb{I}_{k,1}$, $\mathbb{I}_{k,j}$, $\mathbb{I}_{k,m} \subset [0,6]$.

The objective function Eq. (8b) will attempt to find the combination(s) of $v_{1,i}$, $v_{k,i}$ and $v_{m,i}$ that maximize the sum. An optimal combination is an optimal issue-value i_k^*.

4. Experiments

4.1. Setup

Before evaluating the utility propagation algorithm, we identify the criteria that could affect the complexity of a utility space and thus the probability of finding optimal contract(s). In addition to n and m, we distinguish p, defined as the maximal number of issues involved in a constraint. p may be unary ($p = 1$), binary ($p = 2$), ternary ($p = 3$), or p-ary in the general case. The parametrized generation of utility spaces or utility hyper-graphs must meet consistency condition $p \leq n \leq m \times p$, with $n, m, p \in \mathbb{N}^+$, to avoid problems such as attempting to have $8-$ary constraints in a 5-dimensional utility space.

4.2. Utility Propagation

We start by generating the hyper-graph using **Algorithm 5**.

We then compare the MP mechanism to the simulated annealing (SA) approach in [10] in terms of utility and duration for an optimal contract(s) search. The SA optimizer will be randomly sampling from regions corresponding to an overlap of constraints. Generating a random contract satisfying c_j, for instance, is performed backward through the random generation of values from $\mathbb{I}_{k,j}$ $\forall i_k \in \mathcal{N}(\Phi_j)$. Our comparison criteria are based on the utility/duration performed on a set of profiles of the form (n, m, p), with 100 trials for each profile. The first version of the message passing process, SynchMP, is synchronous. That is, all combinations of issue evaluations are generated de-

Algorithm 4: Sum: recursive summing

Algorithm: Sum
Input: $G(\mathbb{I},\Phi),\ \alpha,\ i_k$
Output: max
begin
  $\iota \leftarrow \{i \mid i \in [1,\varphi_j] \wedge \alpha[i] = -1\}$
  if $\iota = \emptyset$ **then**
    $\rho \leftarrow \mathscr{M}_{\Phi_j}[\alpha]$
    for $v \in \mathscr{N}(\Phi_j)\backslash i_k$ **do**
      $\mu \leftarrow v.GetMsg()$
      $\rho \leftarrow \rho + \mu[\alpha[i]]$
    end
    return ρ
  end
  else
    $max \leftarrow -\infty$
    for $i = 1 \rightarrow dim(\mathscr{M}_{\Phi_j}(\iota))$ **do**
      $\alpha[\iota] \leftarrow i$
      $\sigma \leftarrow Sum(\alpha, i_k)$
      if $\sigma > max$ **then**
        $max \leftarrow \sigma$
      end
    end
    return max
  end
end

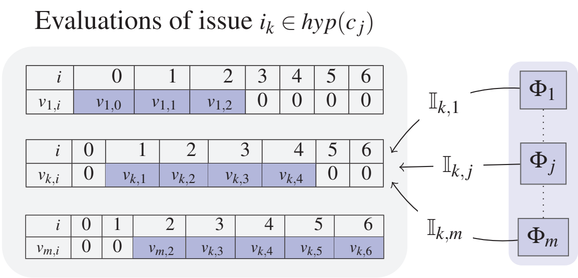


Fig. 5. Finding optimal issue values.

terministically. **Fig. 6** shows SynchMP performance for $(10,[10,20,30],5)$.

The deterministic aspect of SynchMP makes it very slow ($\Delta_{SA} \ll \Delta_{SynchMP}$) indeed compared to its SA counterpart, which exploits randomization, enabling it to perform "jumps" in the search space. To avoid enumeration over local nodes of G, it is possible to add randomization to the way nodes are selected. To introduce the asynchronous mode AsynchMP, we add a new mode after the synchronous mode condition in **Algorithm 1**:

if *mode* is *Asynchronous* **then**
  $v_{src},\ v_{dest} \leftarrow rand_2([1,|V|]),\ v_{dest} \neq v_{src}$
  $v_{src}.MsgPass(v_{dest})$

For $(40,[20,\ldots 100],5)$, **Fig. 7** shows the difference in AsynchMP performance compared to SA.

Algorithm 5: Utility hyper-graph generation

Algorithm: ParamRandHGen
Input: n, m, p
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**
      $\alpha \leftarrow [0] \times n$
      $\alpha[j] \leftarrow rand([\alpha_{min}, \alpha_{max}])\ \forall i \in [1,n]$
      $\Phi[k].\alpha \leftarrow \alpha$
    end
    if $\Phi[k].\theta \in \{bell, cube\}$ **then**
      // refer to (3) or (4)
    end
    $\Phi[k].\beta \leftarrow rand([\beta_{min}, \beta_{max}])$
    $\mu \leftarrow rand([1,n])$, $\mathbb{I} \leftarrow \emptyset$
    while $|\mathbb{I}| \neq \mu$ **do**
      $\iota \leftarrow rand([1,p])$
      if $\iota \notin \mathbb{I}$ **then**
        $\mathbb{I} \leftarrow \mathbb{I} \cup \iota$
      end
    end
    for $j = 1 \rightarrow \mu$ **do**
      $\mathbb{I}[j].a \leftarrow rand([b_{min}, b_{max}])$
      $\mathbb{I}[j].b \leftarrow rand([\mathbb{I}[j].a + \varepsilon, b_{max}])$
    end
    $\Phi[k].\mathbb{I} \leftarrow \mathbb{I}$
  end
  return Φ
end

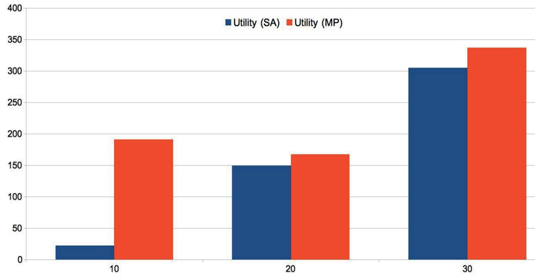
4.3. Multilateral Case and Social Welfare

Next, taking i to be an agent, we assume in a multilateral case that N agents use AsynchMP for bidding over n issues with varying constraints (m) and cardinalities (p). The general mediation protocol involves mediator $\mathscr{M}$ receiving bundle B_t^i from agent $i \in [1,N]$ at time $t \in [0,T]$. In round t, agent i's bundle, described in Eq. (9), contains n_t^i bids.

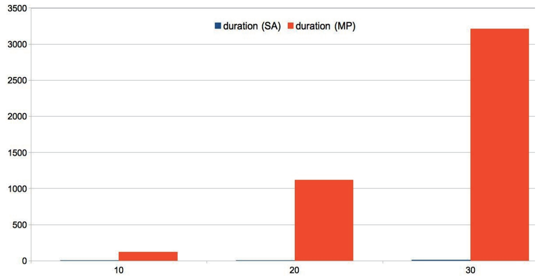
$$B_t^i = \{b_{t,1}^i, \ldots, b_{t,k}^i, \ldots, b_{t,n_t^i}^i\} \quad \ldots\ldots\ldots \quad (9)$$

A bundle must respect preference order $\preceq$ with regard to the agent's subjective utility, *i.e.*, $b_{t,1}^i \preceq \cdots \preceq b_{t,n_t^i}^i$. The feedback of $\mathscr{M}$ at t is contract point $x_{\mathscr{M}}^t$ that agents choose to accept or ignore by providing a new bundle. This process is repeated until deadline T is reached with final contract $x_{\mathscr{M}}^*$.

Due to the randomized nature of utility space, we make mediator $\mathscr{M}$ use a family of sampling algorithms Eq. (10) that attempt to consider the geometrical topology of received bundles to generate candidate contracts. We are interested here in evaluating social welfare [13, 14] yielded

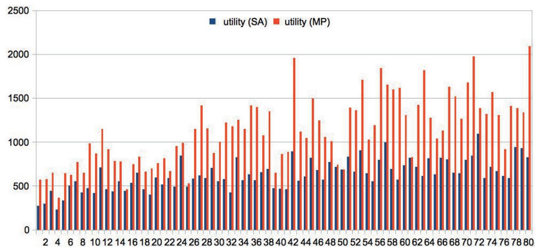


(a) Utility

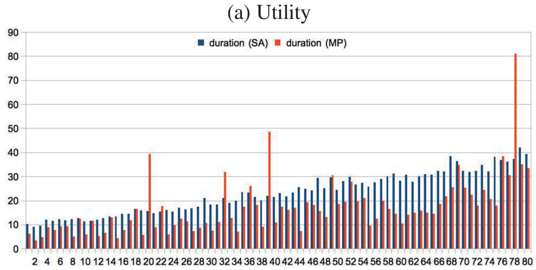


(b) Durations Δ_{SA} and $\Delta_{SynchMP}$ for $m \in \{10, 20, 30\}$

Fig. 6. SynchMP vs. SA for $(10, [10, 20, 30], 5)$.



(a) Utility



(b) Durations Δ_{SA} and $\Delta_{ASynchMP}$ for $m \in [20, 100]$

Fig. 7. AsynchMP vs. SA for $(40, [20, \ldots 100], 5)$.

by individual uses of AsynchMP coupled with sampling algorithms $x_{\mathscr{M}}$. We start by defining the family of algorithms Eq. (10),

$$x_{\mathscr{M}} \left\{ \begin{array}{ll} x^1_{\mathscr{M}} & = c(\{b^i_{t,n^i_t},\ i \in [1,N]\}) \\ x^2_{\mathscr{M}} & = c(\{\mathscr{N}(b^i_{t,n^i_t}),\ i \in [1,N]\}) \\ x^3_{\mathscr{M}} & = c(\{c(C^i_t)\ \forall i \in [1,N]\}) \\ x^4_{\mathscr{M}} & = c(\{f(C^i_t)\ \forall i \in [1,N]\}) \end{array} \right. . \quad (10)$$

with C^i_t being the convex hull of bundle B^i_t and $\mathscr{N}(x)$ the neighbor set of x in $\mathbb{R}^n$. Functions are defined as follows: c returns the centroid of a convex hull of a set of contract points, f, defined in Eq. (11), picks a random contract within a convex hull $x \subset \mathbb{R}^n$, g, defined in Eq. (12), returns a random point from a segment $[x, y] \in \mathbb{R}^{n\times 2}$, h picks a random simplicial facet from a convex hull.

$$f(x) = \left\{ \begin{array}{ll} x & \text{if } x \in \mathbb{R}^n \\ g(c(x),\ (f \circ h)(x)) & \text{else} \end{array} \right. . \quad (11)$$

$$g(x,y) = \alpha x + (1-\alpha)y, \alpha \in [0,1] \quad . . . . . \quad (12)$$

If the final contract is $x^*_{\mathscr{M}}$, we want to evaluate social welfare using a set of social welfare functions (SWF). Specifically, Eqs. (13a) and (13b) assigning high weights to low utilities and low weights to high utilities. Most importantly, we define a differential SWF in Eq. (13c). W_D evaluates the difference between what the mediator proposes to the agents ($x^*_{\mathscr{M}}$) and what the agents' subjective best options (x^*_i, $\forall i \in [1,N]$) are.

$$W_U = \sum_{i=1}^{N} u_i(x^*_{\mathscr{M}}) \quad . . . . \quad (13a)$$

$$W_{\alpha U} = \sum_{i=1}^{N} \alpha_i u_i(x^*_{\mathscr{M}}) \quad . . . . \quad (13b)$$

$$W_D = \sum_{i=1}^{N} u_i(x^*_{\mathscr{M}}) - \sum_{i=1}^{N} u_i(x^*_i) = W_U - W_{parts} \quad (13c)$$

Fig. 8 shows SWFs for $N = 100$ agents, bidding over 10 issues with $(m, p) = (50, 5)\ \forall i \in [1,N]$, where $\mathscr{M}$ uses $x^1_{\mathscr{M}}$, $x^2_{\mathscr{M}}$, $x^3_{\mathscr{M}}$ and $x^4_{\mathscr{M}}$. Note that, interestingly, $W_D > 0\ \forall x^j_{\mathscr{M}}, j = 1,3,4$, reflecting the fact that, on average, agents get more than they expected to get, with $W_U > W_{parts}$. This is due in fact to the complexity of individual utility spaces and that one single agent cannot entirely explore her utility space to find all high-utility bids. By using the mediation mechanism, however, all agents' bids are filtered collectively by the mediator to find optimal bids that increase social welfare. In other words, the problem comes to appear as if agents are searching the same utility space and attempting to find all the Pareto-dominant bids. In case $W_U \leq W_{parts}$, agents get less then expected on average due to individual concessions or to bad contracts being reinforced by $\mathscr{M}$ and depending on the algorithm used, ($x^2_{\mathscr{M}}$). $W_U \neq W_{parts}$ generally corroborates the assumption of nonlinearity by reflecting the idea that the nonlinearity of the (individual) agent's utility space is propagated from the issue-constraint level up to the agreement level by means of mediation (operators $x_{\mathscr{M}}$).

The convex aspects of algorithms Eq. (10) and the nonlinearity of utility space could in fact show a lack of structure and correlation among the different SWFs in **Fig. 8**. That is to say, adopting convex representation when searching for optimal contracts is not appropriate if the utility space is highly nonlinear despite the efficiency of local optimizers (AsynchMP).

5. Conclusions

We have introduced a new representation of utility spaces based on hyper-graphs that enables the modular decomposition of constraints and issues. The exploration

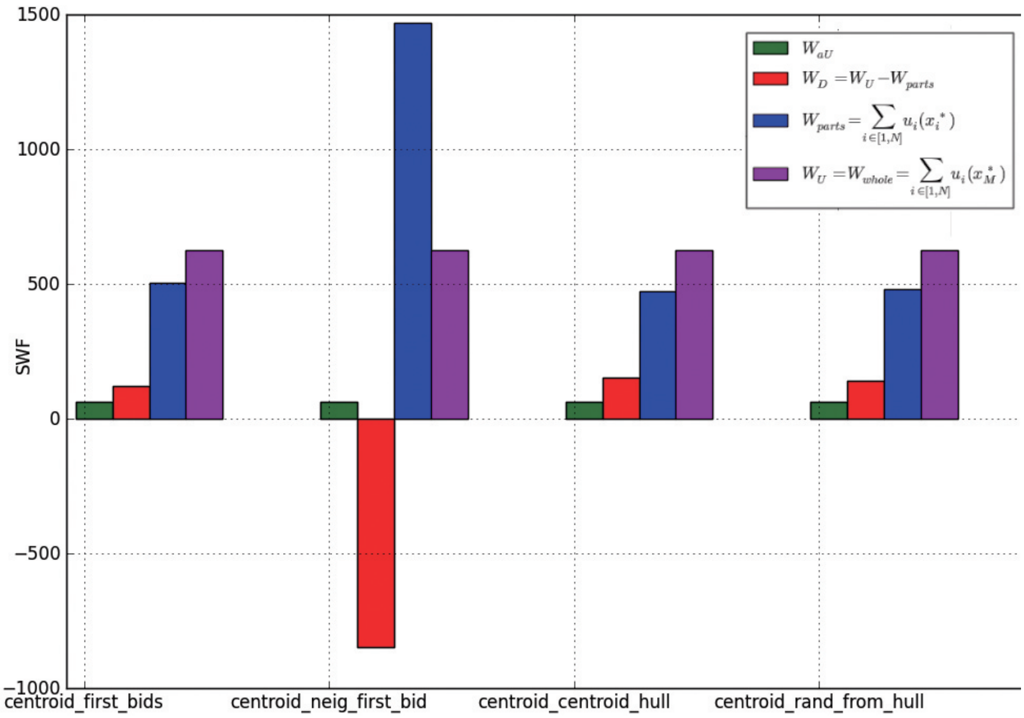


Fig. 8. Social welfare.

and search for optimal contracts is performed based on a message passing mechanism outperforming sampling based optimizers. We also evaluated the model in a multilateral setting to evaluate social welfare resulting from mediated negotiation.

In future work, we intend to exploit the hyper-graph structure to induce hierarchical negotiation. We would also like to evaluate the importance of issues within a specific domain, and to determine how one issue could affect the optimality of contracts. Being able to order issues by importance could result in a sequential negotiation model whereby issues are negotiated by importance and relevance to the domain in hand.

References:

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

[2] F. Bacchus and A. Grove, "Graphical Models for Preference and Utility," Proc. of the 11th Conf. on Uncertainty in Artificial Intelligence (UAI'95), pp. 3-10, San Francisco, CA, USA, 1995, Morgan Kaufmann Publishers Inc.

[3] V. Robu, D. J. A. Somefun, and J. L. Poutre, "Modeling complex multi-issue negotiations using utility graphs," Proc. of the 4th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), pp. 280-287, 2005.

[4] I. Marsa-Maestre, M. A. Lopez-Carmona, J. R. Velasco, and E. d. l. Hoz, "Effective bidding and deal identification for negotiations in highly nonlinear scenarios," Proc. of the 8th Int. Conf. on Autonomous Agents and Multiagent Systems – Vol.2, AAMAS'09, pp. 1057-1064, Richland, SC, 2009, Int. Foundation for Autonomous Agents and Multiagent Systems.

[5] K. Fujita, T. Ito, and M. Klein, "An Approach to Scalable Multiissue Negotiation: Decomposing the Contract Space Based on Issue Interdependencies," Pro. of the 2010 IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology – Vol.2, WI-IAT'10, pp. 399-406, Washington, DC, USA, 2010, IEEE Computer Society.

[6] J. Kwisthout and I. v. Rooij, "Bridging the gap between theory and practice of approximate Bayesian inference," Cognitive Systems Research: Special Issue on ICCM2012, Vol.24, No.0, pp. 2-8, 2013.

[7] X. S. Zhang and M. Klein, "Hierarchical Negotiation Model for Complex Problems with Large-Number of Interdependent Issues," 2012 IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology (IAT 2012), Macau, China, pp. 126-133, December 4-7 2012.

[8] X. S. Zhang, M. Klein, and I. Marsa-Maestre, "Scalable Complex Contract Negotiation with Structured Search and Agenda Management," Proc. of the 28th AAAI Conf. on Artificial Intelligence, Quebec City, Quebec, Canada., pp. 1507-1514, July 27-31, 2014.

[9] I. Marsa-Maestre, M. Lopez-Carmona, J. Carral, and G. Ibanez, "A Recursive Protocol for Negotiating Contracts Under Nonmonotonic Preference Structures," Group Decision and Negotiation, Vol.22, No.1, pp. 1-43, 2013.

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

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

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

[13] K. Arrow, "Social Choice and Individual Values," Cowles Foundation Monographs Series, Yale University Press, 1963.

[14] A. Sen, "Collective choice and social welfare," Mathematical economics texts, Holden-Day, 1970.

Name:
Rafik Hadfi

Affiliation:
Department of Computer Science and Engineering, Nagoya Institute of Technology

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

Brief Biographical History:
2012- M.Eng., Nagoya Institute of Technology
2014- Ph.D. student, Nagoya Institute of Technology

Main Works:
- Multi-agent negotiation, Consensus building, Decision making under uncertainty, Preferences elicitation

Membership in Academic Societies:
- Association for the Advancement of Artificial Intelligence (AAAI)
- Information Processing Society of Japan (IPSJ)
- The Institute of Electrical and Electronics Engineers (IEEE)

Name:
Takayuki Ito

Affiliation:
Professor, Department of Computer Science and Engineering, Nagoya Institute of Technology.

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

Brief Biographical History:
1995,1997,2000 Received the B.E., M.E, and Doctor of Engineering from the Nagoya Institute of Technology, respectively
1999-2001 Research Fellow of the Japan Society for the Promotion of Science (JSPS)
2000-2001 Visiting Researcher at USC/ISI (University of Southern California/Information Sciences Institute)
2001.4-2003.3 Associate Professor of Japan Advanced Institute of Science and Technology (JAIST)
2005-2006 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
2008-2010 Visiting Researcher at the Center for Collective Intelligence, MIT Sloan School of Management

Main Works:
- Computational mechanism design, auction, Agent-mediated electronic commerce, Multi-agent negotiation, Agent-based group decision support systems, Truth maintenance systems, and knowledge representation

Membership in Academic Societies:
- Senior Member, Association for Computational Machinery (ACM)
- Member, American Association for Artificial Intelligence (AAAI)
- Member, Japanese Society for Artificial Intelligence (JSAI)
- Member, Information Processing Society of Japan (IPSJ)
- Member, The Institute of Electronics, Information and Communication Engineers (IEICE)
- Member, Japan Society for Software Science and Technology (JSSST)
- Member, Society of Instrument and Control Engineers (SICE)
- Member, Japanese Economic Association (JEA)