Decision Support Systems 60 (2014) 39–54

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

From problems to protocols: Towards a negotiation handbook

Ivan Marsa-Maestre [a,*], Mark Klein [b], Catholijn M. Jonker [c], Reyhan Aydoğan [c]

[a] Computer Engineering Department, University of Alcala, 28805 Alcala de Henares, Madrid, Spain
[b] Center for Collective Intelligence, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA
[c] Interactive Intelligence Group, Delft University of Technology, Delft, The Netherlands

ARTICLE INFO

Available online 4 June 2013

Keywords:
Automated negotiation
Scenario metrics
Scenario generation
Testbed framework
Negotiation repository

ABSTRACT

Automated negotiation protocols represent a potentially powerful tool for problem solving in decision support systems involving participants with conflicting interests. However, the effectiveness of negotiation approaches depends greatly on the negotiation problem under consideration. Since there is no one negotiation protocol that clearly outperforms all others in all scenarios, we need to be able to decide which protocol is most suited for each particular problem. The goal of our work is to meet this challenge by defining a "negotiation handbook", that is, a collection of design rules which allow us, given a particular negotiation problem, to choose the most appropriate protocol to address it. This paper describes our progress towards this goal, including a tool for generating a wide range of negotiation scenarios, a set of high-level metrics for characterizing how negotiation scenarios differ, a testbed environment for evaluating protocol performance with different scenarios, and a community repository which allows us to systematically record and analyze protocol performance data.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Automated negotiation represents an important class of decision support system, helping self-interested decision makers rapidly reach high-quality agreements [16,47,54]. It has been studied extensively in e-commerce settings [1,36,42] but can also be seen more generally as a paradigm for solving coordination and cooperation problems in complex multi-agent systems, e.g. for task allocation, resource sharing, or cooperative planning and design [21,51,52,59]. The protocol used during the negotiation determines the rules of the negotiation. It determines when parties can make offers, how long offers stand, whether or not there are deadlines, or a fixed set of rounds, and all other "rules of encounter" [48]. A wide range of negotiation protocols have been explored in research to date, ranging from auctions to iterative concession to mediated single- and multiple-text protocols, including both bilateral and multilateral variants [15,26,34,35,43,46,49].

A critical challenge facing negotiation research is that no protocol is the best choice for every possible negotiation scenario. Protocols that work quickly and well for simple negotiations with a few independent issues, for example, fare poorly when applied to complex negotiations with many interdependent issues [33]. Some protocols may be fast but produce less-than-optimal results, and others the opposite. The question then becomes, *which* protocols should be used for *which* negotiation problems.

The research community has been unable to answer this key question, to date, because the different efforts all tend to test their protocols using their own idiosyncratic negotiation scenarios, varying widely across dimensions that include the number of agents, size of the contract space, degree of issue interdependency, and so on. It's been like comparing apples and oranges. This problem is exacerbated by the fact that there currently is no systematic enumeration of what types of negotiation challenges exist, nor do we have standard tools for identifying the type of any given real-world negotiation problem.

Our work aims to address this challenge directly, and enable a much more systematic and cumulative effort by the negotiation research community. To achieve this, we have been creating the infrastructure depicted in Fig. 1.

This infrastructure consists of the following components:

- a *scenario generator* that creates a range of test scenarios
- scenario *metrics* that can be used to characterize key properties of such scenarios
- a *testbed* for evaluating protocols with these scenarios
- *repositories* that allow researchers to upload test scenarios, in addition to experimental results on how well their protocols performed on these scenarios, for use and analysis by the larger community.

Researchers can use standard test scenarios already existing in the repository, or upload their own for use by others. Researchers can also use the testbed to evaluate their protocols, or do their own evaluation and simply upload the results to the experiment repository. The experiment repository can then be data-mined by any interested party to find correlations between scenario properties and protocol performance. As this process continues, the research community can converge upon a *negotiation handbook*, suited for use by negotiation

* Corresponding author.
E-mail address: ivan.marsa@uah.es (I. Marsa-Maestre).

0167-9236/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.dss.2013.05.019

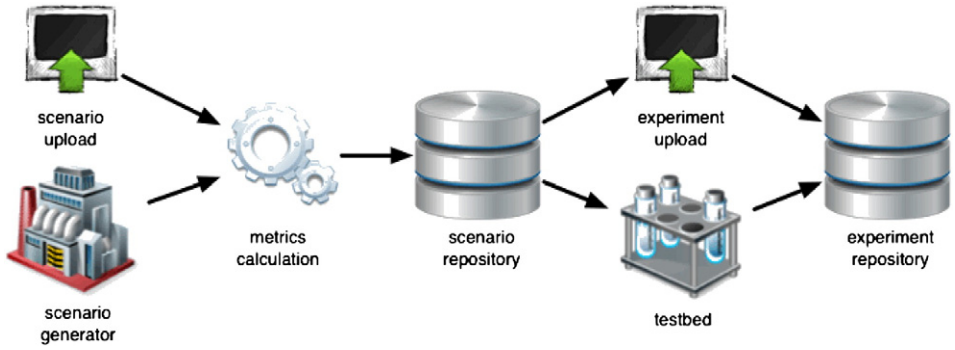


Fig. 1. The negotiation handbook infrastructure.

practitioners, that specifies design rules concerning which protocols to use for which kinds of problems.

We describe, in the remainder of this paper, the progress we have made towards achieving these goals. We begin by defining what we mean by a negotiation scenario (Section 2), describe the algorithms we are developing to generate test scenarios (Section 3), review the metrics we have developed to capture key properties of such scenarios (Section 4), and describe the protocol testbed (Section 5) and repository elements (Section 6) that are used to gather empirical data about the relationships between scenario properties and protocol performance. We conclude with a review of the unique contributions of this work and possible directions for future work.

2. What is a negotiation scenario?

A negotiation scenario should specify everything we need to know about a negotiation problem in order to pick the right protocol for the job. These elements include:

- The *domain* i.e., the range of contracts being negotiated over, which is defined by the set of issues being negotiated over as well as the valid values for each issue. The domain for a purchase negotiation, for example, would probably include *price* (whose values are positive real numbers) and *delivery date* (whose values are dates). A *contract* specifies a single valid value for every issue in the domain, and an *agreement* is generally the contract that all negotiating parties agree upon. Key properties of a domain include the *types* of the issues (e.g., continuous vs. discrete) as well as the *size* of a domain (i.e., the number of possible contracts).
- The number and characteristics of the *players* involved in the negotiation, including:
  - *Agent preferences*, which specify how highly an agent values any particular set of issue values. Key factors here include the *types* of the agent preferences (e.g., ordinal vs. cardinal, fixed vs. mutable), *reservation values* (i.e., the threshold, possibly time-sensitive, for deciding whether or not to accept an agreement), as well as the *preferential dependencies* (i.e., whether the agent's preferences for one issue are influenced by the choices made for other issues). An agent's preferences may also include such factors as retaining goodwill with some other agents because they will likely negotiate again in the future.
  - *Agent decision-making style*, such as the agent's *risk attitude* (i.e., its willingness to risk poor outcomes in the pursuit of better ones) [12], *cognitive style* (i.e., whether it bases its decision on economic rationality, emotion, or other factors) [44], *cognitive capability* (i.e., whether agents have sufficient computational capability to pick the best negotiation moves in the time available), *truthfulness* (i.e., whether agents choose to report their preferences in a truthful vs. a manipulative way) [13], and characteristic *strategies* (e.g., some agents may adopt a Boulware strategy, wherein they concede as slowly as possible towards the other agents' proposals, until the negotiation deadline comes close). An agent's strategy, of course, will be deeply influenced by the protocol being used.
- The *context* of other negotiations. An agent, for example, can use knowledge of previous similar negotiations to predict the other agents' preferences and styles (e.g. it can predict that buyers always want to maximize price), which can in turn inform its own decision-making during a negotiation. Furthermore, there might be other negotiations going on at the same time for related topics, and the progress in the negotiations could mutually influence each other.

These elements define a large space, and many dimensions of this space are as yet incompletely understood. The set of possible agent strategies, for example, is large, and possibly unbounded. Accordingly we have chosen to focus, in our initial efforts, on characterizing the two most basic elements of any negotiation: the *domain* being negotiated over, and the *preferences* of the agents doing the negotiation. Since we have already discussed how to represent a negotiation domain, we focus in the discussion below on how to represent agent preferences.

3. Agent preference representations

There are many different ways to represent agent preferences. Perhaps the simplest is the *ordinal graph*, which consists of nodes (each representing one possible agreement) connected by links that indicate that the source contract is preferred, by the agent, to the target contract. Ordinal graphs are fully expressive (they can capture any internally consistent set of preferences), and can handle cases where only partial preference information is available (i.e. where some preference orders are not yet known), but they capture only *ordinal* preferences, providing no information on *how much* one contract is preferred over another. Ordinal graphs also scale poorly; capturing the preferences for a domain with 10 issues, 10 values per issue, would require a graph with 10 billion nodes.

If we wish to capture *cardinal* preferences (i.e. where we can specify how much an agent prefers one contract over another), we move into the realm of *utility functions*. A utility function is defined, for each agent *j*, as

$$U^j : \mathrm{D} \rightarrow \mathbb{R}.$$

It assigns, to each possible agreement in the domain $\mathrm{D} = \prod_{i=1,\ldots,n} d_i$, a real number representing the utility that agreement yields for agent *j*. Note that cardinal preferences *subsume* ordinal preferences, since we can always derive the preference order of two contracts by comparing their cardinal utility values.

The simplest form of cardinal utility function, and the one most widely used in the negotiation literature, is the *additive utility function* [9,46]. In such functions, an agent's utility for a contract

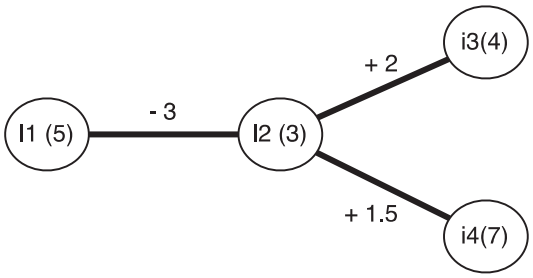


Fig. 2. An example utility graph, adopted from [47].

is calculated as the weighted sum of the agent's utility for each issue X_i:

$$U(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} w_i * U_i(X_i).$$

Additive utility functions provide a compact preference representation, but they are unable to capture the important phenomenon of *preferential dependency*, i.e., where the utility of the choice for one issue is dependent upon the choices made for other issues. Preferential dependencies are ubiquitous in real-life negotiations. When we negotiate over the stereo system we are going to purchase, for example, the utility of any particular issue value choice (e.g. which tuner we select) depends fundamentally on the choices made for other issues (e.g. whether we selected speakers that are compatible with that tuner). Utility functions without preferential dependencies always have a simple *linear* structure, with a single optimum, while utility functions with preferential dependencies can have a much more complex, *nonlinear*, structure, with multiple optima.

There is a range of utility function representations suited for dealing with preferential dependencies. *Auction preference languages* [5,10] capture preferential dependencies in the context of allocation negotiations (i.e. where there are only binary issues, each representing whether an agent acquires a particular good or not). The preference language expression:

$$\langle\langle a \wedge b, 0\rangle \vee \langle c, 5\rangle, 50\rangle$$

for example, expresses the idea that the agent assigns a value of 50 to acquiring either goods *a* and *b*, or good *c*, and it assigns a premium of 5 to the latter.

Utility graphs [47] capture allocation preferences as a set of nodes (each representing the issue of whether or not a given good was purchased) in addition to a set of links between these nodes that capture the (positive or negative) complementarities between the goods. In the utility graph in Fig. 2, for example, there are four goods as well as three dependency relationships.

The utility of a given contract, to an agent, is given by the sum of the values for goods acquired, plus the sum of the values for the complementarity links whose source and target goods were both acquired. In this graph, for example, the contract [I1, I2, I3] (i.e. where the agent acquires the goods I1 I2 and I3) has the utility $5 + 3 + 4 + -3 + 2 = 11$. Utility graphs only capture binary preference dependencies (i.e. dependencies between pairs, but not higher-order tuples, of issues).

While allocation negotiations are of course very common, especially in e-commerce contexts, there are many real-world negotiations with N-ary issues that concern more than just whether the agent acquires a given good, or not. When engineers are negotiating over the design of a car, for example, many of the issues (e.g. What kind of engine should we have? What kind of transmission? How many passengers should we make room for?) will have more than two possible values.

CP-nets [7] capture preferential dependencies for N-ary issues using directed graphs in which each node represents the agent's preference for an issue, and each link captures the impact one issue choice has on the preferences for another. In the CP-net example in Fig. 3a, for example, there are three issues involved in a vacation negotiation:

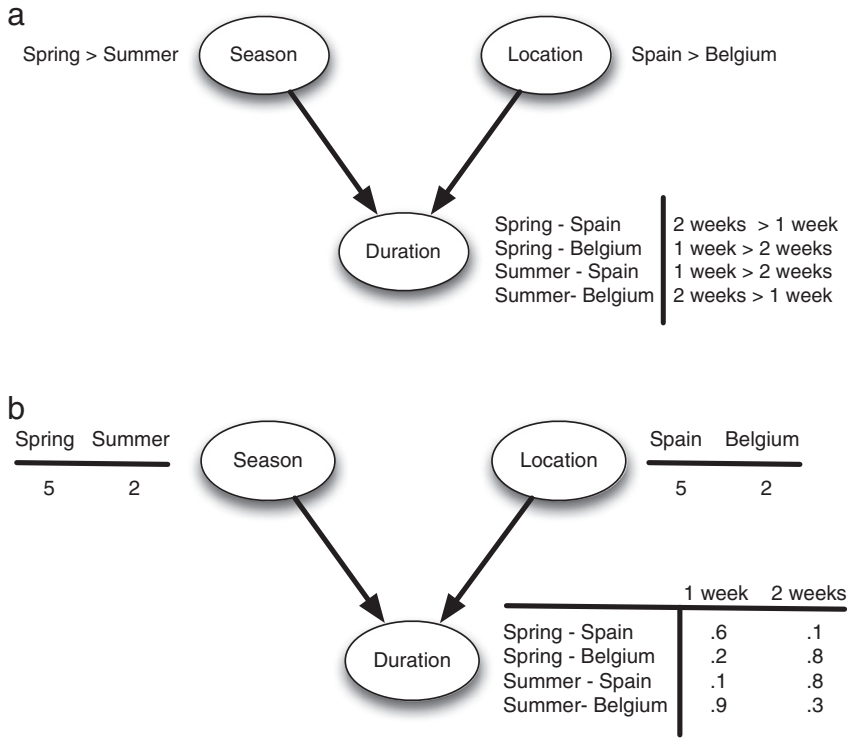


Fig. 3. a. An example of a CP-net. b. An example of a UCP-net.

season, location, and duration. The agent prefers Spring over Summer, Spain over Belgium, and has preferences for duration that depend on the selected season and location.

CP-nets can be used with N-ary issues (i.e. where an issue can have more than just two possible values), and can be used when we do not fully know an agent's preferences, but they capture only ordinal, as opposed to cardinal, utility information. Cardinal utilities can only be *estimated*, in this representation, by applying heuristics such as those described in [2].

UCP-nets [6] address this challenge by including utility values, rather than simply preference orderings, with each node in the network. In the UCP-net in Fig. 3b, for example, the utility for the agreement [Spring, Spain, 2 weeks] is 5 + 5 + .1 = 10.1.

K-additive utility functions [18] are a generalization of additive utility functions, where the utility of a contract for an agent is the sum of the agent's value for each issue choice *plus* the values due to the preferential dependencies between two (or more) issues. The following example:

$$U(j) = U(<i1>, j) + U(<i2>, j) + U(<i1, i2>, j)$$

calculates the utility for agent *j* of a contract as the sum of the values of the choices for issues i1 and i2, *plus* the value due to the dependency (synergistic or antagonistic) between those two issues. This approach can be applied to preferential dependencies between any number of issues.

The *weighted hypercube* approach [25] represents a utility function as a collection of hypercubes that each assigns a numeric weight to a sub-volume of the domain of possible contracts:

utility_function: = (ufun <weighted_hypercube>*)
weighted_hypercube: = (hypercube <weight> + <bound>*)
bound: = (<issue> <min value> <max value>).

The utility for a given contract is then calculated as the sum of the weights for the hypercubes that *include* that contract, which means the contract is within all the bounds the hypercube imposes to the different issues. Imagine we have a scenario with three issues, each of which can have a value between 1 and 20. Imagine, further, that the utility function for an agent consists of the following weighted hypercubes:

Weight	Bounds
10	(Issue1 from 2 to 4)
12	(Issue2 from 3 to 6) (issue3 from 12 to 17)
7	(Issue1 from 1 to 5) (issue2 from 7 to 10) (issue3 from 10 to 12)

The contract [issue1 = 5, issue2 = 4, issue3 = 11] is included in the second and third hypercubes, so the utility of that contract is 12 + 7 = 19. This representation is capable of representing any imaginable utility function with a discrete domain; at worse, we simply need to create unit-volume hypervolumes for every contract in the domain.

We can summarize these different utility representations using the following table:

	Binary issues (allocation negotiations)	N-ary issues (general negotiations)	
No preferential dependencies		Additive utility functions	
Binary preferential dependencies	Utility graphs		
N-ary preferential dependencies	Auction preference languages	*Ordinal* Ordinal graphs CP-nets	*Cardinal* UCP-nets k-Additive functions Weighted hypercubes

The most expressive utility representations are the three (in the lower right cell) which support N-ary issues, N-ary dependencies, and cardinal utilities. These three representations are equivalent in terms of expressiveness, since they are theoretically capable of representing any well-defined utility function for discrete issue domains.

We adopted, in our work, an extended version of the weighted hypercube representation, wherein the hypervolume weights can be a constant, linear, *or* sine function of the issue values:

scenario: = (scenario:name <string>
:description <string>
:domain <domain>
:agents (<agent>*))
domain: = (<def>*)
def: = (issue <string> real from <number> to <number>) |
(issue <string> integer from <number> to <number>) |
(issue <string> one-of <string>*)
agent: = (agent:name <string>:ufun ufun)
ufun: = (<hypervolume>*)
hypervolume: = (cube (<bound>*) <weight>) |
(plane (<bound>*) (<weight>*) <bias>) |
(sine (<bound>*) <height>) |
bound: = (limit <string> from <number> to <number>) |
(limit <string> one-of <symbol>*).

This representation can capture any imaginable utility function for both discrete *and* continuous domains, since we know that any waveform can be represented with arbitrary accuracy (using the Fourier transform) as the sum of sine waves [8]. Fig. 4 presents two examples of utility functions (for a two-issue contract domain) that can be generated using a weighted hypervolume approach.

The utility functions have high regions where many high-weight hypervolumes reside, and lower regions where there are few and/or low-weight hypervolumes.

4. Scenario generation

How can we acquire the scenarios we need to test different negotiation protocols? One obvious approach is to simply collect the scenarios that have been used in previous negotiation research and store them in our scenario repository for general use (we describe how this can be done in Section 6: "Repositories"). A key limitation with this approach, however, is that there is a vast number of possible negotiation scenarios, and previous research has necessarily only covered a tiny fraction of this space, predominantly scenarios without preferential dependencies. We need, therefore, a way to cover the scenario space more comprehensively. This is the purpose of the scenario generator. It allow users to generate negotiation scenarios by systematically varying a compact set of high-level parameters, including (1) the *negotiation domain*, including the issues and possible issue values, (2) a *scenario utility histogram* (see below), which defines the distribution of contracts in utility space, and (3) several *structural parameters* that specify the topography (e.g. ruggedness) of each agent's utility function. Our scenario generator then creates, using a stochastic process, utility functions (sets of weighted hypervolumes) for each agent that satisfy the user's specifications. We describe this process in the sections below.

4.1. Specifying scenario utility histograms

Utility histograms are a generalization of utility diagrams. The latter provide a simple way to visualize the possible agreements between negotiating agents. In such diagrams, there is one axis for

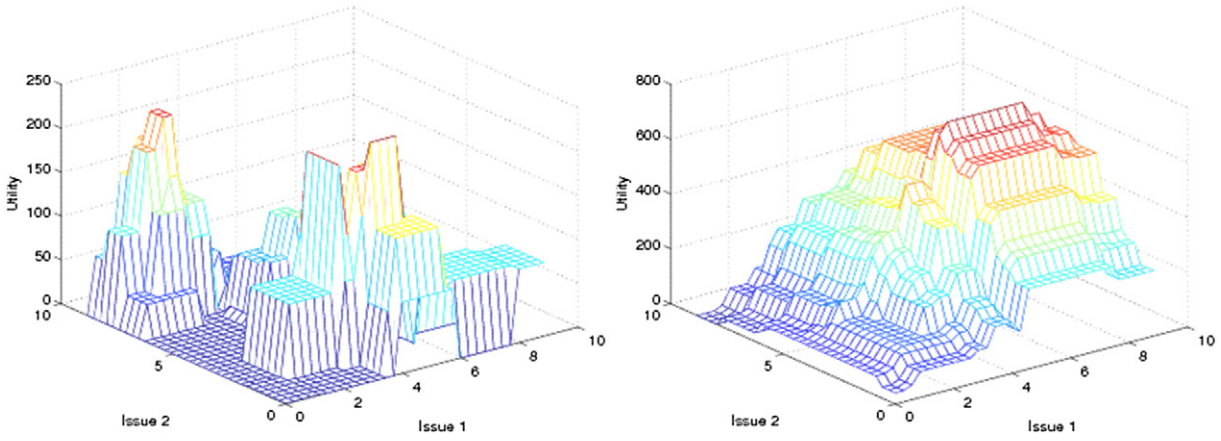


Fig. 4. Examples of utility functions expressed using weighted hypervolumes.

every negotiating agent, and one point per possible agreement. The position of a point represents the utility of an agreement for each agent. In the (two-agent) utility diagram in Fig. 5a, for example, the y axis represents the utility of an agreement to agent A, and the x axis represents the utility of an agreement to agent B. The Pareto front represents agreements that cannot be improved upon for one agent without sacrificing utility for other agents. The optimality of any proposed agreement can be assessed by measuring its distance from the Pareto front.

The problem of scenario utility diagrams is that they do not capture the multiplicity of agreements. That is, if we have, for instance, one thousand contracts which give the same utility values for the agents, they would be represented in the diagram as a single point. This multiplicity, however, may be highly significant in negotiation. For instance, the scenario utility diagram in Fig. 5a may correspond to a negotiation scenario where 80% of the contracts are within the Pareto front, or to an scenario where only 1% of the contracts are, both scenarios being quite different in terms of potential impact on protocol performance. Taking this into account, we generalized scenario utility diagrams into scenario utility *histograms* where each point in the diagram is assigned a number, representing either the number of contracts in the domain which yield the corresponding utility values to the agents or the ratio of contracts relative to the volume of the domain. The former can be used only with discrete domains, while the latter can be used with both discrete and continuous domains. Fig. 5b shows a utility histogram for two agents. For a number of agents $n_a > 2$, we can represent the utility histogram as a set of $(n_a + 1)$-dimensional vectors, where the first n_a dimensions represent the utility values u_i for every agent i, and the last dimension represents the number or volume v of contracts corresponding to these utility values. We can also use a functional notation $H(\overline{u}) = v$ to refer to the different points in the histogram.

Specifying scenarios in terms of their utility histograms has two great advantages. One advantage is that it represents an excellent way, as we shall see, of manipulating important properties of the scenario. We can specify, for example, whether the negotiation is highly competitive (the Pareto front is flat or even concave) or not, what proportion of the contracts are close to the Pareto front (making a good contract easier to find), and so on. The other advantage of this approach is that the Pareto fronts for our scenarios are known up-front, making it easy to get an exact value for the optimality of the agreements negotiated for that scenario. Negotiation researchers typically start by defining utility functions for the individual agents, and then *estimating* the Pareto fronts using an optimization approach such as genetic algorithms. If the contract space is large, and there are preferential dependencies (which as we've noted create multi-optima utility functions) then the Pareto front estimate may be quite inaccurate. Our approach obviates this whole problem by deriving the agent utilities from the Pareto front itself, as opposed to the other way around.

Our scenario generator provides a simple way for users to specify scenario utility histograms, based on a small set of high-level parameters. The user first defines the Pareto front shape as a set of *weighted hyper-spheres*, each which we can define as follows:

$$\sum_{i=1}^{n_a} \left(u_i^{\alpha_i}\right)^2 = 1$$

where n_a is the number of negotiating agents, u_i is the utility obtained by agent i and α_i is a weighting parameter which allows to control the shape of the Pareto front. In particular, as seen in Fig. 6(a) to (c), giving the same weights to all agent (that is, $\propto_i = \propto_j \forall\, i,j$) allows to control the concavity or convexity of the Pareto front, and thus the competitiveness of the scenario (see Section 4). Note that we can theoretically model all ranges of competitiveness, including the zero-sum game ($\propto_i = 0.5\ \forall\ i$).

In a similar way, as shown in Fig. 6(d), giving different weights to different agents allows us to adjust the symmetry of the Pareto

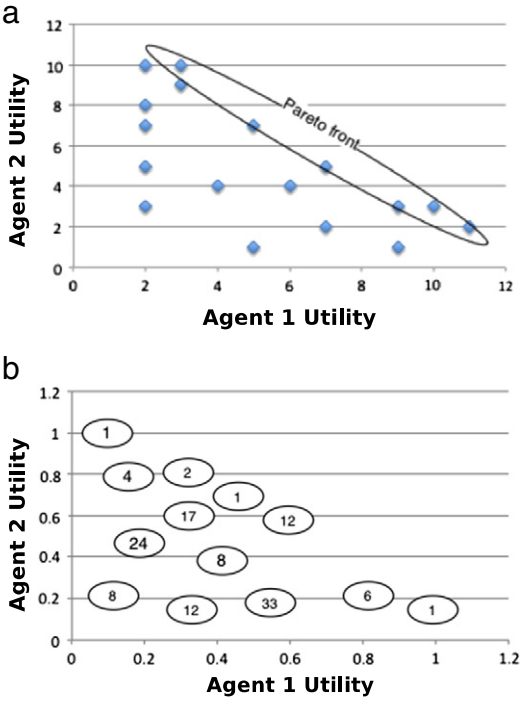


Fig. 5. a. An example of a scenario utility *diagram*. b. An example of a scenario utility *histogram*.

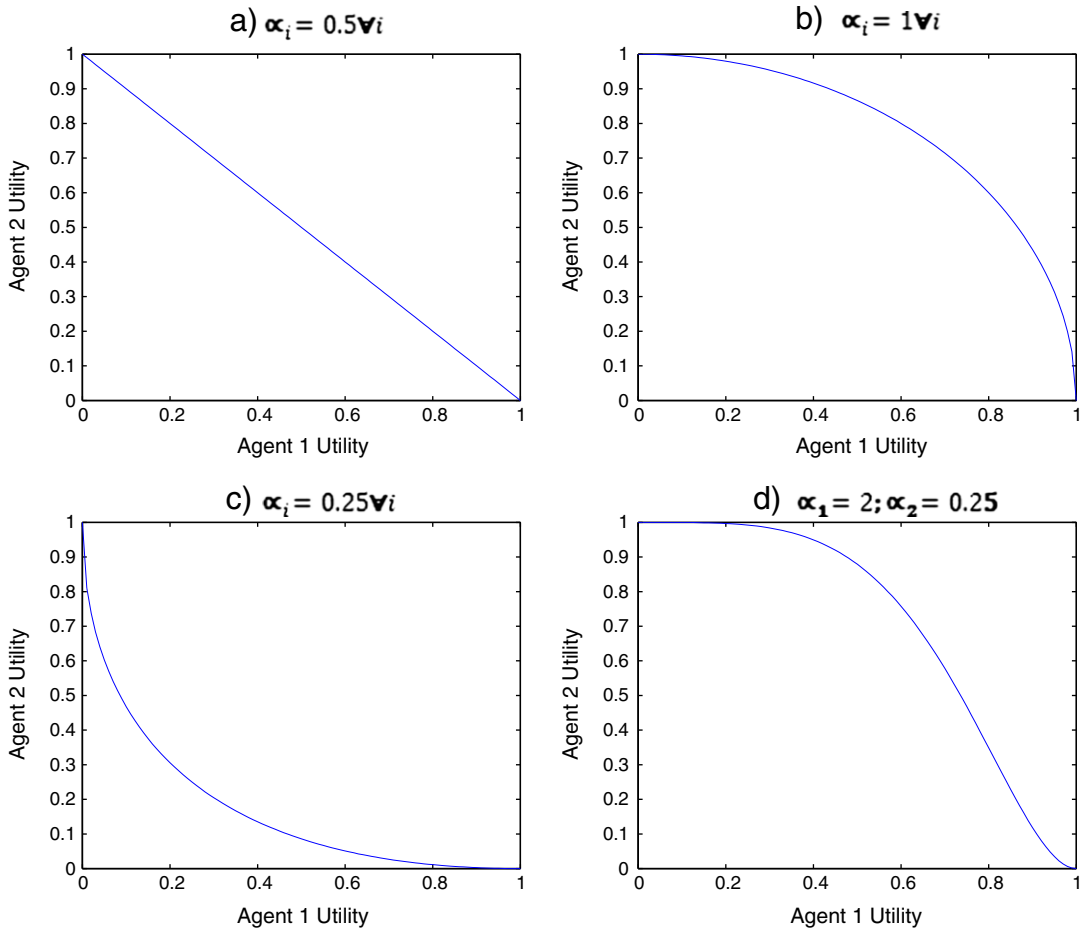


Fig. 6. Pareto fronts generated as 2-dimension weighted hyper-spheres. (a) $\propto_i = 0.5\ \forall\ i$ (b) $\propto_i = 1\ \forall\ i$ (c) $\propto_i = 0.25\ \forall\ i$ (d) $\propto_1 = 2; \propto_2 = 0.25$.

front, thus varying the symmetry of the scenario. In particular, the different weights given to the utilities of the agents may be seen as modeling their bargaining power, as suggested in [37]. More complex Pareto fronts may be obtained by aggregating different weighted hyper-spheres, though in these cases the absence of non-dominant points in the resulting front cannot be assured, so residual points need to be removed after generation. Fig. 7 shows an example of such a Pareto front. The figures show continuous Pareto Fronts, though our generator uses discrete utility histograms for scenario generation. To discretize the Pareto fronts, the points have been generated uniformly throughout the hyper-sphere, using a generalization of the method in [35].

Once we have generated the Pareto fronts, we need to generate the rest of the points in the scenario utility histogram. To do this, we apply different probability distributions to a generalization of the Marsaglia algorithm [40]. Fig. 8 shows the results of using a uniform and two normal distributions (one centered at the origin and the other at the Pareto front). We can see that varying the distribution used allows us, for instance, to study the effect of different utility or social welfare centroids on the negotiations, or the impact of having a high or low density of points near the Pareto front.

4.2. Specifying agent utility functions

Once the utility histogram has been defined for a scenario, our scenario generator can then define the utility function for the individual agents. Our algorithm does so using *shared hypervolumes*. The idea is to include *similar* hypervolumes (that is, hypervolumes involving the same contracts but with different shapes) in the utility functions of the different agents, adjusting the parameters of the hypervolumes so that they generate appropriate points within the scenario utility histogram. For instance, if we want to generate utility functions for a trivial scenario utility histogram $H(\overline{u})$ for two agents, where the histogram value is v for $\overline{u} = [a, b]$ and 0 otherwise, we could achieve this by generating two utility functions which share a hypercube of volume v, with weight a for the first agent and weight b for the second. Fig. 9 shows another simple example for two agents and two integer issues whose values may take values ranging from 1 to 10. In this case, the utility histogram (shown in Fig. 9a) includes four points, as follows:

- Two pareto-dominant points: $H([1,0.6]) = 16$ and $H([0.5,1]) = 18$.
- A non-dominant, nonzero utility point: $H([0,25,0.5]) = 29$.
- A point representing the rest of the contracts, which yield zero utility for both agents: $H([0,0]) = 37$.

The generation algorithm must find a set of hypervolumes which satisfy the volume values imposed by the diagram. In our example, this can be achieved using the three hypervolumes shown in Fig. 9b and c. Hypervolumes C1 and C2 create an intersection $C1 \cap C2$ of 3 by 6 (18) contracts, while the union $C1 \cap C2$ encompasses 63 contracts, leaving 37 contracts unassigned (that is, with zero utility). Finally $C3$ has 4 by 4 (16) contracts, and the area within $C1 \cap C2$ which is not inside $C1 \cap C2$ or inside $C3$ encompasses a total of 29 contracts. Once the hypervolumes have been found, the algorithm assigns the appropriate weights to the hypervolumes in the utility function of every agent so that the utility values in the histogram are satisfied. In our example, this is achieved with the weights shown in Fig. 9b and c. The resulting utility functions are shown together in Fig. 9d.

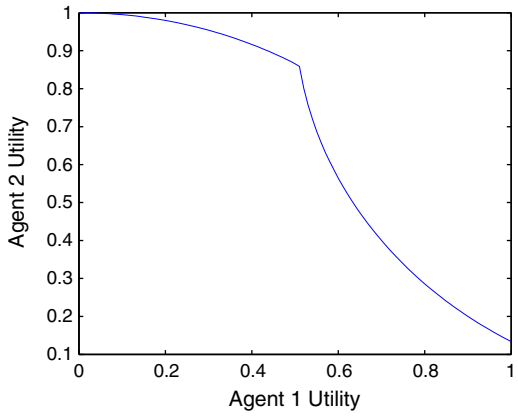


Fig. 7. A complex Pareto front generated using two weighted hyper-spheres.

Of course, as the number of points in the utility diagram increase, the complexity of the generation process also increases, since we have to take into account the effect of the intersection between shared hypervolumes. A first approximation to avoid this problem is to divide the utility space in non-overlapping regions and to assign shared hypervolumes (one for each agent) to each region, with the utility values needed to match the points in the scenario utility diagram. The problem here is that, depending on the shape of the hypervolumes we are using, the resulting scenario utility histogram may differ from the original (e.g. a bell-shaped hypervolume adds more points to the histogram apart from the one corresponding to its peak value). A solution to this problem may be to feed this first approximation of the scenario utility histogram to a nonlinear optimizer which tries to minimize the approximation error. However, the computational cost of running that kind of optimizer for large cardinality contract spaces makes it unfeasible. Other alternatives are to use hypervolumes that do not introduce approximation error (i.e. hypervolumes with a constant weight), or simply accept that there will be some approximation error. As long as the hypervolumes do not overlap, the approximation error will never affect the shape of the Pareto front.

A second downside of the non-overlapping hypervolumes approach is that the utility function representation is not very compact, since it will have at least one hypervolume per point in the utility histogram. In some cases, this can be solved by using compression techniques to convert from non-overlapping hypervolumes to overlapping ones (this works relatively well with hypercubes). However, since the purpose of the generator is not to allow for compact utility representation, but to provide a consistent comparison of negotiation scenarios, we believe that succinctness of the representation is not an issue at this point. Furthermore, the use of shared, non-overlapping hypervolumes has some advantages for the computation of scenario and utility function metrics, so this is the approach we finally adopted. This simplified algorithm works as follows:

1. For each utility histogram point $H(\overline{u}) = v$, where $\overline{u} = \{u_i | i = 1 \ldots n_a\}$ and at least one u_i is nonzero, we partition the associated volume v into a set of region volumes v_j such that $\sum_j v_j = v$, according to the desired structural properties (see Section 4). For instance, if we want utility functions with low correlation length, the partition will involve small region volumes.
2. We map each region volume v_j to a particular hypercube region in the contract space which has the appropriate volume and does not overlap any previously allocated region. In this way, we guarantee that our hypervolumes do not intersect. Again, the shape of the hypercube may be altered to satisfy desired structural properties. For instance, if we want asymmetric epistatic interactions, the hypercubes should be wider in some dimensions than in others.
3. We include the generated hypercube in the utility function of each agent i, assigning to it a weight equal to the corresponding utility value associated to that agent in $\overline{u}$, u_j.
4. The algorithm progresses until all volume has been allocated.

An important property of this scenario generation strategy is that the shape and parameters of the shared hypervolumes may be varied so that additional properties of the generated functions are satisfied. Fig. 10 shows an example for two agents and weighted hypercubes, where we have generated two scenarios with identical utility histograms and different correlation lengths. The type of hypervolumes or the aggregation operators used may be adjusted as well, but in these cases we have to take into account the effect that changes may have on the resulting scenario utility histogram.

5. Metrics calculation

In this section we discuss the metrics we have developed to capture the "key properties" of negotiation scenarios, i.e. those that help predict which protocol(s) will work best for which scenarios. Our metrics are divided into three categories: (1) Structural metrics, which capture properties of the individual agents' utility functions, (2) Relational metrics, which capture how the negotiating agents' utilities relate to each other, and (3) Contextual metrics, which capture properties of the context in which the negotiations occurs. We describe these metrics in the sections below.

5.1. Structural metrics

Our structural metrics attempt to capture how difficult it is to find high-utility contracts in a given agent's utility function. Many such metrics have been defined in such fields as artificial intelligence, operations research, and evolutionary computing [57].

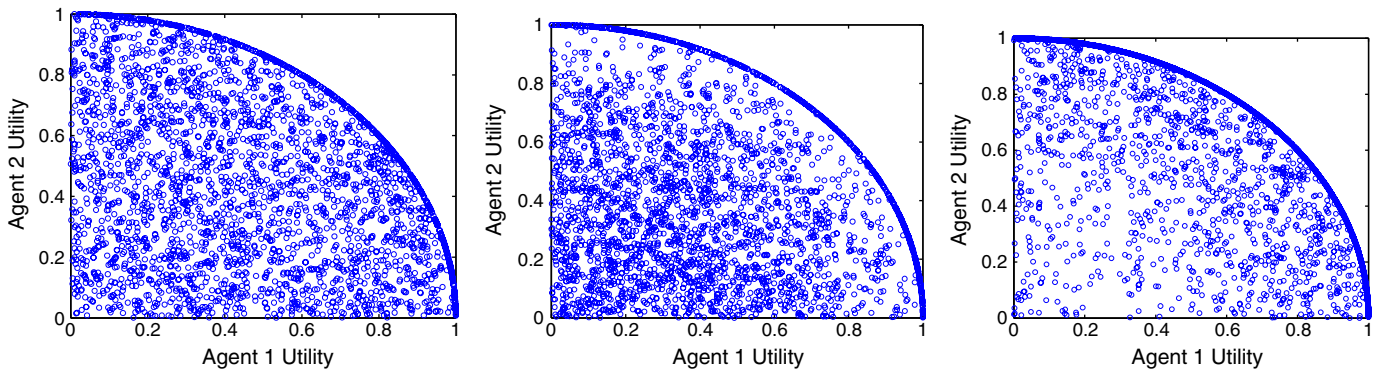


Fig. 8. Scenario utility histograms for the same Pareto front, using probability distributions that are (a) uniform, (b) favor points away from the Pareto front, and (c) favor points near the Pareto front.

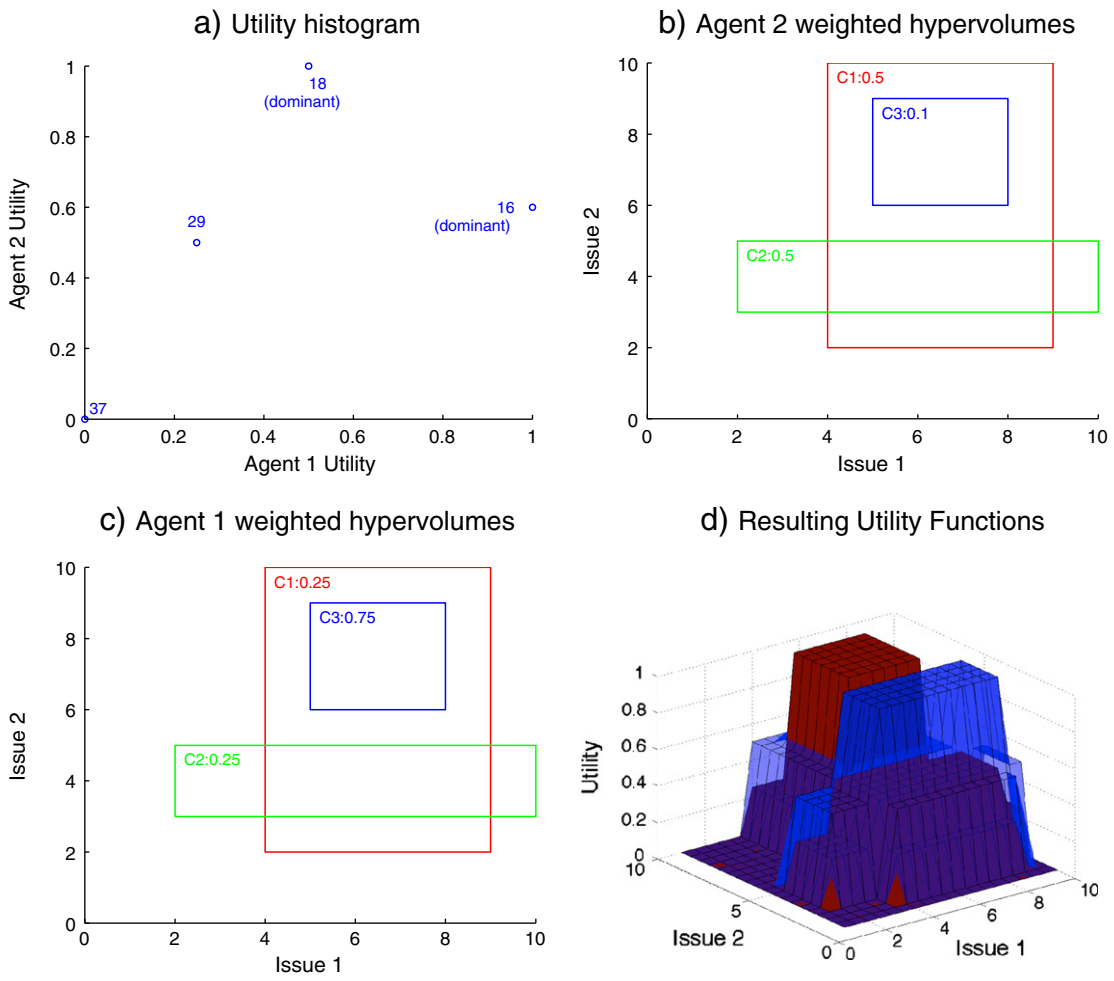


Fig. 9. Example of scenario generation from utility histograms.

Perhaps the simplest metric is based on the skew of agent's individual utility histograms. Consider two histograms in Fig. 11, giving the number of contracts at each utility value within a given agent's utility function.

The utility function whose histogram is shown on the left is probably harder to optimize because most of its contracts have a relatively low utility, while the one of the right is probably easier to optimize.

Another metric is *modality variance*, which measures the variance in the heights of the local optima in the utility function. In Fig. 12, for example, the modality variance is low in the utility function on the left, and higher in the utility function on the right.

The intuition behind this metric is that, if the modality variance is zero, an agent can simply hill-climb from any random point to find an optimal contract, since every local optimum is also globally optimal. As the modality variance increases, agents are increasingly likely to get stuck in poor local optima.

Our *ruggedness* metric assesses how "bumpy" a utility function is, and therefore how many local optima a search algorithm is likely to encounter when looking for the global optimum. Ruggedness can be assessed using *fitness distance correlation* (FDC) [28,56], which measures the correlation between the utility of a contract and it's Euclidean distance, in the contract space, from the global optimum. If there is a strong correlation between distance and utility, this implies a smooth utility function. If the correlation is weak, this implies a rugged utility function with many "bumps" along the way. Consider the examples in Fig. 13.

The top charts in the figure show two different utility functions in a single-issue contract domain, and the bottom charts show utility as a function of a contracts' distance from the globally optimal contract. The left utility function is smooth, and shows a strong correlation between utility and distance (a high FDC). The right utility function is more rugged, and has a correspondingly lower FDC value. The lower the FDC (in absolute value), the more local optima exist and the more difficult it will probably be to find the global optimum. A related metric is the *correlation length* (AKA *correlation distance*) which is defined as the minimum contract space distance which makes the utility correlation between contracts pairs fall below a given threshold (usually 0.5) [39,41].

Our *information content* metric takes a different tack, assessing the information content of the utility space based on the intuition that the higher the information content of a utility function, the more difficult it will be to perform a search in that utility space. The extreme cases would be a totally flat utility space (minimum information content, a single sample gives information about all the space) and a totally random utility space (maximum information content). Finding an optimal contract in a random utility space would of course be much harder, because you can't use slope information to guide your search at all. Information content can be assessed by measuring high-frequency components of the utility functions, for instance by using a n-dimensional Fast Fourier Transform (FFT) [8]. For solution spaces which are too large to allow for comprehensive sampling, we can use *random walk* entropy measurements, where a random walk is performed throughout the utility space, and the entropy of the utilities of the contracts in the walk is measured.

Other metrics assess how effective optimizers are at locating global maximums in the utility functions. One example is the *hill-climbing success ratio*, which measures how often a hill-climbing optimizer can

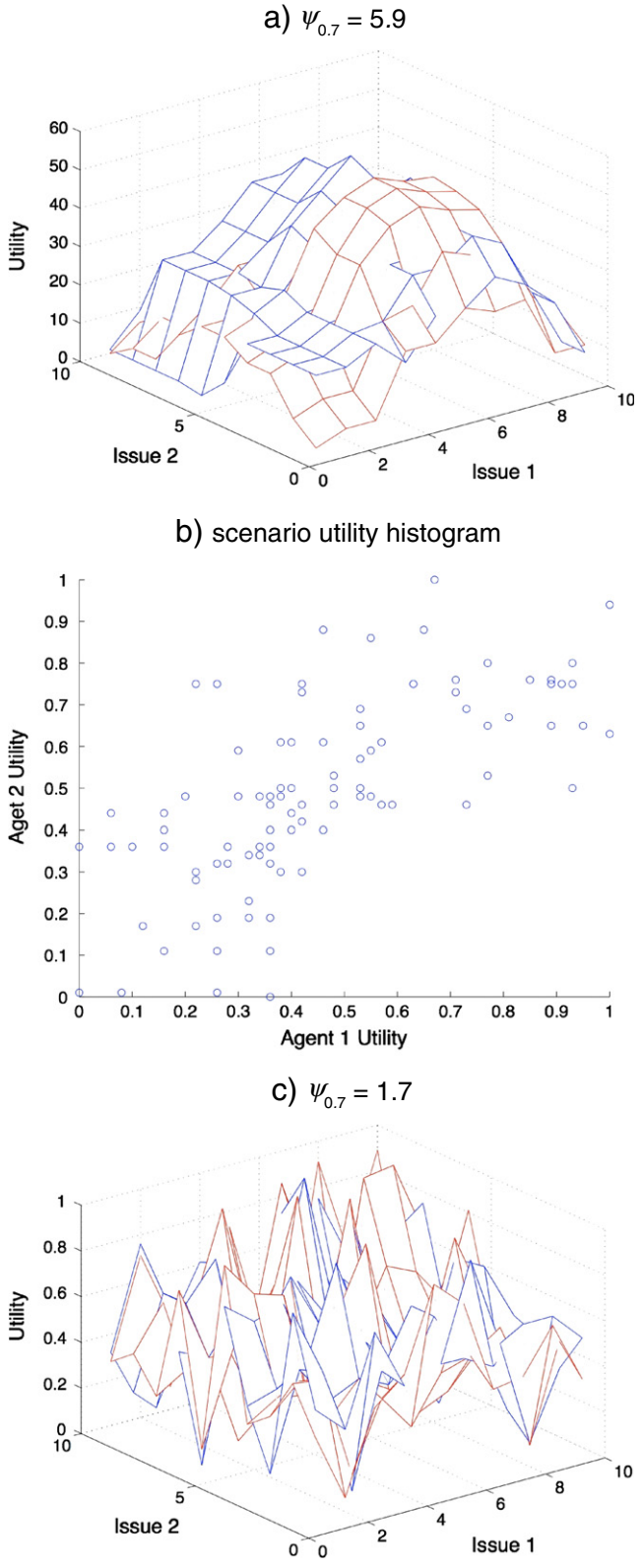


Fig. 10. Utility functions from the same utility histogram but with different correlation lengths.

find optimal contracts within an agent's utility function. The *minimum annealing temperature* metric is a variant that assesses the minimum starting temperature needed for simulated annealing-based optimizer [31] to achieve a given success ratio. The higher the temperature is needed, the more rugged and challenging the utility functions.

A final metric assesses the degree of preferential interdependency. Utility functions with preferential dependencies create multi-optimum topographies that are typically much harder to optimize than those involving independent issues [33]. Preferential dependency can be assessed using an information-theoretic measure known as *epistasis* [58], originally developed in the context of genetic biology. It is defined as follows:

$$\epsilon_{ij} = \begin{cases} \dfrac{I(i;U)+I(j;U)}{I(i,j;U)} - 1 & \text{if } I(i,j;U) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where $I(i;U)$ and $I(j;U)$ are the amount of information that each issue i and j reveal about the value of the utility function U, and $I(i,j;U)$ is the amount of information *both* issues reveal about the value of U. An alternative technique to assess interdependency relationships between issues is to perform a Walsh analysis of the utility function [20].

Structural measures can be less meaningful in *anisotropic* utility functions where there are significant local variations in geography (e.g. where the utility function is smooth in some regions, and rugged in others) [19,53]. In this situation, we can calculate these measures separately for different regions of the space, using local optimum for example to delineate the centroids for the different regions.

5.2. Relational metrics

We have also defined a set of metrics that assess *relationships* between the utility functions of the agents involved in a negotiation. Relational metrics capture important properties that are not obvious from examining any one agent's utility function. We may have, for instance, scenarios where agents have highly rugged utility functions that make it difficult for them to determine their own high utility regions, but where once these regions have been found agreements are fairly straightforward, because high utility regions for the different agents coincide. On the other hand, we may have scenarios where the agents have smooth easy-to-optimize utility functions, but where *mutually acceptable* regions are nevertheless hard to find.

It should be noted that relational metrics will be especially critical in domains with strong preferential dependencies. When there are no preferential dependencies, utility diagrams have simple symmetric structures, with convex Pareto fronts, and differ mainly in whether the agent utility functions have strongly negative (zero-sum) correlations, strongly positively (win–win) correlations, or something in between (see Fig. 14).

In this context, many of our relational metrics will do little to distinguish between negotiation scenarios. This picture changes completely, however, when we consider "nonlinear" utility functions with strong preferential dependencies. As we can see (Fig. 15), when preferential dependencies are introduced, the utility diagrams can become much more complex and diverse, with non-symmetric structures and concave Pareto fronts, and capturing key dimensions of variation between such scenarios requires a richer set of metrics.

Our *goodness* metrics assesses the proportion of high-utility agreements in the contract domain. This includes *the ratio of high-social-welfare contracts* (which measures what fraction of the contracts have top-quartile social welfare), the *ratio of mutually-acceptable contracts* (which measures what fraction of the contracts lie above the reservation values for the agents) and the *ratio of Pareto-efficient contracts* (which measures what fraction of the contracts are within a distance ϵ of the pareto front). A scenario where these ratios are small will probably be more challenging for a negotiation protocol.

Our *symmetry* metric assesses the symmetry of the utility histogram across the equal utility line. Some scenarios may be inherently asymmetric, in the sense that there may be few or no contracts that offer equally high utility values for all the negotiating agents.

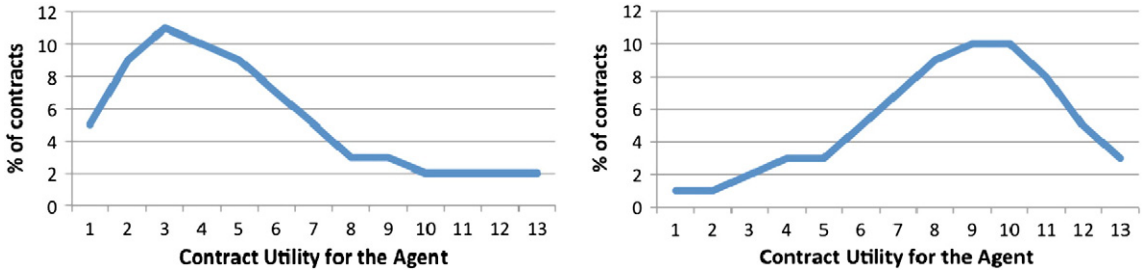


Fig. 11. Agent utility histograms with differing skews.

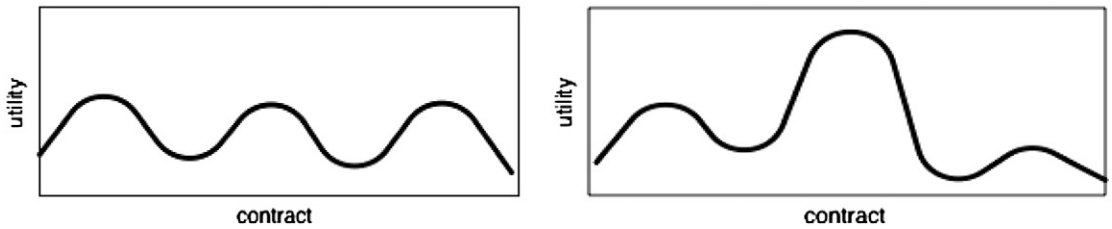


Fig. 12. Agent utility functions with different modality variances.

Scenarios with a low symmetry metric will be more challenging if we want our protocol to find a fair agreement, where fairness can be defined for example as in [17]. We compute symmetry by adding up, for every agent, the distances of possible contracts from the equal utility line. The standard deviation of these sums gives an idea of how much a scenario favors some agents against others.

Our *competitiveness* metric captures the extent to which a scenario is win–win, or win–lose, for the agents involved. We calculate this by finding the average fairness for contracts that have high social welfare (and thus are more apt to include the final agreement). Intuitively, a scenario where most such contracts are highly unfair is more competitive than those where most such contracts are fair. A scenario with a highly convex Pareto front (where the highest social welfare contracts are very unfair) is, by this definition, more competitive than one with a highly concave Pareto front (where the highest social welfare contracts are good for all the agents involved).

Our *correlation* metric measures the correlation between the utility functions for different agents. Scenarios with high correlation values will presumably be easier for a protocol seeking win–win agreements.

Our *locality* metric measures the correlation between the distance of contracts in the solution and utility spaces, which is likely to have a strong impact on the suitability of similarity-based protocols.

Our *basin* metrics, finally, checks for the presence and size of *basins of mutual gain* (regions in the contract space where agents may find dominant directions for increasing mutual gain), which favors hill-climbing protocols.

Our relational metrics involve calculating some function of the utility values for all the contracts in the contract space. This can be computationally impractical, of course, if the contract domain is too large. In this case, we can calculate these values for a representative (e.g. random) sample of the contracts in the domain. It is also straightforward to calculate these metrics from the utility histogram for a scenario, so we can get exact metrics values for scenarios created by our scenario generator.

5.3. Contextual metrics

This set of metrics is used to assess the potential impact of a negotiation scenario's *context*. We have focused, in our work so far, on

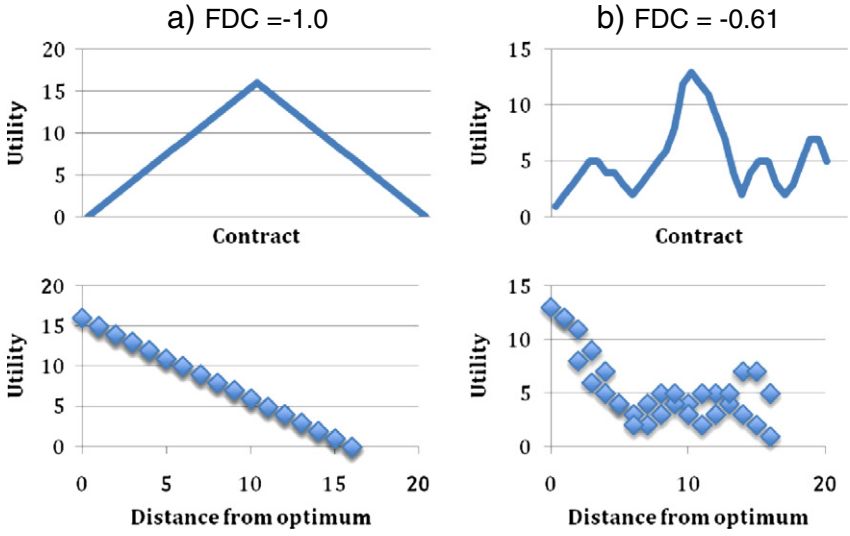


Fig. 13. Agent utility functions with different FDC values.

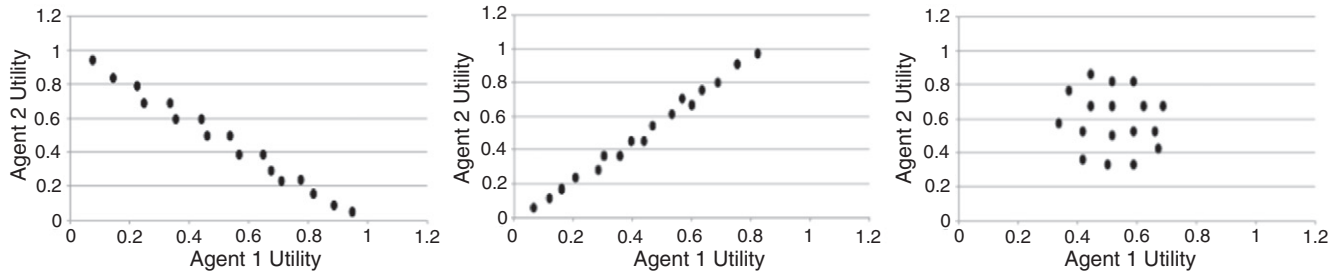


Fig. 14. Typical utility diagrams for zero–sum, win–win, and uncorrelated scenarios with no preferential dependencies.

metrics to assess *predictability*, i.e. how well agents can anticipate what will happen in their current negotiation based on the context of previous, similar, negotiations. The notion of negotiation predictability was first introduced in [22] and later extended in [24]. Predictability is important because it helps negotiating agents make offers that will be appreciated by the opponent, as well as develop strategies that bring the highest utility to the negotiator. For the first we introduce the notion of preference profile predictability and for the latter we introduce bidding strategy predictability.

5.3.1. Preference profile predictability

A preference profile reflects which issues are important to the negotiator, which values per issue are preferred over other values, whether or not there are dependencies between issues, and as a whole provides a (partial) ranking over all possible agreements. Being able to (partly) predict the preference profile of the other negotiators makes it easier to make an offer that the other party can accept, and increases the possibility of reaching a good negotiation outcome more quickly [14,60]. The general question we help to answer is, given a negotiation domain and the negotiators, to what extent is the preference profile of a negotiator predictable? This question can be broken down into several sub-questions [22] that we can define as follows:

- *Role predictability*. Role predictability refers to the existence of typical negotiation roles. For example, in many domains (e.g., markets) there are sellers versus buyers, in job contract negotiations there are applicants and hiring managers. With a role comes information about the preference profile of the agent playing that role. For example, if the negotiation domain is about selling something, then there is typically an issue price, and it is predictable that sellers prefer higher prices, and buyer prefer lower ones.
- *Issue predictability*. An issue is called predictable if we can predict negotiators' preference profile (or some global properties thereof) for an issue. For example, in a negotiation for a supercomputer, the issue of the number of Teraflops and the issue of the number of petabytes are predictable: the buyer would prefer more of both. When negotiating with friends about an upcoming holiday, typical issues are location, duration, start date, way of traveling, and price. Of these issues only price is predictable without knowing anything more about the people involved.
- *Issue weight predictability*. Issue weight predictability refers to predicting the importance of an issue in a negotiation, in absolute or relative terms. Jonker and Robu [29] showed that knowing what the most important issue is to the opponent can produce significant improvements in the outcome of the negotiation.
- *Dependency predictability*. This refers to the question of whether it is reasonable to expect that a negotiating agent will have a dependency between some issues. For example, in a negotiation about a car, the price depends on most other issues, such as the age, the motor, size, and which accessories will be put in the car. Although a buyer would always prefer low price, he can adjust his reservation value for the price with respect to those issues. He may be willing to pay more for a new model while he would be reluctant to pay the same amount for an old model. Thus, predicting such interdependencies may help a negotiating agent generate appropriate offers. The same can be said for predicting preferential dependencies.
- *Domain predictability*. This refers to how many predictable issues there are in a given negotiation, and how much we can predict about the importance of those issues. Domain predictability is important in research into bidding strategies [3,24].

Preference profile predictability, or one of the related notions of predictability can be established in various ways. Domains can be studied

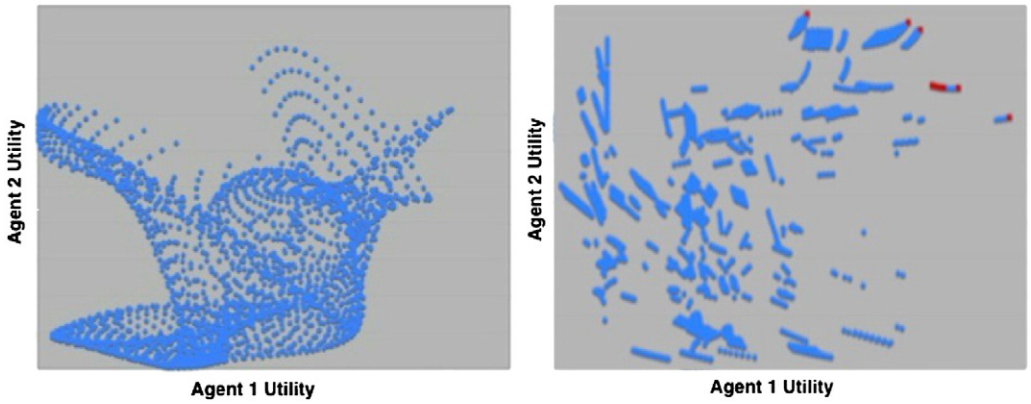


Fig. 15. Utility diagrams for scenarios with strong preferential dependencies.[1]
In both cases, the utility functions were generated[1] as the sum of a bell-shaped weighted hypervolumes.

using conceptual analysis. For a negotiation on computing power, a conceptual analysis will reveal that the number of computing nodes and memory nodes has an effect on the electricity consumption and the cooling required for the nodes (which in turn requires additional electricity). Similarly, conceptual analysis of negotiations on a purchase reveals that there is a certain degree of role predictability (buyer-seller), issue predictability (price is preferred to be higher by the seller and lower by the buyer), and issue weight predictability (price is typically an important issue). Conceptual analysis can also reveal issue predictability for issues that are not predictable in arbitrary domains. For example, someone buying clothes for a wildlife observation tour will prefer colors that fit in with nature. So understanding the underlying concerns of the negotiation helps. In other cases observing your opponent during the negotiation, or just knowing your opponent better might help. For example, the colors the other negotiator picked for e.g., clothing or home decoration, might make the issue of color predictable.

Statistical analysis and data mining are techniques that can be applied on a set of preferences of a multitude of people for a comparable issue or domain to extract a preference profile prediction (or part thereof). For example, although color is in general not predictable, an analysis of the car domain shows that in 2007 (see http://www.cars.com), white was the most popular color in North America, making up 19% of all vehicles. Silver, black, red and gray rounded out the top five.

Various machine learning techniques are used for opponent modeling, for example, to gain more information on the zone of agreement [60] or on the shape of the evaluation functions per issue [23].

The grounds for drawing a conclusion about preference profile predictability can be economic rationality, surveys over a large enough set of people, or an emergent pattern in a set of preference profiles for that issue or for that domain, or in the subsequent bids made by the opponent.

5.3.2. Strategy predictability

Being able to predict the behavior of the other party enables anticipation on that behavior, see e.g., [27]. The results of the Automated Negotiating Agent Competition (ANAC), see [3], show which bidding strategies perform best against each other bidding strategy. This knowledge will probably soon be used to create negotiation strategies that will adapt to the current strategy used by the opponent.

An opponent's strategy is called predictable, if on the basis of earlier encounters or a personality/experience profile, the opponent strategy can be characterized in terms of some global style, such as the negotiation styles of [50,55], or a known conflict-handling style, as the one shown in Fig. 16 [30,45].

6. Protocol testbed

The purpose of the protocol testbed is to allow researchers to assess how well their protocols work in different negotiation scenarios. Our testbed is called GENIUS (General Environment for Negotiation with Intelligent multi-purpose Usage Simulation) [4,38]. A user begins by specifying the experiment they want to run, selecting the scenario (domain and utility functions), protocol, and negotiating agents (Fig. 17).

Once the setup is complete, GENIUS can simulate the negotiation, deriving such statistics as the current utility values for each agent, the product of the agent utilities, and so on (Fig. 18).

The experiment results for different combinations of scenarios and protocols can then be compared with each other. GENIUS has focused, to date, on bilateral alternating offer protocols and linear additive utility functions, and in this form has been used to support the Automated Negotiating Agents Competitions (http://mmi.tudelft.nl/anac).

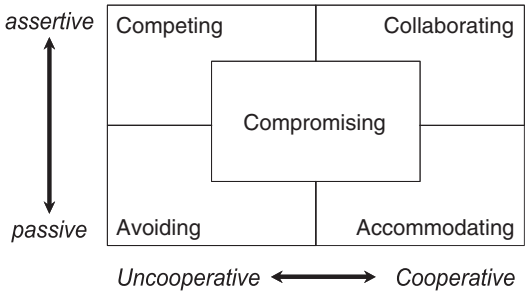


Fig. 16. The Thomas–Kilmann Conflict Mode Instrument.

7. Repositories

The final component of the Negotiation Handbook is the repositories for scenarios as well as for experimental data concerning the performance of different protocols in these scenarios. At present we have two, complimentary, systems that serve this function. GENIUS, as noted above, includes repositories for scenarios and experiment data, but has focused to date on linear utility functions. The second system, a web-based tool called *Negowiki*, has focused on capturing scenarios and experimental results for *nonlinear* cases. Negowiki supports:

- *Downloading negotiation scenario.* Users can search the Negowiki scenario repository for scenarios with desired properties and download them in an XML format that instantiates the weighted hypervolume formalism described in Section 2. The repository currently includes a range of scenarios drawn from the research literature, including [14,25,33,39]. Users can search for scenarios based on their properties (Fig. 19). Every scenario includes a description of the rationale behind it, as well as the values for the scenario metrics as described in Section 4 (Fig. 20). Users can also download function libraries (currently available in Matlab and Common Lisp) that calculate the utility of a given contract for a given scenario, making it straightforward for a research group to use the scenarios in their protocol evaluation experiments.
- *Uploading negotiation scenario for analysis.* Users can upload their scenarios to the Negowiki repository using the scenario formalism described in Section 2. When uploaded, the website computes and displays the metrics values for that scenario. Users can archive these scenarios into the repository so they are available for use by other researchers.
- *Uploading negotiation results data for analysis.* If users perform experiments with protocols from the literature or with their own protocols using any of the Negowiki scenarios, they can upload their negotiation results into Negowiki for analysis. When uploaded, the website computes a range of negotiation outcome metrics (such as individual agent utility, social welfare, fairness, distance from the Pareto front) and shows them to the user. The Negowiki team has devoted substantial effort to ensuring that the site calculates accurate outcome metrics for negotiation experiment results. It is important to note that calculating some outcome measures can be very different in nonlinear scenarios. Traditional solution concepts such as the Nash product, for example, can become very misleading in such contexts [17]. Users can archive their experimental results in the repository to make them available to the community, and can search this repository for insights into which best protocols are best for solving the negotiation problem at hand, as well as for inspiration on how to design better protocols.

For more information about Negowiki, go to http://negowiki.mit.edu/.

8. Contributions and future work

One of the key challenges in automated negotiation research is that it has been difficult to come up with a systematic picture of

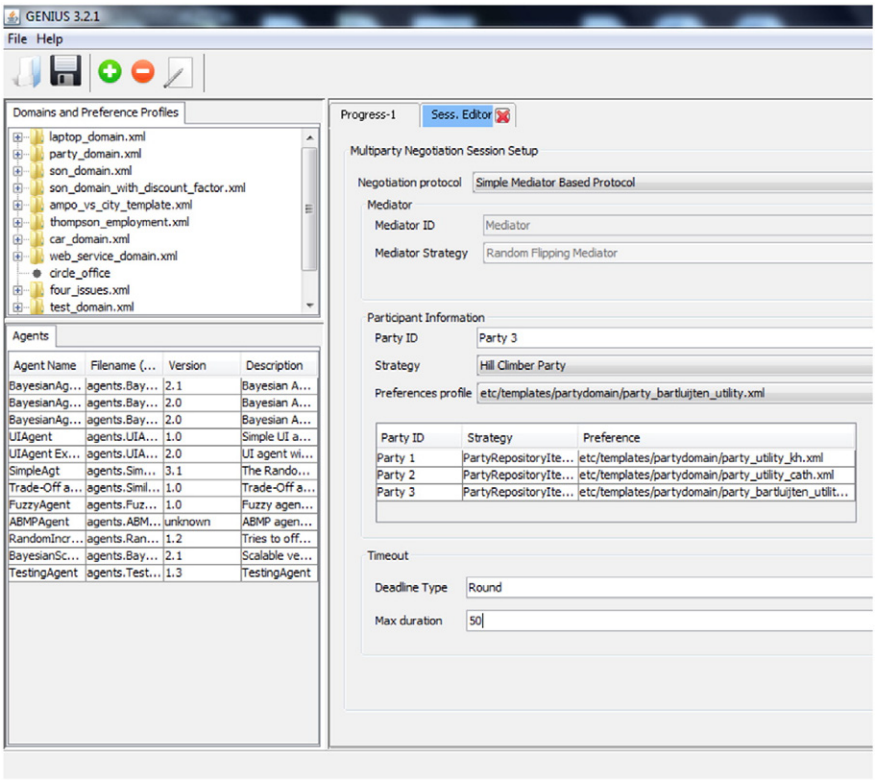


Fig. 17. GENIUS: negotiation simulation setup interface.

which negotiation protocols work best for which problems, due to the diverging scenarios considered by the different research groups. In this paper we have described an infrastructure that aims to fill this gap. This infrastructure provides:

- A tool for *generating negotiation scenarios*, based on the aggregation of hypervolumes to generate utility functions, and on the use of shared hypervolumes and nonlinear regression to generate negotiation scenarios from utility histograms. To our knowledge, despite it's obvious advantages, no authors have previously tried to generate negotiation scenarios from utility diagrams.
- A set of *metrics* to measure high-level scenario parameters, taking into account both the structural properties of the agent utility functions, the relationships between the utility functions of the different agents, and the context in which the negotiation occurs.
- A *testbed* that allows researchers to evaluate how well their protocols work on different scenarios.

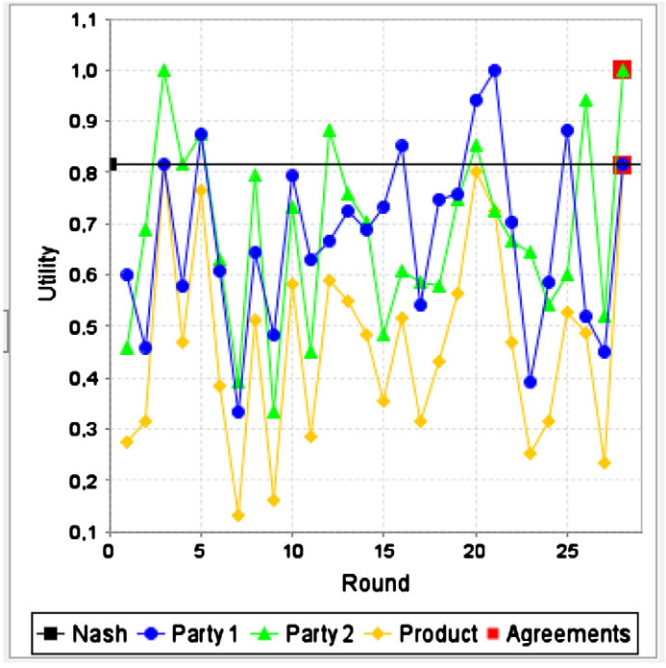


Fig. 18. Overview of a negotiation as it progresses over multiple rounds.

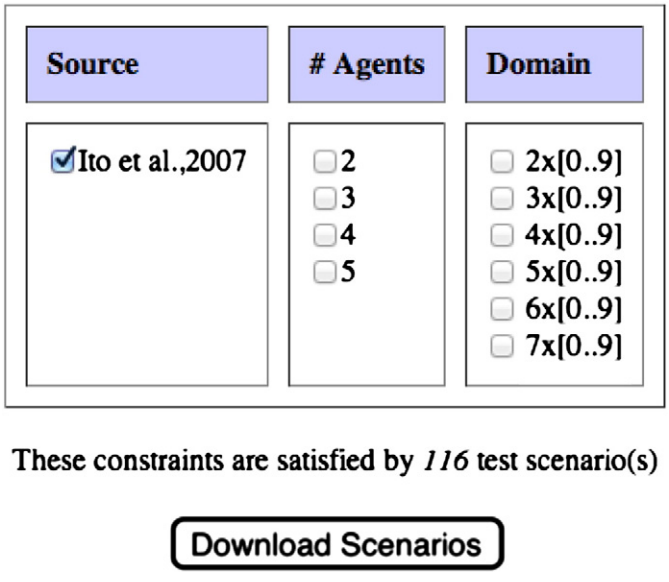


Fig. 19. The Negowiki scenario selection interface.

Scenario: Ito et al 2007

EDIT DOWNLOAD

ID E-1M5UXGG-27
Agents 10
Issues 10
Domain 10
Description

Category:Ito et al.,2007

This scenario involves the kind of utility functions used in the paper by Ito et al *Multi-issue Negotiation Protocol for Agents: Exploring Nonlinear Utility Spaces', published in IJCAI 2007* [1]. *The paper used a weighted_sum of plain utility functions called* constraints. *The utility functions were generated as follows:*

- Number of issues: 2 to 10.
- Domain for issue values: integers from 0 to 9.
- Constraint distribution: 5 unary constraints, 5 binary constraints, 5 trinary contraints, etc. (a unary contraint relates to one issue, an binary constraint relates to two issues, and so on).
- Maximum utility value for a constraint : 100 × (Number of Issues). Constraints that satisfy many issues thus have, on average, larger weights. This seems reasonable for many domains. In meeting scheduling, for example, higher order

Ito et al.,2007

Weighted sum of random plain utility functions

Summary

Weighted sum of random plain utility functions drawn from a uniform distribution

Key features

Number of issues	2,4,6,8,10
Domain of issues	$(x_1, x_2)\|x_i \in [0, 9] \subset \mathbb{R}$

Fig. 20. An example of a Negowiki scenario description.

- *Community repositories* that allow the research community to share scenarios and experiment results and thereby carry on a much more cumulative and systematic exploration of the critical question of which protocols work best for which negotiation problems.

While our approach has been designed to be as general as possible, it does face one inherent limitation. Our scenario representation will not work for domains where it is impractical to capture the entire utility function up front. Imagine, for example, that agents are negotiating over the placement of pieces on a chess board. There is an astronomical number of possible "contracts" (i.e. chess board configurations). It would require an enormous amount of time to calculate the utility values for every possible contract, and an impractically large number of hypervolumes to capture this information. To handle this case, we would need to adopt a scenario representation which represents the *algorithms used to calculate the utility functions* (so contract utilities can be calculated as needed) rather than the utility functions themselves. This is a possible direction for future work.

Our more immediate efforts will involve extending our infrastructure. This will include work on:

- *Scenario generation*: we plan to extend the scenario generator to allow more comprehensive control over other potentially important scenario properties, such as epistasis, fractal statistics in the utility functions [11], and so on.
- *Metrics*: our current set of scenario metrics, while promising, are preliminary and will no doubt have to be refined and extended as we develop a better understanding of which scenario properties most deeply impact the performance of different protocols.
- *Testbed*: currently, the GENIUS testbed environment only supports linear additive utility functions and bilateral protocols. We are extending the GENIUS testbed to support our weighted-hypervolume scenario representation and to enable multilateral negotiations.
- *Repositories*: We will continue to populate the scenario repository with examples from the research literature and from our scenario generator. A critical part of this will be developing a *scenario taxonomy*, i.e. a systematic enumeration of all the important distinct classes of negotiation scenarios, so we can try to assure that the scenario repository has a comprehensive collection of test cases.
- *Experiments*: we will perform comprehensive comparisons of negotiation performance for different protocol and scenario combinations. We are creating a taxonomic representation of known negotiation protocols, based on ideas developed in the Process Handbook project [32], which will allow us to systematically explore the space of protocol variants by a recombination process. Since the space of possible scenarios and protocols is very large, so we will need to develop heuristics for focusing our search on scenario-protocol pairs that are apt to be important for real-world challenges.
- *Data mining*: we will explore the experiment data repositories generated by these experiments to develop keener insights into the determinants and patterns of protocol performance.
- *Community development*: we will work to enable a vibrant community process, building on the infrastructure we have developed, wherein researchers share scenarios, protocols, and experiment results, all in the service of creating a Negotiation Handbook that can substantially impact how well humanity is able to solve its complex and critical negotiation challenges.

Acknowledgments

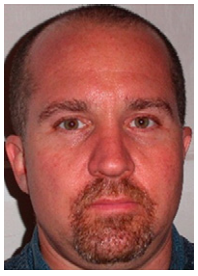
This work has been supported by: the ITEA-2 project 2008005, "Do-it-Yourself Smart Experiences"; the Spanish Ministry of Education and Science grants TIN2008-06739-C04-04 (project T2C2) and JC-2010-035 (mobility grant); the Dutch Technology Foundation STW, Applied Science Division of NWO and the Technology Program of the Ministry of Economic Affairs; the Pocket Negotiator project with grant number VICI-project 08075; the US National Science Foundation; and the New Governance Models for Next Generation Infrastructures project, NGI grant number 04.17.

References

[1] S. Athey, I. Segal, Designing efficient mechanisms for dynamic bilateral trading games, American Economic Review 97 (2) (2007) 131–136.
[2] R. Aydogan, T. Baarslag, K.V. Hindriks, C.M. Jonker, P. Yolum, Heuristic-based approaches for CP-nets in negotiation, Proceedings of The Fourth International Workshop on Agent-based Complex Automated Negotiations (ACAN 2011), 2011, pp. 56–59.
[3] T. Baarslag, K. Hindriks, C.M. Jonker, S. Kraus, R. Lin, New trends in agent-based complex automated negotiations, series of studies in computational intelligence, Proceedings from the First Automated Negotiating Agents Competition (ANAC 2010), 2010.
[4] T. Baarslag, K. Fujita, E.H. Gerding, K.V. Hindriks, T. Ito, N.R. Jennings, et al., Evaluating practical negotiating agents: Results and analysis of the 2011 international competition, Artificial Intelligence 198 (May 2013) 73–103.
[5] C. Boutilier, H.H. Hoos, Bidding languages for combinatorial auctions, Proceedings of the International Joint Conference on Artificial Intelligence, 2001, pp. 1211–1217.
[6] C. Boutilier, F. Bacchus, R.I. Brafman, UCP-networks: a directed graphical representation of conditional utilities, Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, 2001, pp. 56–64.
[7] C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, D. Poole, CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements, Journal of Artificial Intelligence Research 21 (2004) 135–191.
[8] R.N. Bracewell, The Fourier Transform & Its Applications, 3rd ed. McGraw-Hill Science/Engineering/Math, 2000.
[9] R.I. Brafman, C. Domshlak, Preference handling — an introductory tutorial, AI Magazine 30 (1) (2009) 58–86.
[10] P. Cramton, Y. Shoham, R. Steinberg, Introduction to Combinatorial Auctions, MIT Press, 2006.
[11] W. Da Cruz, Fractons and Fractal Statistics, 1999. (Arxiv, preprint hep-th/9905229).
[12] C.C. Eckel, P.J. Grossman, Forecasting risk attitudes: an experimental study using actual and forecast gamble choices, Journal of Economic Behavior & Organization 68 (1) (October 2008) 1–17.
[13] T. Ellingsen, M. Johanneson, J. Lilja, Trust and truth, The Economic Journal 119 (534) (2009) 252–276.
[14] P. Faratin, C. Sierra, N.R. Jennings, Using Similarity Criteria to Make Negotiation Trade-offs, IEEE Computer Society, Washington, DC, USA, 2000, pp. 119–126.
[15] S. Fatima, M. Wooldridge, N.R. Jennings, An analysis of feasible solutions for multi-issue negotiation involving nonlinear utility functions, Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, Budapest, Hungary, 2009, pp. 1041–1048.
[16] F. Ren, M. Zhang, K.M. Sim, Adaptive conceding strategies for automated trading agents in dynamic, open markets, Decision Support Systems 46 (3) (2009) 704–716.
[17] K. Fujita, T. Ito, M. Klein, A secure and fair protocol that addresses weaknesses of the Nash bargaining solution in nonlinear negotiation, Group Decision and Negotiation (2010) 1–19.
[18] M. Grabisch, k-order additive discrete fuzzy measures and their representation, Fuzzy Sets and Systems 92 (2) (1997) 167–189.
[19] G.W. Greenwood, X. Hu, On the use of random walks to estimate correlation in fitness landscapes, Computational Statistics and Data Analysis 28 (1998) 131–137.
[20] R.B. Heckendorn, D. Whitley, Predicting epistasis from mathematical models, Evolutionary Computation 7 (1999) 69–101.
[21] M. Hemaissia, A.E.F. Seghrouchni, C. Labreuche, J. Mattioli, A multilateral multi-issue negotiation protocol, Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, 2007.
[22] K. Hindriks, C. Jonker, D. Tykhonov, Analysis of negotiation dynamics, in: M. Klusch, K. Hindriks, M. Papazoglou, L. Sterling (Eds.), Delft University of Technology, Man–machine Interaction, Mekelweg 4, 2628CD Delft, Springer Berlin/Heidelberg, The Netherlands, 2007, pp. 27–35 (4676).
[23] K. Hindriks, C.M. Jonker, D. Tykhonov, The benefits of opponent models in negotiation, Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, Washington, DC, USA, 2009.
[24] K. Hindriks, C.M. Jonker, D. Tykhonov, Let's dans! An analytic framework of negotiation dynamics and strategies, Web Intelligence and Agent Systems 9 (2011) 319–335.
[25] T. Ito, M. Klein, H. Hattori, A multi-issue negotiation protocol among agents with nonlinear utility functions, Journal of Multiagent and Grid Systems 4 (1) (2008) 67–83.
[26] N.R. Jennings, P. Faratin, A.R. Lomuscio, S. Parsons, C. Sierra, M. Wooldridge, Automated negotiation: prospects, methods and challenges, Group Decision and Negotiation 10 (2) (2001) 199–215.
[27] S.-j Ji, LeungH.-f , An adaptive prediction-regret driven strategy for bilateral bargaining, Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence, Washington, DC, USA, 2010.
[28] T. Jones, S. Forrest, Fitness distance correlation as a measure of problem difficulty for genetic algorithms, Proceedings of the 6th international conference on genetic algorithms, 1995.
[29] C. Jonker, V. Robu, Automated multi-attribute negotiation with efficient use of incomplete preference information, Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Washington, DC, USA New York, New York, 2004.
[30] R.H. Kilmann, K.W. Thomas, Developing a forced-choice measure of conflict-handling behavior: the MODE instrument, Educational and Psychological Measurement 37 (2) (1977) 309–325.
[31] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[32] M. Klein, C. Petti, A handbook-based methodology for redesigning business processes, Knowledge and Process Management 13 (2) (2006) 108–119.
[33] M. Klein, P. Faratin, H. Sayama, Y. Bar-Yam, Negotiating complex contracts, Group Decision and Negotiation 12 (2) (2003) 111–125.
[34] S. Kraus, Automated negotiation and decision making in multiagent environments, in: M. Luck, V. Marik, S. Olga, R. Trappl (Eds.), Bar-Ilan University Dept. of Mathematics and Computer Science Ramat-Gan 52900, Springer Berlin/Heidelberg, Israel, 2006, pp. 150–172, (2086).
[35] G. Lai, C. Li, K. Sycara, J. Giampapa, Literature review on multiattribute negotiations, 2004. (CMU-RI-TR-04-66).
[36] R.Y.K. Lau, Y. Li, D. Song, R.C.W. Kwok, Knowledge discovery for adaptive negotiation agents in e-marketplaces, Decision Support Systems 45 (2) (2008) 310–323.
[37] E. Ley, Statistical inference as a bargaining game, Economics Letters 93 (1) (2006) 142–149.
[38] R. Lin, S. Kraus, T. Baarslag, D. Tykhonov, K.V. Hindriks, C.M. Jonker, Genius: an integrated environment for supporting the design of generic automated negotiators, Computational Intelligence (2012), http://dx.doi.org/10.1111/j.1467-8640.2012.00463.x.
[39] M.A. Lopez-Carmona, I. Marsa-Maestre, E.d. l Hoz, J.R. Velasco, A region-based multi-issue negotiation protocol for non-monotonic utility spaces, Computational Intelligence 27 (2) (2011) 166–217.
[40] G. Marsaglia, Choosing a point from the surface of a sphere, Annals of Mathematical Statistics 43 (2) (1972) 645–646.
[41] I. Marsa-Maestre, M.A. Lopez-Carmona, M. Klein, T. Ito, F. Katsuhide, Addressing utility space complexity in negotiations involving highly uncorrelated, constraint-based utility spaces, Computational Intelligence (2012), http://dx.doi.org/10.1111/j.1467-8640.2012.00461.x.
[42] R. Myerson, M. Satterthwaite, Efficient mechanism for bilateral trading, Journal of Economic Theory 28 (1983) 265–281.
[43] S. Parsons, J.A. Rodriguez-Aguilar, M. Klein, Auctions and bidding: a guide for computer scientists, Communications of the ACM 43 (2) (2011).
[44] C. Patton, P.V. Balakrishnan, The impact of expectation of future negotiation interaction on bargaining processes and outcomes, Journal of Business Research 63 (8) (2010) 809–816.
[45] D.G. Pruitt, Trends in the scientific study of negotiation and mediation, Negotiation Journal 2 (1986) 237–244.
[46] H. Raiffa, The Art and Science of Negotiation, Harvard University Press, Cambridge, USA, 1982.
[47] V. Robu, D.J.A. Somefun, J.A. La Poutré, Modeling complex multi-issue negotiations using utility graphs, Proceedings from AAMAS'05, Utrecht, The Netherlands, 2005.
[48] J.S. Rosenschein, G. Zlotkin, Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, MIT Press, Cambridge, Mass., 1994
[49] T.W. Sandholm, Distributed rational decision making, in: G. Weiss (Ed.), Multi-Agent Systems, 1998.
[50] G.R. Shell, Bargaining for Advantage: Negotiation Strategies for Reasonable People, Penguin Books, 2006.
[51] K.M. Sim, B. Shi, Concurrent negotiation and coordination for grid resource co-allocation, IEEE Transactions on Systems, Man and Cybernetics Part B 40 (2) (2009).
[52] R.G. Smith, The contract net protocol: high-level communication and control in a distributed problem solver, IEEE Transactions on Computers C-29 (12) (August 1980) 1104–1113.
[53] P.F. Stadler, W. Gruner, Anisotropy in fitness landscapes, Journal of Theoretical Biology 165 (3) (1993) 373–388.
[54] K.P. Sycara, Machine learning for intelligent support of conflict resolution, Decision Support Systems 10 (2) (1993) 121–136.
[55] L.L. Thompson, The Mind and Heart of the Negotiator, Pearson, 2011.
[56] M. Tomassini, L. Vanneschi, P. Collard, M. Clergue, A study of fitness distance correlation as a difficulty measure in genetic programming, Evolutionary Computation 13 (2) (2005) 213–239.
[57] V.K. Vassilev, T.C. Fogarty, J.F. Miller, Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application, Advances in Evolutionary Computing: Theory and Applications, Springer-Verlag New York, Inc., New York, NY, USA, 2003, pp. 3–44.
[58] M. Ventresca, B. Ombuki-Berman, Epistasis in multi-objective evolutionary recurrent neuro-controllers, Proceedings of the 2007 IEEE Symposium on Artificial Life (ALIFE '07), 2007.
[59] M. Wu, M. Weerdt, H. Poutré, Efficient methods for multi-agent multi-issue negotiation: allocating resources, Proceedings of the 12th International Conference on

Principles of Practice in Multi-Agent Systems (PRIMA '09), Berlin, Heidelberg Nagoya, Japan, 2009.
[60] D. Zeng, K. Sycara, Benefits of learning in negotiation, Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence (AAAI'97/IAAI'97), Providence, Rhode Island, 1997.

Dr. Ivan Marsa-Maestre is a PhD lecturer at the Computer Engineering Department of the University of Alcala in Spain. He received his Telecommunication Engineering degree from the University of Alcala in 2003 and the Ph.D. from the University of Alcala in 2009. His research interests focus on the use of negotiation and nonlinear optimization techniques for distributed coordination of complex systems, such as computer networks, supply chains or vehicle management systems. He has taken part in many public and private research projects in these matters, has a number of publications in high impact international conferences and journals, and serves as program chair and reviewer for some of them. From his research have emerged collaborative research lines with international research groups like the Center for Green Computing, at the Nagoya Institute of Technology (Japan), the Technical University of Delft (TUDelft) or the Center for Collective Intelligence, at the Massachusetts Institute of Technology (USA), where he worked as visiting researcher during the year 2010–2011.



Dr. Mark Klein is a Principal Research Scientist at the MIT Center for Collective Intelligence, as well as an Affiliate at the MIT Computer Science and AI Lab (CSAIL) and the New England Complex Systems Institute (NECSI). His research draws from such fields as computer science, economics, operations research, and complexity science in order to develop and evaluate computer technologies that enable greater 'collective intelligence' in large groups faced with complex decisions, especially those in the sustainability realm. His current projects are looking at: large-scale on-line deliberation; negotiation protocols for problems with many interdependent issues; knowledge management that integrates virtual reality and social media; and managing 'emergent' dysfunctions in large-scale networked systems. He has also made contributions in the areas of computer-supported conflict management for collaborative design, design rationale capture, business process re-design, exception handling in workflow and multi-agent systems, and service discovery.

Prof. Catholijn Jonker is full professor of Man–machine Interaction at the Faculty of Electrical Engineering, Mathematics and Computer Science of the Delft University of Technology. She studied computer science, and did her PhD studies at Utrecht University. After a post-doc position in Bern, Switzerland, she became assistant (later associate) professor at the Department of Artificial Intelligence of the Vrije Universiteit Amsterdam. From September 2004 until September 2006 she was a full professor of Artificial Intelligence/Cognitive Science at the Nijmegen Institute of Cognition and Information of the Radboud University Nijmegen. She chaired De Jonge Akademie (Young Academy) of the KNAW (The Royal Netherlands Society of Arts and Sciences) in 2005 and 2006, and she was a member of the same organization from 2005 to 2010. She is a board member of the National Network Female Professors (LNVH) in The Netherlands. Since 2013 she is a member of the Academia Europae. Her publications address cognitive processes and concepts such as trust, negotiation, teamwork and the dynamics of individual agents and organizations. In Delft she works with an interdisciplinary team to create synergy between humans and technology by understanding, shaping and using fundamentals of intelligence and interaction. End 2007 her NWO-STW VICI project "Pocket Negotiator" has been awarded. In this project she develops intelligent decision support systems for negotiation.

Dr. Reyhan Aydoğan is a postdoctoral researcher in the Interactive Intelligence group at Delft University of Technology, the Netherlands. She received her MSc and PhD degrees from the Department of Computer Engineering at Boğaziçi University and her BS degree from the Department of Computer Engineering at Dokuz Eylül University. During her MSc and PhD studies, Dr. Aydoğan was funded by the Scientific & Technological Research Council of Turkey (TUBITAK) under the national MSc and PhD student scholarship program. Her research interests are agreement technologies, particularly negotiation, decision support systems, multiagent systems, machine learning, service oriented architectures and semantic Web. She serves as a program committee member in reputable conferences such as AAMAS and IJCAI, and a reviewer in highly referred journals such as JAAMAS and JAIR.