

M-PREF10
5th Multidisciplinary Workshop on Advances in
Preference Handling

Jérôme Lang
LAMSADE, France

Patrice Perny
LIP6 - Paris 6 University, France

Preface

Preferences are a central concept of decision making. As preferences are fundamental for the analysis of human choice behavior, they are becoming of increasing importance for computational fields such as artificial intelligence, databases, and human-computer interaction. Preference models are needed in decision-support systems such as web-based recommender systems, in automated problem solvers such as configurators, and in autonomous systems such as Mars rovers. Nearly all areas of artificial intelligence deal with choice situations and can thus benefit from computational methods for handling preferences. Moreover, social choice methods are also of key importance in computational domains such as multi-agent systems.

This broadened scope of preferences leads to new types of preference models, new problems for applying preference structures, and new kinds of benefits. Preferences are studied in many areas of artificial intelligence such as knowledge representation, multi-agent systems, game theory, social choice, constraint satisfaction, decision making, decision-theoretic planning, and beyond. Preferences are inherently a multi-disciplinary topic, of interest to economists, computer scientists, operations researchers, mathematicians and more.

This workshop promotes this broadened scope of preference handling and continues a series of events on preference handling at AAAI-02, Dagstuhl in 2004, IJCAI-05, ECAI-06, VLDB-07, AAAI-08, and ADT-09. Since 2008, this series of workshops is organized by the multidisciplinary working group on Advances in Preference Handling, which is affiliated to the Association of European Operational Research Societies EURO.

Jérôme Lang, LAMSADE, France

Patrice Perny, LIP6 - Paris 6 University, France

August 2010

Organization

Workshop Chairs

Ulrich Junker, France
Jérôme Lang, LAMSADE, France
Patrice Perny, LIP6 - Paris 6 University, France

Program Committee

Carlos Henggeler Antunes, University of Coimbra, Portugal
Wolf-Tilo Balke, University of Braunschweig, Germany
Sylvain Bouveret, Onera, France
Ronen Brafman, Ben-Gurion University, Israel
James Delgrande, Simon Fraser University Vancouver, Canada
Carmel Domshlak, Technion - Israel Institute of Technology, Israel
Jon Doyle, North Carolina State University, USA
Matthias Ehrgott, University of Auckland, New Zealand
Edith Elkind, Nanyang Technological University, Singapore
Judy Goldsmith, University of Kentucky, USA
Salvatore Greco, University of Catania, Italy
Ulrich Junker, France
Werner Kießling, University of Augsburg, Germany
Jérôme Lang, LAMSADE, France
Denis Mindolin, University at Buffalo, USA
Barry O'Sullivan, University College Cork, Ireland
David Parkes, Harvard University, USA
Patrice Perny, LIP6 - Paris 6 University, France
Francesca Rossi, University of Padova, Italy
Roman Slowinski, Poznan University of Technology, Poland
Olivier Spanjaard, LIP6 - Paris 6 University, France
Alexis Tsoukiàs, LAMSADE, France
Joel Uckelman, University of Amsterdam, The Netherlands
Paolo Viappiani, University of Toronto, Canada
Toby Walsh, UNSW, Australia
Neil Yorke-Smith, American University of Beirut, Lebanon and SRI International, USA

Table of Contents

Preference Handling in Nested Temporal Networks with Alternatives <i>Roman Barták</i>	1
Using Soft Constraints for Deduction and Abduction with Soft Constraints <i>Stefano Bistarelli, Fabio Martinelli and Francesco Santini</i>	7
Finding the next solution in constraint- and preference-based knowledge representation formalisms <i>Ronen Brafman, Francesca Rossi, Domenico Salvagnin, Kristen Brent Venable and Toby Walsh</i>	13
Factors Influencing User Motivation for Giving Online Preference Feedback <i>Joost Broekens, Alina Pommeranz, Pascal Wiggers and Catholijn Jonker</i> ..	19
Synthesizing Preference Information of Multiple Decision Makers in Terms of Collective Decision Rules <i>Salem Chakhar and Inès Saad</i>	25
Multi-agent soft constraint aggregation: a sequential approach <i>Giorgio Dalla Pozza, Francesca Rossi and Kristen Brent Venable</i>	31
Local search for stable marriage problems with ties and incomplete lists <i>Mirco Gelain, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable and Toby Walsh</i>	37
Dynamic House Allocation <i>Sujit Gujar, James Zou and David Parkes</i>	43
Computation of the robust preference relation combining a Choquet integral and utility functions <i>Christophe Labreuche, Pedro Miranda and Fabien Lehuédé</i>	49
On Finding Compromise Solutions in Multiobjective Markov Decision Processes <i>Patrice Perny and Paul Weng</i>	55
Extracting Features from Ratings: The Role of Factor Models <i>Joachim Selke and Wolf-Tilo Balke</i>	61
Preference Dominance Approaches for Conversational Recommender Systems <i>Walid Trabelsi, Nic Wilson, Derek Bridge and Francesco Ricci</i>	67
Alice and Bob Will Fight: The Problem of Electing a Committee in the Presence of Candidate Interdependence <i>Joel Uckelman</i>	73
Preferences, utilities, and quantitative judgments <i>Richard Wallace and Gordon Rios</i>	79

A Database Approach to Dynamic Group Formation	
<i>Florian Wenzel and Werner Kießling</i>	85

Preference Handling in Nested Temporal Networks with Alternatives

Roman Barták¹

Abstract. Temporal Networks with Alternatives (TNA) and their nested form were proposed to model processes in manufacturing environments but they can be applied to other scheduling problems dealing with alternative ways to achieve goals. In this paper we show how to extend the Nested TNA with preferences on alternatives. We propose an equivalence tree to model logically equivalent nodes in a Nested TNA and we show how preference reasoning can be realized using this novel representation.

1 INTRODUCTION

The scheduling task is usually formulated as a problem of allocating known activities to scarce resources, such as machines, over time. The scheduling task is typically preceded by a planning task, where the problem is to decide which activities are necessary to achieve a given goal. Recently planning and scheduling problems were integrated to achieve greater flexibility and better quality of schedules. In particular, the selection of activities is being done during scheduling rather than before. Beck and Fox [4] introduced the idea of alternative activities where a Boolean validity variable indicates whether the activity is included in the schedule or not. This idea has been brought to practice by ILOG in their MaScLib [8] and the same concept was independently formalized in the Extended Resource Constrained Project Scheduling Problem (x-RCPSP) [7]. Both MaScLib and x-RCPSP allow explicit binary constraints between the Boolean validity variables. Temporal Networks with Alternatives (TNA) [1] formalized the original concept by Beck and Fox of modeling parallel and alternative processes. Basically, the TNA is a directed acyclic graph with nodes corresponding to the activities and arcs corresponding to the temporal relations, where the input and output branchings in nodes are annotated either as parallel or alternative branching (Figure 1). Hence the logical constraints between the Boolean validity variables are implicit there. For example, the input parallel branching for node x means that all direct predecessors of x must be processed, while the input alternative branching means that exactly one direct predecessor of x must be processed to allow processing of x . If only precedence relations instead of the temporal ones are used, then we are talking about a *parallel/alternative graph* (P/A graph in short). We showed that the selection of nodes to satisfy all branching constraints (a so called P/A graph assignment problem) is NP-hard for general TNAs [1] and we proposed a restricted form of a TNA called a Nested TNA where this assignment problem is tractable, if only precedence constraints are used. Nested TNAs are very close to real-life manufacturing processes as Nested TNAs are obtained by parallel and alternative decompositions of processes. The structure of Nested TNAs is also identical to Temporal Planning Networks

(TPN) proposed in [6] for description of temporal relations between the operations obtained from the Hierarchical Task Networks in planning.

In this paper we focus on the optimization version of the P/A graph assignment problem that corresponds to selection of the cheapest process among the alternative processes. Briefly speaking, the nodes in a Nested TNA are annotated by the cost expressing the preference and the task is to select a subset of nodes satisfying the branching constraints such that the sum of their costs is minimized. This is a practically important problem that is novel to scheduling. Traditional scheduling criteria functions, such as makespan, earliness, and tardiness, are related to time, while we suggest including cost of alternatives in the objective. We will introduce a so called *equivalence tree* that can be used to solve the P/A graph assignment problem and then we will show that this data structure can also be used to solve the optimization version of the P/A graph assignment problem. In general, the equivalence tree can be used as a data structure inside the global constraint encapsulating the logical relations in a Nested P/A graph (a Nested TNA) and also describing the preference reasoning in this graph (estimating the cost of the schedule or a cost-based domain filtering).

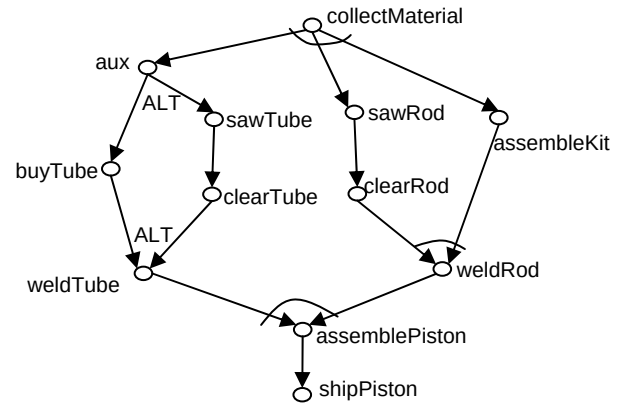


Figure 1. A real-life manufacturing process with alternatives.

The paper is organized as follows. We will first recall the formal definitions of Nested Temporal Networks and the P/A graph assignment problem. Then we will introduce the equivalence tree as a data structure capturing information about logical equivalence of nodes in Nested TNAs. After that we will formulate the optimization version of the P/A graph assignment problem and discuss its practical importance. Finally, we will show how the equivalence tree can be exploited when solving the P/A graph optimization problem and how the cost information can be propagated through the equivalence tree.

¹ Charles University, Czech Republic, e-mail: bartak@ktiml.mff.cuni.cz

2 PRELIMINARIES

P/A graphs were proposed in [1] to formally describe processes with parallel and alternative branches. Let us first recapitulate the formal definitions of a P/A graph and a P/A graph assignment problem. Let G be a directed acyclic graph. A sub-graph of G is called a *fan-out sub-graph* if it consists of nodes $x, y_1, \dots, y_k$ (for some k) such that each $(x, y_i), 1 \leq i \leq k$, is an arc in G . If $y_1, \dots, y_k$ are all and the only successors of x in G (there is no z such that (x, z) is an arc in G and $\forall i = 1, \dots, k: z \neq y_i$) then we call the fan-out sub-graph complete. Similarly, a sub-graph of G is called a *fan-in sub-graph* if it consists of nodes $x, y_1, \dots, y_k$ (for some k) such that each $(y_i, x), 1 \leq i \leq k$, is an arc in G . A complete fan-in sub-graph is defined similarly as above. In both cases x is called a *principal node* and all $y_1, \dots, y_k$ are called *branching nodes*.

Definition 1: A directed acyclic graph G together with its pairwise edge-disjoint decomposition into complete fan-out and fan-in sub-graphs, where each sub-graph in the decomposition is marked either as a *parallel* sub-graph or an *alternative* sub-graph, is called a *P/A graph*.

Fan-in and fan-out sub-graphs impose logical relations between the nodes – we call them *branching constraints*. In particular, we are interested in finding whether it is possible to select a subset of nodes in such a way that they form a feasible graph according to the branching constraints. Formally, the selection of nodes can be described by an *assignment* of 0/1 values to nodes of a given P/A graph, where value 1 means that the node is selected (valid) and value 0 means that the node is not selected (invalid). The assignment is called *feasible* if

- in every parallel sub-graph all nodes are assigned the same value (both the principal node and all branching nodes are either all 0 or all 1),
- in every alternative sub-graph either all nodes (both the principal node and all branching nodes) are 0 or the principal node and exactly one branching node are 1 while all other branching nodes are 0.

Notice that the feasible assignment naturally describes one of the alternative processes in the P/A graph. For example, *weldRod* is present if and only if both *clearRod* and *assembleKit* are present (Figure 1). Similarly, *weldTube* is present if exactly one of nodes *buyTube* or *clearTube* is present (but not both).

It can be easily noticed that given an arbitrary P/A graph the assignment of value 0 to all nodes is always feasible. On the other hand, if some of the nodes are required to take value 1, then the existence of a feasible assignment is by no means obvious. Let us now formulate this decision problem formally.

Definition 2: Given a P/A graph G and a subset of nodes in G which are assigned to 1, *P/A graph assignment problem* is “Is there a feasible assignment of 0/1 values to all nodes of G which extends the prescribed partial assignment?”

Intuition motivated by real-life examples says that it should not be complicated to select the nodes to form a valid process according to the branching constraints described above. The following proposition from [1] says the opposite.

Proposition 1: The P/A graph assignment problem is NP-complete.

The P/A graph has a very general structure, but the structure of real-life processes is frequently more restricted as it is obtained by decomposing a single meta-process into serial, parallel, or alternative sub-processed until primitive activities are reached. Based on this observation, we proposed a nested P/A graph in the following way [2]:

Definition 3: A directed graph $G = (\{s, e\}, \{(s, e)\})$ is a *(base) nested graph*. Let $G = (V, E)$ be a graph, $(x, y) \in E$ be an arc in G , and $z_1, \dots, z_k (k > 0)$ be nodes such that no z_i is in V . If G is a nested graph (and $I = \{1, \dots, k\}$) then graph $G' = (V \cup \{z_i \mid i \in I\}, E \cup \{(x, z_i), (z_i, y) \mid i \in I\} - \{(x, y)\})$ is also a *nested graph*.

According to Definition 3, any nested graph can be obtained from the base graph with a single arc by repeated substitution of any arc (x, y) by a special sub-graph with k nodes (see Figure 2). Notice that a single decomposition rule covers both the serial decomposition ($k = 1$) and the parallel/alternative process decomposition ($k > 1$).

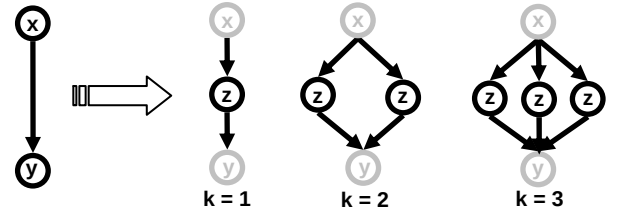


Figure 2. Arc decompositions in nested graphs.

The directed nested graph defines topology of the nested P/A graph but we also need to annotate all fan-in and fan-out sub-graphs as either alternative or parallel sub-graphs. The idea is to annotate each node by input and output label which defines the type of branching. Recall that a fan-out sub-graph with principal node x and branching nodes z_i is a sub-graph consisting of nodes $x, z_1, \dots, z_k$ (for some k) such that each $(x, z_i), 1 \leq i \leq k$, is an arc in G . Fan-in sub-graph is defined similarly.

Definition 4: *Labeled nested graph* is a nested graph where each node has (possibly empty) input and output labels defined in the following way. Nodes s and e in the base nested graph and nodes z_i introduced during decomposition have empty initial labels. Let k be the parameter of decomposition when decomposing arc (x, y) . If $k > 1$ then the output label of x and the input label of y are unified and set either to PAR or to ALT (if one of the labels is non-empty then this label is used for both nodes).

Figure 3 shows how the labeled nested graph is constructed for the example from Figure 1. Notice how the labels are introduced (a semicircle for PAR label and A for ALT label) or unified in case that one of the labels already exists (see the third step). When a label is introduced for a node, it never changes in the decomposition process.

Definition 5: A *nested P/A graph* is obtained from a labeled nested graph by removing the labels and defining fan-in and fan-out sub-graphs in the following way. If the input label of node x is non-empty then all arcs (y, x) form a fan-in sub-graph which is parallel for label PAR or alternative for label ALT. Similarly, nodes with a non-empty output label define fan-out sub-graphs. Each arc (x, y) such that both output label of x and input label of y are empty forms a parallel fan-in sub-graph.

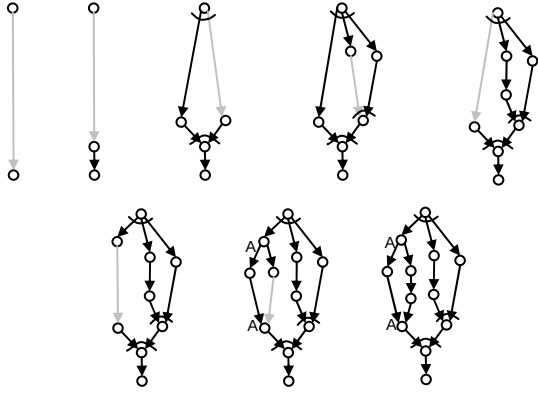


Figure 3. Construction of a labelled nested graph.

Note, that requesting a single arc to form a parallel fan-in sub-graph is a bit artificial. We use this requirement to formally ensure that each arc is a part of some sub-graph. In [2] we showed that a nested P/A graph is a P/A graph and we also justified that indeed the restricted nested structure makes the assignment problem easy.

Proposition 2: The assignment problem for a nested P/A graph is tractable (can be solved in a polynomial time).

Note finally that Temporal Network with Alternatives is obtained from the P/A graph by annotating each arc by a simple temporal constraint (defines minimal and maximal distances between the nodes allocated to time) and similarly for Nested TNA. In this paper we assume that simple temporal constraints are only in the form of precedence relations which are always satisfied as the graphs are acyclic. However, as showed in [2] if unrestricted simple temporal constraints are assumed then the assignment problem becomes NP-hard even for the Nested TNA.

3 EQUIVALENCE TREE

For solving the P/A graph assignment problem, it is useful to know which nodes are equivalent in the sense that either all appear in a feasible assignment or none of them appears there [2]. One of the problems of Nested TNAs is that information about equivalent nodes is not explicitly represented in the network. Let us first formally introduce the notion of equivalent nodes:

Definition 6: Let G be a P/A graph and u and v be arbitrary two nodes of G . Then u and v are called *equivalent* if and only if there is no feasible assignment of nodes in G in which u and v are assigned different values.

As showed in [2] equivalent nodes can be detected during the process of constructing the nested network and hence logical reasoning is tractable for nested P/A graphs. We propose now a novel data structure called an *equivalence tree* that explicitly keeps information about the equivalent nodes. This data structure can be used both for logical reasoning (what happens if some node becomes valid or invalid) and for preference reasoning as we shall present later.

Notice that the nodes participating in the parallel branching are all equivalent in the sense of being all or none present in the solution graph. A non-equivalence relation between the nodes is introduced only via the alternative branching. The idea of the equivalence tree is to keep all equivalent nodes from a Nested TNA

together in a so called E-node. The alternative branching in a Nested TNA is represented using a so called A-node in the equivalence tree that introduces new subsets of equivalent nodes (Figure 4).

We build the equivalence tree along the process of building the Nested TNA in the following way. The base nested graph $G = (\{s, e\}, \{(s, e)\})$ is represented using a root E-node annotated by the set $\{s, e\}$. Now, if we decompose arc (s, e) and the decomposition is either parallel or a single node is introduced then we add the new nodes to the annotation of the root E-node. If the decomposition is alternative with new nodes $z_1, \dots, z_k$ ($k > 1$) then we introduce a new A-node connected to the root and k E-nodes connected to this A-node such that i -th E-node is annotated by the set $\{z_i\}$. Hence there are as many A-nodes in the equivalence tree as the number of decompositions steps with alternative branching. In general, each time arc (x, y) is decomposed we find E-nodes in the current equivalence tree containing x and y in their annotation. It may happen that both x and y are in the same E-node and then we continue like above (in case of alternative branching the new A-node is connected to this E-node, otherwise the new nodes are added to the annotation of that E-node). If x and y are in different E-nodes than we do the above extension of the equivalence tree with the E-node that is farther from the root (we shall show later that both E-nodes must be in the same branch of the equivalence tree). Figure 4 shows the equivalence tree for the Nested TNA from Figure 1.

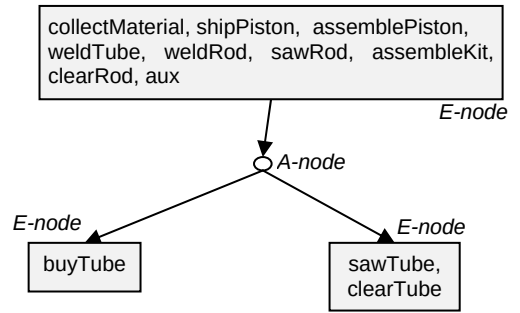


Figure 4. Example of an equivalence tree (the boxes represents the E-nodes, the circle represents an A-node).

Before going into formal definitions, it should be noted that the equivalence tree is unique for given Nested TNA independently of the order of decompositions steps, but a single equivalence tree may correspond to several Nested TNAs.

Definition 7: Let $G = (V, E)$ be a nested P/A graph. Then the *equivalence tree* for graph G is a tree with two types of nodes: E-nodes and A-nodes satisfying the following properties. The root of the tree is an E-node, the leaves of the tree are E-nodes, the parent of A-node is an E-node, and the parent of an E-node is an A-node (hence the E-nodes and the A-nodes alternate in any branch of the equivalence tree). Each E-node is annotated by a non-empty subset of nodes from V such that all these nodes are equivalent with respect to G . Each node from V is present in exactly one E-node. Finally, let X be an E-node, Y be its child node (must be an A-node), and $Z_1, \dots, Z_k$ be all children of Y (must be E-nodes) and let x be from the annotation of X and each z_i be from the annotation of Z_i ($1 \leq i \leq k$). Then in any feasible solution to the assignment problem for graph G , either all $x, z_1, \dots, z_k$ are invalid or x is valid and exactly one of nodes $z_1, \dots, z_k$ is valid.

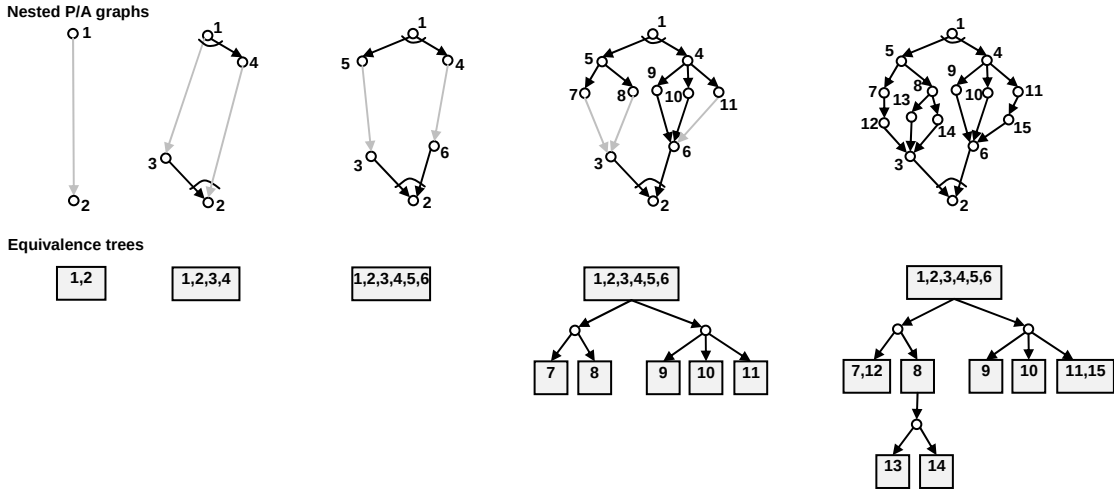


Figure 5. Construction of equivalence trees for the nested P/A graphs.

As already mentioned the motivation behind the equivalence tree is keeping all equivalent nodes together and representing alternative branching relations. We sketched the process of obtaining an equivalence tree for the nested P/A graph at the beginning of this section. The following **Algorithm 1** formalizes the process.

1. The base nested graph $G = (\{s, e\}, \{(s, e)\})$ is represented using an equivalence tree T with a single E-node annotated by the set $\{s, e\}$.
2. Let $G = (V, E)$ be a nested P/A graph, $(x, y) \in E$ be its arc that will be decomposed to new nodes $z_1, \dots, z_k$ ($k > 0$), and G' be the P/A graph after the decomposition (see Definition 3). Let $T = (N, C)$ be an equivalence tree for G and X denotes the E-node from N that is the farthest node from the root among the nodes whose annotation contain either x or y .
 - a. If the decomposition is parallel or $k = 1$ then $T' = T$, where the annotation of X is extended by nodes $z_1, \dots, z_k$, z_k is an equivalence tree for G' .
 - b. If the decomposition is alternative and $k > 1$ then let Y be a new A-node (not in N) and $Z_1, \dots, Z_k$ be new E-nodes (not in N) such that Z_i is annotated by $\{z_i\}$. The graph $T' = (N \cup \{Y\} \cup \{Z_i \mid i \in I\}, C \cup \{(X, Y)\} \cup \{(Y, Z_i) \mid i \in I\})$, where $I = \{1, \dots, k\}$, is an equivalence tree for G' .

Figure 5 demonstrates the above process of building an equivalence tree.

Lemma 1: Let $G = (V, E)$ be a nested P/A graph, $(x, y) \in E$ be an arc in G , $T = (N, C)$ be an equivalence tree for G constructed using the Algorithm 1, and X and Y be E-nodes in T containing x and y respectively in their annotations. Then either $X = Y$ or the path between the nodes X and Y is a part of a branch in T .

Proof: The equivalence tree for the base nested graph trivially satisfies the lemma. All arcs added in step 2 of the algorithm also satisfy the lemma. ■

Proposition 3: The graph T obtained by Algorithm 1 for a nested P/A graph $G = (V, E)$ is an equivalence tree for G .

Proof: First, it should be clear that the graph T obtained by Algorithm 1 is a tree with alternating E-nodes and A-nodes on each branch, the root of T is an E-node and the leaves of T are E-nodes.

Also each node from V appears in annotation of exactly one E-node from the equivalence tree T .

Let us prove now that the nodes in the annotation of a given E-node are equivalent with respect to G . The nodes s and e from the base graph are equivalent. New nodes can be added to the existing annotation of the E-node only by step (2a) of Algorithm 1. Either these nodes are a part of a parallel sub-graph (a decomposition) and all such nodes are equivalent (the definition of a feasible assignment for parallel sub-graphs) or the new node is a part of the decomposition of size 1. In this second case, assume that arc (x, y) was decomposed and a new node z was added. Either x and y are equivalent and hence part of the annotation of a single E-node; z is added to this E-node because it is equivalent with x and y , or x is a principal node and y is a branching node in some alternative sub-graph (a symmetrical situation is similar) and then z is equivalent with y . Then y was added later to graph G and hence the E-node Y with y is farther from the root of T than the node with x (due to step 2b of Algorithm 1) and z is added to the annotation of Y .

It remains to prove the last condition of Definition 7 which is a direct consequence of step (2b) of Algorithm 1, definition of feasible assignment for alternative sub-graphs, and having only equivalent nodes in the annotation of each E-node. ■

The equivalence trees represent a compact way for describing the branching constraints in nested P/A graphs. Instead of specifying the branching constraints for the nodes in a P/A graph, we can specify the branching constraints for the E-nodes in the corresponding equivalence tree. In particular, let Val_X be a 0/1 variable for the E-node X . Then we define the branching constraints in the following way. Let X be the E-node, Y be its child node (must be an A-node), and $Z_1, \dots, Z_k$ are all children of Y (must be E-nodes) then the branching constraint is $\text{Val}_X = \sum_{i=1, \dots, k} \text{Val}_{Z_i}$. The soundness of this model follows directly from the Definition 7. Moreover, this model is clearly Berge acyclic where arc consistency guarantees global consistency [5]. Hence this is an alternative proof of the Proposition 2. Actually, the proposed model is equivalent to the model proposed in [2]; the main difference is that instead of a set of validity variables for the equivalent nodes in the P/A graph that are connected via the equality constraints we use a single validity variable for the corresponding E-node in the equivalence tree and so the new model has a smaller number of variables and constraints.

4 PREFERENCE MODEL WITH ALTERNATIVES

Traditional scheduling problems are optimization problems where the utility function is frequently dependent on time such as makespan, tardiness, and earliness. In this paper we suggest a different type of the utility function that depends on the selection of activities among the alternatives. Such a utility function is more typical for AI planning where actions have cost and the task is to find a plan (a sequence of actions) where the sum of action costs is minimized. Even the earliest formulations of planning problems where the task was to find the shortest plan (a plan with the smallest number of actions) can be formulated as a cost optimization problem (each action has identical cost 1). According to our experience with the real-life scheduling problems, the alternative processes are not of the same cost and the users want to specify that one alternative is preferred to another one. One of the possible ways how to model such preferences is using the cost model from AI planning. Briefly speaking, each action has a non-negative cost and the task is to select a subset of actions where the sum of action costs is minimized and all other constraints are satisfied. In our formal model of P/A graphs we assume only the logical branching constraints (in practice, the temporal and resource constraints must also be assumed) so the problem can be formalized as follows:

Definition 8: Given a P/A graph G such that each node has a non-negative cost and a subset of nodes in G which are assigned to 1, the *P/A graph optimization problem* is the problem of finding a feasible assignment of 0/1 values to all nodes of G which extends the prescribed partial assignment and minimizes the sum of costs of nodes assigned to 1.

The P/A graph optimization problem for $G = (V, E)$ can be naturally modeled as a constrained optimization problem. Let X be the Boolean (0/1) validity variable for each node $x \in V$, cost_x be the cost of node x , and $S \subseteq V$ be the set of nodes assigned to 1. Then the following constraint model describes the problem:

```

minimize  $\sum_{x \in V} X * \text{cost}_x$  under the constraints
 $X = Y$            for each  $(x, y) \in E$  s.t.  $x$  and  $y$  are part of
                    parallel sub-graph
 $X = \sum_{i=1, \dots, k} Z_i$  for each complete alternative sub-graph with
                    the principal node  $x$  and branching nodes
                     $Z_1, \dots, Z_k$ 
 $X = 1$            for each  $x \in S$ 
 $X \in \{0, 1\}$     for each  $x \in V$ 

```

As shown in [2], the above model does not propagate a lot, in particular, arc consistency does not remove all infeasible values. Even if we modify the logical part of the model as described in [2] or earlier in this paper, we still do not achieve global consistency regarding the utility function. The following example demonstrates that:

```

minimize  $A+B+C+D$  under the constraints
 $A = B+C$ 
 $D = B+C$ 
 $A = 1$ 
 $A, B, C, D \in \{0, 1\}$ 

```

In the example we assume the cost of each action to be 1 hence utility is $A+B+C+D$. Clearly, the value of the utility function equals 3 if the logical (branching) constraints are satisfied.

However, the lower bound of the utility function as computed via arc consistency is 1 while the upper bound is 4. If we add the implied constraint $A = D$ as suggested in [2], we obtain global consistency for the logical constraints ($D = 1$), the lower bound of the utility function increases to 2, but the upper bound does not change (it is still 4). In the following section, we shall show how to achieve global consistency for the utility function by integrating logical and preference reasoning by means of the equivalence tree.

5 REASONING WITH PREFERENCES ON ALTERNATIVES

As we already presented, the equivalence tree can be used to define an alternative constraint model for the P/A graph assignment problem and this model achieves global consistency. The main idea was that the equivalent nodes arising from parallel sub-graphs in the P/A graph are kept together in a single E-node of the equivalence tree while the alternative sub-graphs are modeled via the A-nodes. We used 0/1 variable Val_X to describe whether all nodes from the annotation of E-node X (or none of them) belong to the solution sub-graph defined by the nodes of the P/A graph that are assigned to 1. Hence, a feasible assignment of the nodes in the P/A graph corresponds to a feasible assignment of the nodes in the corresponding equivalence tree. In a similar way, we can accumulate the costs of all nodes in the annotation of E-node X : $\text{fCost}_X = \sum_{x \in \text{annotation}(X)} \text{cost}_x$. Then the utility function can be reformulated in terms of the equivalence tree $T = (N, C)$:

$$\sum_{X \in N} \text{Val}_X * \text{fCost}_X.$$

The above reformulation of the utility function strengthens domain filtering (the estimate of bounds) in the same way as adding the implied equality constraints (see the previous section), but it still does not achieve global consistency. We shall now present how to compute the lower and upper bounds of the utility function precisely for any consistent (possibly partial) assignment of variables Val_X . Let $\text{dom}(A)$ be the domain of variable A (possible values to be assigned to A). For each node X (both E-nodes and A-nodes) in the equivalence graph we compute the values minCost_X and maxCost_X using the following **Algorithm 2**:

let the nodes X be visited in the order from leaves to the root

if X is a leaf E-node **then**

case $1 \in \text{dom}(\text{Val}_X)$: $\text{minCost}_X = \text{maxCost}_X = \text{fCost}_X$

case $\text{Val}_X = 0$: $\text{minCost}_X = \infty$, $\text{maxCost}_X = -\infty$

if X is an A-node with $Z_1, \dots, Z_k$ being all its child E-nodes **then**

$\text{minCost}_X = \min\{\text{minCost}_{Z_i} \mid i = 1, \dots, k\}$

$\text{maxCost}_X = \max\{\text{maxCost}_{Z_i} \mid i = 1, \dots, k\}$

if X is a non-leaf E-node with $Y_1, \dots, Y_k$ being all its child A-nodes **then**

case $1 \in \text{dom}(\text{Val}_X)$:

$\text{minCost}_X = \text{fCost}_X + \sum_{i=1, \dots, k} \text{minCost}_{Y_i}$

$\text{maxCost}_X = \text{fCost}_X + \sum_{i=1, \dots, k} \text{maxCost}_{Y_i}$

case $\text{Val}_X = 0$:

$\text{minCost}_X = \infty$, $\text{maxCost}_X = -\infty$

In the algorithm we assume that $\min \emptyset = \infty$ and $\max \emptyset = -\infty$. After running the Algorithm 2, the values minCost and maxCost of the root node of the equivalence tree represent the lower and upper bounds of the utility function provided that at least one node is valid. We do not assume the trivial situation where all nodes are invalid, for which the utility function equals zero. Notice also that

if at least one node is pre-assigned to 1 (it's valid) then $Val_R = 1$ for the root node R of the equivalence tree.

Proposition 4: Let R be the root node of an equivalence tree T and $minCost_R \neq \infty$. Then there exists a feasible assignment of nodes in T such that the utility function equals $minCost_R$ (and similarly for $maxCost_R$).

Proof: The feasible assignment will contain the root node, that is, $Val_R = 1$. From the Definition 7 we know that for each A-node Y that is a child of R , exactly one child Z_i of Y must be valid. Let $j = \text{argmin}\{minCost_{Z_i} \mid i = 1, \dots, k\}$, where $Z_1, \dots, Z_k$ are all children of Y , then put $Val_{Z_j} = 1$ and $Val_{Z_i} = 0$ for $i \neq j$ (if there are more such j then select one). Note that this is possible because there must be some $minCost_{Z_j}$ different from ∞ and hence $1 \in \text{dom}(Val_{Z_j})$. Moreover, $0 \in \text{dom}(Val_{Z_i})$ for $i \neq j$ because otherwise $Val_{Z_j} = 0$ due to global consistency of variables Val_X . This way we continue until we reach the leaf nodes. $Val_{Z_i} = 0$ means that all descendants of Z_i are invalid; $Val_{Z_j} = 1$ means that we select a valid children as above. Clearly, the selected nodes form a feasible assignment and the value of the utility function equals $minCost_R$. A similar approach can be applied to obtain a feasible assignment for $maxCost_R$. ■

Proposition 5: The P/A graph optimization problem for nested P/A graphs is solvable in polynomial time.

Proof: The Algorithm 2 runs in the linear time in the number of nodes of the equivalence tree and Proposition 4 shows how to find the optimal assignment. ■

The Algorithm 2 is used to compute the bounds of the utility function. It is based on idea of propagating the cost information from the leaves to the root. It represents one-way propagation from variables Val_X to the utility function. We can also propagate in the opposite direction from the bounds of the utility function to variables Val_X . This way, we can invalidate nodes ($Val_X = 0$) that cannot be part of the feasible assignment whose value of the utility function is within given bounds. Again, we can utilize the values $minCost$ and $maxCost$ but now they are recomputed in the direction from the root R to the leaves as the proof of Proposition 4 showed. **Algorithm 3** describes a general approach when $minCost_R$ is increased and/or $maxCost_R$ is decreased:

```

let the nodes  $X$  be visited in the order from the root to the leaves
case  $X$  is an E-node (with possible children  $Y_1, \dots, Y_k$ ):
  if  $minCost_X > maxCost_X$  then
     $Val_X = 0$ 
    for  $j = 1$  to  $k$  do  $minCost_{Y_j} = \infty, maxCost_{Y_j} = -\infty$ 
  else
    for  $j = 1$  to  $k$  do
       $minCost_{Y_j} = \max(minCost_{Y_j}, minCost_X - fCost_X - \sum_{i=1, \dots, k, i \neq j} maxCost_{Y_i})$ 
       $maxCost_{Y_j} = \min(maxCost_{Y_j}, maxCost_X - fCost_X - \sum_{i=1, \dots, k, i \neq j} minCost_{Y_i})$ 
case  $X$  is an A-node (with children  $Z_1, \dots, Z_k$  such that  $1 \in \text{dom}(Val_{Z_i})$ ):
  for  $j = 1$  to  $k$  do
     $minCost_{Z_j} = \max(minCost_{Z_j}, minCost_X)$ 
     $maxCost_{Z_j} = \min(maxCost_{Z_j}, maxCost_X)$ 

```

The Algorithm 3 invalidates the only nodes that cannot be part of a feasible assignment, but not all of them. We are currently working on its stronger version.

6 DISCUSSION AND CONCLUSIONS

The paper describes equivalence trees as a compact representation of the logical relations in Nested Temporal Networks with Alternatives. We showed that the equivalence trees can be used for efficient propagation of branching (logical) constraints. We proposed a novel utility function for scheduling problems where the alternative activities are assumed and we showed that the equivalence trees simplify propagation of cost information for that utility function. The utility function is based on summation operation which usually propagates weakly but via integration with logical reasoning we can compute the exact bounds of the utility function and we can also filter-out some activities based on the restricted bounds of the utility function. We focused on efficient logical and cost reasoning that is relatively novel for scheduling problems. To model real-life problems one needs to assume temporal and resource constraints as well. They can be easily added when using constraint satisfaction framework and we already know that if simple temporal constraints are assumed then the P/A graph assignment problem becomes NP-hard [3]. It is still interesting to find out how to integrate logical, temporal, resource, and cost reasoning to achieve stronger domain filtering. An interesting question is also how to combine the utility functions based on time (tardiness, earliness) with the utility function based on the selection of alternatives.

ACKNOWLEDGEMENTS

The research is supported by the Czech Science Foundation under the contract P202/10/1188.

REFERENCES

1. Barták, R. and Čepek, O.: Temporal Networks with Alternatives: Complexity and Model. In *Proceedings of the Twentieth International Florida AI Research Society Conference (FLAIRS)*, pp. 641–646, AAAI Press (2007)
2. Barták, R. and Čepek, O.: Nested Precedence Networks with Alternatives: Recognition, Tractability, and Models. In *Artificial Intelligence: Methodology, Systems, and Applications (AIMSA 2008)*, LNAI 5253, pp. 235–246. Springer Verlag (2008).
3. Barták R., Čepek, O., Hejna, M.: Temporal Reasoning in Nested Temporal Networks with Alternatives. In *Recent Advances in Constraints - CSCP 2007*, LNAI 5129, pp. 17–31. Springer Verlag (2008).
4. Beck, J. Ch. and Fox, M. S.: Constraint-directed techniques for scheduling alternative activities. *Artificial Intelligence* 121, 211–250 (2000)
5. Beeri, C., Fagin, R., Maier, D., and Yannakakis, M.: On the desirability of acyclic database schemes. *Journal of the ACM* 30, 479–513 (1983)
6. Kim, P. Williams, B. and Abramson, M.: Executing Reactive, Model-based Programs through Graph-based Temporal Planning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 487 – 493, (2001)
7. Kuster, J., Jannach, D., and Friedrich, G.: Handling Alternative Activities in Resource-Constrained Project Scheduling Problems. In *Proceedings of Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pp. 1960–1965 (2007)
8. Nuijten, W., Bousonville, T., Focacci, F., Godard, D., and Le Pape, C.: MaScLib: Problem description and test bed design, <http://www2.ilog.com/masclib/>

Using Soft Constraints for Deduction and Abduction in Policy-based Access Authorization¹

Stefano Bistarelli² and Fabio Martinelli³ and Francesco Santini⁴

Abstract. We show that soft constraints can be used to model logical reasoning, that is deduction and abduction. In particular, we focus on the abduction process and we show how it can be implemented with a (soft) constraint removal operator. As a running application example throughout the paper, we reason with access control policies and credentials. In this way, we can associate the level of preference defined by the “softness” of the constraint with a “level” of trust. The main benefit comes during the process of automated access authorization based on trust: soft constraint operations can be easily adopted to measure the level of trust required for each operation. Moreover, when the level is not sufficient, abduction can be used to compute the missing credentials and the levels that grant the access, making the request a (weighted) logical consequence.

1 Introduction and Motivations

In this work we interpret *Logical Reasoning* [16, 17] in the soft constraint field [1], in order to grant an authorization based on credentials and trust derivation. The classical operators as deduction and abduction are implemented by soft constraint operators, having in this way a powerful and expressive tool for the automated computation of these related problems, which also take a preference score into account.

Using logic programming languages for security policy is customary in computer security: in [4, 5] the authors have presented a variant of *RT* language [13] (called it RT^W) that use soft constraints and is able to deal with weights on ground facts and to consequently compute a feedback result for the goal satisfaction. The proposed RT^W framework is able to represent policies: by querying such programs we can infer if the authorization, given a specific set of credentials, is allowed or not with a specific level of trust. For example, a credential can now state that the referred entity is a “student” with a probability of 80% because her/his identity of student is based on what an acquaintance asserts (thus, it is not as certain as declared in IDs). Therefore, also the final authorization decision can be taken according to a trust value related to the composition of all the used credentials, e.g. with a total probability greater than 90% [4, 5].

However, often the user simply wants to obtain a service or to access to a resource, and he does not know which credential is needed [12], or which level of trust is needed to obtain the access. Thus, often the user simply presents a very small set of credentials and expects the system to return enough information to guide him to-

wards the missing credentials that grant him to access. When the user obtains such information, he can decide to present such credentials or, for instance, to buy the necessary credentials with “enough” trust level, in order to definitely access to the service. Two basic services can be used inside authorization systems. The first one is represented by deduction [12, 16]: given a policy and a set of additional facts and events, the service finds out all consequences (actions or obligations) of the policy and the facts, i.e. whether granting the request can be deduced from the policy and the current facts. Abstracting away the details of the policy implementation, we can observe that only one reasoning service is actually used by policy based self-management: deduction.

Access authorization usually needs another reasoning service: abduction [12, 16]. Loosely speaking, abduction is deduction in reverse: given a policy and a request for access to (e.g.) services, it consists in finding the credentials/events that would grant access, i.e. a (possibly minimal) set of facts that added to the policy would make the request a logical consequence. The intuition behind an interactive (client-server) access control system is the following: *i*) initially a client submits a set of credentials and a service request then, *ii*) the server checks whether the request is granted by the access policy according to the client’s set of credentials. If the check fails, *iii*) by using abductive reasoning the server finds a (minimal) solution set of (disclosable) missing credentials that unlocks the desired resource and *iv*) returns them to the client, so that *v*) he can provide them in the second round.

These services provided by logical reasoning are important in autonomic networks of nodes [12], where partners offer services and lightly integrate their efforts into one (hopefully coherent) network. This cross enterprise scenario poses novel security challenges with aspects of trust management systems. Access to services is offered by autonomic nodes on their own and the decision to grant or deny access must rely on attribute credentials sent by the client [12]. Therefore, when speaking about autonomic authorization, the abduction process it is exactly what one node looks for. By using soft constraints, the user wishes to obtain a resource with a specific trust level, and the server answer has to provide not only the expected credential (that the user needs to present), but also the trust level of such credential. The claim is that developing the notion of deduction and abduction operations in soft constraints allows to naturally reason on trust policies: these policies have been initially assembled from the logic point of view, then constraints have been added for expressivity reasons [13]. We have further extended this step by adding levels of trust (i.e. trust scores) [4, 5].

In Sec. 2 we summarize the fundamental notions about soft constraints, while Sec. 3 shows how classical deduction and abduction operations can be implemented by using constraint composition and division, i.e. $\otimes$ and $\oslash$; to complete the presentation of logical rea-

¹ Partially supported by the MIUR PRIN 20089M932N: “Innovative and multi-disciplinary approaches for constraint and preference reasoning”.

² Università di Perugia, Italy, email: bista@dmf.unipg.it

³ Istituto di Informatica e Telematica, CNR Pisa, Italy, email: fabio.martinelli@iit.cnr.it

⁴ Università di Perugia, Italy, email: santini@dmf.unipg.it

soning. Section 4 shows an implementation of deduction/abduction operations by using *Constraint Handling Rules* [10]. At last, Sec. 5 discusses the related work and Sec. 6 draws the final conclusions.

2 Background on Soft Constraints

An absorptive semiring [3] S can be represented as a $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ tuple such that: *i)* A is a set and $\mathbf{0}, \mathbf{1} \in A$; *ii)* $+$ is commutative, associative and $\mathbf{0}$ is its unit element; *iii)* $\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element. Moreover, $+$ is idempotent, $\mathbf{1}$ is its absorbing element and $\times$ is commutative. Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that (see [6]): *i)* $\leq_S$ is a partial order; *ii)* $+$ and $\times$ are monotonic on $\leq_S$; *iii)* $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum; *iv)* $\langle A, \leq_S \rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = \text{lub}(a, b)$ (where *lub* is the *least upper bound*). Informally, the relation $\leq_S$ gives us a way to compare semiring values and constraints. In fact, when we have $a \leq_S b$ (or simply $a \leq b$ when the semiring will be clear from the context), we will say that b is *better* than a .

In [3] the authors extended the semiring structure by adding the notion of *division*, i.e. $\div$, as a weak inverse operation of $\times$. An absorptive semiring S is *invertible* if, for all the elements $a, b \in A$ such that $a \leq b$, there exists an element $c \in A$ such that $b \times c = a$ [3]. If S is absorptive and invertible, then, S is *invertible by residuation* if the set $\{x \in A \mid b \times x = a\}$ admits a maximum for all elements $a, b \in A$ such that $a \leq b$ [3]. Moreover, if S is absorptive, then it is *residuated* if the set $\{x \in A \mid b \times x \leq a\}$ admits a maximum for all elements $a, b \in A$, denoted $a \div b$. With an abuse of notation, the maximal element among solutions is denoted $a \div b$. This choice is not ambiguous: if an absorptive semiring is invertible and residuated, then it is also invertible by residuation, and the two definitions yield the same value.

To use these properties, in [3] it is stated that if we have an absorptive and complete semiring⁵, then it is residuated. For this reason, since all classical soft constraint instances (i.e. *Classical*, *Fuzzy*, *Probabilistic* and *Weighted*) are complete and consequently residuated, the notion of semiring division can be applied to all of them. Therefore, for all these semirings it is possible to use the $\div$ operation as a “particular” inverse of $\times$; its extension to soft constraints, defined as $\oplus$, can be used to remove soft constraints from the store (i.e. the considered set of constraints).

A *soft constraint* [6, 1] may be seen as a constraint where each instantiation of its variables has an associated preference. Given $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and an ordered set of variables V over a (finite) domain D , a soft constraint is a function which, given an assignment $\eta : V \rightarrow D$ of the variables, returns a value of the semiring. Using this notation $C = \eta \rightarrow A$ is the set of all possible constraints that can be built starting from S , D and V . Any function in C involves all the variables in V , but we impose that it depends on the assignment of only a finite subset of them. So, for instance, a binary constraint $c_{x,y}$ over variables x and y , is a function $c_{x,y} : (V \rightarrow D) \rightarrow A$, but it depends only on the assignment of variables $\{x, y\} \subseteq V$ (the *support*, or *scope*, of the constraint). Note that $c\eta[v := d_1]$ means $c\eta'$ where η' is η modified with the assignment $v := d_1$. Notice also that, with $c\eta$, the result we obtain is a semiring value, i.e. $c\eta = a$ with $a \in A$.

Given the set C , the combination function of constraints $\otimes : C \times C \rightarrow C$ is defined as $(c_1 \otimes c_2)\eta = c_1\eta \times c_2\eta$ (see also [6, 1]).

⁵ If S is an absorptive semiring, then S is complete if it is closed with respect to infinite sums, and the distributivity law holds also for an infinite number of summands.

Having defined the operation $\div$ on semirings, the constraint division function $\oplus : C \times C \rightarrow C$ (which subtracts the second constraint from the first one) is instead defined as $(c_1 \oplus c_2)\eta = c_1\eta \div c_2\eta$ [3]. Basing ourselves on the definition of $\div$ and given $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, $a \div b$ is defined as the m maximal element of the set $\{x \in A \mid b \times x \leq a\}$. Therefore, $n \leq_S m$ for every other element n in the set and, extended to constraints, $c_n \sqsubseteq_S c_m \iff c_n(x = n) \leq_S c_m(x = m)$, i.e. the other possible constraints obtained by choosing a different x in $\{x \in A \mid b \times x \leq a\}$ imply the one obtained by choosing the m maximal element. For this reason, the $\oplus$ operator is able to find the minimal explanation (see Sec. 3). Given a constraint $c \in C$ and a variable $v \in V$, the *projection* [6, 1] of c over $V - \{v\}$, written $c \downarrow_{(V \setminus \{v\})}$ is the constraint c' such that $c'\eta = \sum_{d \in D} c\eta[v := d]$. Informally, to project means eliminating some variables from the support.

Informally, performing the $\otimes$ or the $\oplus$ between two constraints means building a new constraint whose support involves all the variables of the original ones, and which associates with each tuple of domain values for such variables a semiring element which is obtained by multiplying or, respectively, dividing the elements associated by the original constraints to the appropriate sub-tuples. The partial order $\leq_S$ over C can be easily extended among constraints by defining $c_1 \sqsubseteq c_2 \iff c_1\eta \leq c_2\eta$. Consider the set C and the partial order $\sqsubseteq$. Then, an entailment relation $\vdash \subseteq \wp(C) \times C$ is defined s.t. for each $C \in \wp(C)$ and $c \in C$, we have $C \vdash c \iff \bigotimes C \sqsubseteq c$ (see also [1]). Therefore we can say that if, given a constraint store σ , $\sigma \sqsubseteq c$ (or $\sigma \vdash c$), then c is a *logical consequence* of σ .

Considering a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, a domain of the variables D , an ordered set of variables V and the corresponding structure C , then $S_C = \langle C, \otimes, \oplus, \mathbf{1}, \exists_x, d_{x,y} \rangle$ is a cylindric constraint system (“*a la Saraswat*”)⁶.

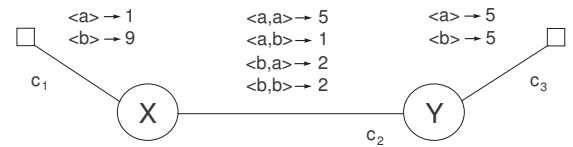


Figure 1. A soft CSP based on a Weighted semiring.

A SCSP Example. Figure 1 shows a weighted SCSP as a graph: the used semiring is the *Weighted* semiring, i.e. $\langle \mathbb{R}^+, \min, \hat{+}, \infty, 0 \rangle$ ($\hat{+}$ is the arithmetic plus operation). Variables and constraints are represented respectively by nodes and by arcs (unary for c_1 and c_3 , and binary for c_2), and semiring values are written to the right of each tuple. $D = \{a, b\}$. The solution of the CSP in Fig. 1 associates a semiring element to every domain value of variables X and Y by combining all the constraints together, i.e. $Sol(P) = \bigotimes C$. For instance, for the tuple $\langle a, a \rangle$ (that is, $X = Y = a$), we have to compute the sum of 1 (which is the value assigned to $X = a$ in constraint c_1), 5 (which is the value assigned to $\langle X = a, Y = a \rangle$ in c_2) and 5 (which is the value for $Y = a$ in c_3). Hence, the resulting value for this tuple is 11. For the other tuples, $\langle a, b \rangle \rightarrow 7$, $\langle b, a \rangle \rightarrow 16$ and $\langle b, b \rangle \rightarrow 16$. The *blevel* for the example in Fig. 1 is 7, related to the solution $X = a, Y = b$.

⁶ Notice that in SCCP, algebraicity is not required, since the algebraic nature of C strictly depends on the properties of the semiring [1].

3 Logical Reasoning with Soft Constraints

In this Section we redefine two basic logical reasoning processes [16], deduction and abduction, by using the framework based on soft constraints and presented in Sec. 2. To accomplish this, we adopt the $\otimes$, $\oplus$ and $\sqsubseteq$ operators as summarized in Sec. 2.

Deduction. Deduction means determining the conclusion, thus to use a rule and its precondition to make a conclusion [12, 16]. Constraint solving techniques are frequently based on constraint propagation, a technique that can be seen as a specific form of deduction. An Example is “When a paper is poorly presented, it receives a bad review. The paper I am reviewing is poorly written, then I write a bad review”. An argument is said to be deductive when the truth of the conclusion necessarily follows or is a logical consequence of the truth of the premises and (consequently) its corresponding conditional is a necessary truth. Deductive arguments are said to be valid or invalid, never true or false. A deductive argument is valid if and only if the truth of the conclusion actually does follow necessarily (or is indeed a logical consequence of) the premises and (consequently) its corresponding conditional is a necessary truth. In Def. 1 we define deduction through soft constraints:

Definition 1 [Deduction with soft constraints] *Given a soft constraint store σ which represents the current knowledge and a soft constraint c , if $\sigma \sqsubseteq c$ then the store entails (i.e. deduces) c .*

Therefore, w.r.t. Def. 1, the store σ represents the premise and c a logical consequence of σ . The $\vdash$ (or $\sqsubseteq$, see Sec. 2) consequently implements the deduction operator as for the crisp $\vdash$ entailment operator [8].

As introduced in Sec. 1, we use a running example based on Policy-based Access Control to show how “soft” deduction and abduction can be used. We suppose that the soft constraint store contains all the collected information, i.e. the access policy and the credentials presented by the requestor and other parties, that is, for example, external databases. A credential, in a general definition, is an attestation of qualification, competence, or authority issued to an individual by a third party with the authority or the competence to do so. A policy describes the rules that grant the authorization. With respect to Def. 1, the σ store represents the policy p and the collected credentials c , i.e. $\sigma = p \otimes c$, and the store entails (i.e. deduces) the access request r if $\sigma \sqsubseteq r$. An example of policy and credentials is given in the program in Tab. 1: we adopt the RT_0^W language [4, 5], a weighted extension of the RT_0 language [13].

```
EPub.disct ← EPub.preferred  $\cap$  EPub.brightStudent.  
EPub.preferred ← EOrg.highBudget  $\cap$   
EPub.oldCustomer.  
EPub.brightStudent ←  
EPub.goodUniversity.highMarks.  
EPub.goodUniversity ← ABU.accredited.  
ABU.accredited ← { StateU, 9 }.  
StateU.highMarks ← { Alice, 8 }.  
EOrg.highBudget ← { Alice, 6 }.  
EOrg.oldCustomer ← { Alice, 7 }.
```

Table 1. An example in RT_0^W , with weights associated to the credentials.

The example in Tab. 1 describes a fictitious Web publishing service, *EPub*, which offers a discount to anyone who is both a preferred customer and a bright student. *EPub* delegates the authority

over the identification of preferred customers to its parent organization, *EOrg*. In order to be evaluated as a preferred customer, *EOrg* must issues two different types of credentials stating that the customer is not new (i.e. *EOrg.oldCustomer*) and has already spent some money in the past (i.e. *EOrg.highBudget*). *EOrg* assigns a cost value to both these two credentials to quantify its evaluation. *EPub* delegates the authority over the identification of bright students to the entities that are accredited universities. To identify such universities, *EPub* accepts accrediting credentials issued by the fictitious *Accrediting Board for Universities (ABU)*. *ABU* evaluates a university with a fuzzy score and each university evaluates its enrolled students. A student is bright if she/he is both enrolled in a good university and has high marks. To solve the example in Tab. 1, we use a *Weighted* semiring $S_{Weighted} = \langle \mathbb{R}^+, \min, \hat{+}, +\infty, 0 \rangle$, where $\hat{+}$ is the arithmetic sum and where the preference levels in $\mathbb{R}^+$ represents the money cost that is needed to buy or retrieve a given credential (e.g. the cost to pay the office supplies and the clerk service); for example, $StateU.highMarks \leftarrow \langle Alice, 8 \rangle$ in Tab. 1 certifies that Alice has obtained a good number of high marks for the exams completed at the StateU university (the credential is issued by StateU), and the cost associated with this credential is 8 euro.

The last four rules represent the weighted credentials presented to the authorization entity, while the other rules consist in the access policy. The example, together with the deduction process, is described in the next paragraph. The final money cost, obtained by composing together all the values of the used credentials, can be compared with a threshold to authorize the discount: e.g. only entities whose set of credentials produced a score greater than 7 are authorized. The four credentials, which are all used by the policy rules, prove that Alice is eligible for the discount with a money cost of 30: $6\hat{+}7\hat{+}8\hat{+}9 = 30$ euro.

By considering Def. 1 and the example in Tab. 1, we can use the deduction operation in the following simple and crisp example (also classical crisp problems can be cast in the soft framework [6, 1] by using the *Classical* semiring $\langle \{true, false\}, \vee, \wedge, false, true \rangle$): if $\sigma = Student(Alice) \otimes HighMarks(Alice) \otimes (Student(x) \wedge HighMarks(x) \sqsubseteq Access(x))$, then $\sigma \sqsubseteq Access(Alice)$, we can deduce that Alice can access. The policy in Tab. 1 has been also implemented in *CIAO Prolog* in [5].

Abduction. Abduction consists in using the conclusion and the rule to support that the precondition could explain the conclusion [12, 16]. For example, “When a paper is poorly presented, it receives a bad review. I received a bad review for my paper, therefore it is poorly presented”. Therefore, if the σ does not store enough information to satisfy c (i.e. the c conclusion cannot be obtained with deduction), it is interesting to automatically obtain the missing information that would allow to satisfy c . In Def. 2 we define abduction by using the $\oplus$ operation presented in Sec. 2:

Definition 2 [Abduction with soft constraints] *Given a soft constraint store σ and a constraint c , then the abduction process is aimed at finding a constraint d , such that $\sigma \otimes d \sqsubseteq c$, i.e. $d = c \oplus \sigma$.*

Notice that the division operator we are adopting is a deterministic operator (see Sec. 2) and provides a single solution d (which is also the minimal solution, as explained later in this section). Considering Def. 2, the trivial case is when $\sigma \sqsubseteq c$ since $c \oplus \sigma = \bar{1}$, that is nothing must be added to the store in order to satisfy c , which is already implied by it. Following the definition of $\div$ (see Sec. 2), the abducted constraint is an approximation since the operator returns the maximum in $\{x \in A \mid b \times x \leq a\}$: this approximation d contains less or

equal information than the real result u , i.e. $u \sqsubseteq d$.

Access Control for autonomic communication needs the abduction reasoning service [19]. The key idea is that in an autonomic network a client may have the right credentials but may not know them and thus an autonomic communication server needs a way to avoid leaving the client stranded. If we consider our example on policy-based access control, $\sigma = p \otimes c$ represent the policy p and a c credential, and the abducted d constraint is such that $\sigma \otimes d \sqsubseteq r$, i.e. $d = r \oplus \sigma$, where d is the abducting credential for the access request r .

An explanation is an input which can explain the result (via the given the inference rules). A minimal explanation, is one which makes just as many assumptions as needed to obtain an explanation [8]. The concept of minimal explanation is important because it consists in the minimal amount of information needed to obtain the desired consequence: disclosing more credentials could lead to privacy problems since more than enough information is revealed. The abduction operation in Def. 2 is able to find the minimal (in this case, soft) explanation as proposed in [8] for the crisp version, according to the definition of $\div$, and consequently $\oplus$ shown in Sec. 2 and in [3]. Therefore, the d constraint in Def. 2 is the minimal explanation of c .

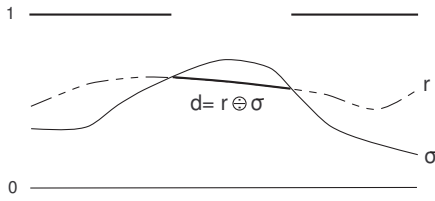


Figure 2. The representation of σ , the request r and the abducted soft constraint d , represented with the thick line.

A graphical intuition of the “soft” abduction definition is given in Fig. 2. There we consider the *Fuzzy* semiring $S_{Fuzzy} = \langle [0..1], \max, \min, 0, 1 \rangle$, that is we analyze the truth degree connected to a credential and evaluated by the entity which signs and issues it, as proposed for the examples in [4, 5]. Suppose that the soft constraints σ and r (and consequently $d = r \oplus \sigma$) have as their support only the variable x , which represents the entity that requests the authorization; these constraints are represented as functions in x in Fig. 2. Suppose that $\sigma = Student(Alice) \otimes (Student(x) \wedge HighMarks(x) \sqsubseteq Access(x))$, and that $r = Access(x)$, then $d = HighMarks(x)$. Moreover, if $\sigma(x=Alice) = 0.9$ and $r(x=Alice) = 0.7$, then $d(x=Alice) = 0.7$. Thus Alice needs to attest the fact that she has high marks with a trust score 0.7 or greater. Where $\sigma(x) \leq r(x)$, then $d(x) = 1$, otherwise $d(x) = r(x)$, according to the definition of $\div$ given in [3]:

$$a \div b = \max\{x \mid \min\{b, x\} \leq a\} = \begin{cases} 1 & \text{if } b \leq a \\ a & \text{if } a < b \end{cases}$$

Now suppose to consider the *Weighted* semiring $S_{Weighted} = \langle \mathbb{R}^+, \min, \hat{+}, +\infty, 0 \rangle$. If $\sigma(x=Alice) = 7$ and $r(x=Alice) = 10$, then $d(x=Alice) = 3$ ($d = r \oplus \sigma$). Since $d = HighMarks(x)$, this operation suggests Alice to provide a credential attesting that she has *HighMarks* and to spend at least 3 euro, in order to be able to access. The definition of $\div$ (i.e. the arithmetic subtraction $\hat{-}$) is:

$$a \div b = \min\{x \mid b \hat{+} x \geq a\} = \begin{cases} 0 & \text{if } b \geq a \\ a \hat{-} b & \text{if } a > b \end{cases}$$

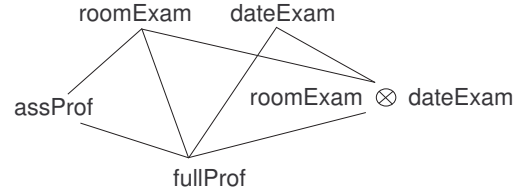


Figure 3. A lattice representing the strength of roles: *fullProf* entails all the other roles.

Notice that the credentials can be seen as partially ordered: *assistant professor* $<$ *associate professor* $<$ *full professor*, that is the *full professor* credential provides higher access capabilities w.r.t. *assistant professor*. This partial order can be easily represented with semiring (see Sec 2). This role hierarchy, where higher the role in the hierarchy, more powerful it is, is adopted also in [12]. A role dominates another role if it is higher in the hierarchy and there is a direct path between them in the lattice representing the hierarchy of roles. For example, $fullProf(X) \sqsubseteq dateExam(X) \otimes roomExam(X)$, but $assProf(X) \sqsubseteq roomExam(X)$ only, i.e. the assistant professor has less rights. The complete lattice of roles is shown in Fig. 3.

It is worth to notice that our $\oplus$ operation can be considered as a “relaxation” of the store, and not only as a strict removal of the token representing the constraint, because in soft constraints we do not have the concept of token. Thus, if we perform $\sigma \oplus c$ during the abduction phase, c can be removed even if it is different from any other constraints previously added to σ ; therefore, it is also possible to remove only a part of a piece of information (e.g. a part of a credential). For example, considering the five soft constraints reported in Fig. 4 and the *Weighted* semiring, $c_1 \otimes c_2 = c_3$, and at this point we can also remove c_4 from c_3 , that is $c_3 \oplus c_4 = c_5$, even if c_4 has been not already added to the store before, but it is however implied by it. To show this concept with a crisp example, it is like if the constraint store is equivalent to *I have a car* and we could remove *A tire of my car is punctured* obtaining *I have a car with three not punctured tyres*.

$$\begin{aligned} c_1 : (\{x\} \rightarrow \mathbb{N}) &\rightarrow \mathbb{R}^+ & \text{s.t. } c_1(x) &= x + 3 \\ c_2 : (\{x\} \rightarrow \mathbb{N}) &\rightarrow \mathbb{R}^+ & \text{s.t. } c_2(x) &= x + 5 \\ c_3 : (\{x\} \rightarrow \mathbb{N}) &\rightarrow \mathbb{R}^+ & \text{s.t. } c_3(x) &= 2x + 8 \\ c_4 : (\{x\} \rightarrow \mathbb{N}) &\rightarrow \mathbb{R}^+ & \text{s.t. } c_4(x) &= x + 1 \\ c_5 : (\{x\} \rightarrow \mathbb{N}) &\rightarrow \mathbb{R}^+ & \text{s.t. } c_5(x) &= x + 7 \end{aligned}$$

Figure 4. Five weighted soft constraints; notice that $c_3 = c_1 \otimes c_2$ and $c_5 = c_3 \oplus c_4$.

4 Deduction and Abduction in CHR

In this section we implement the deduction and abduction operations presented in Sec. 3 by using the *Constraint Handling Rules* [10] (CHR) language.

CHR is a high-level language designed for writing user-defined constraint systems. It is essentially a committed-choice language consisting of guarded rules that rewrite constraints into simpler ones until they are solved. There are three kinds of CHR rules: *simplification*, *propagation* and *simplagation*. Simplification rules replace

user-defined constraints by simpler ones. Propagation rules add new redundant constraints that may be necessary to do further simplifications. A simplification rule is equivalent to a simplification rule with some of the heads repeated in the body. On a simplification rule only the heads after the $\backslash$ sign are removed. CHR libraries are available in most Prolog implementations: in particular, for the following examples we have used *SWI-Prolog* [20]. The complexity of the deduction and abduction is polynomial in the length of the program, since at most all the CHR rules need to be checked for their applicability.

```
:- use_module(library(chr)).

:- chr_constraint preferred/2, brightStudent/2, highBudget/3,
    oldCustomer/3, highMarks/3, goodUniversity/2,
    accredited/3, discount/2, access/1.

policyrule1 @ preferred(X, A), brightStudent(X, B), discount(X, T) <=>
    (A+B)>= T | access(X).
policyrule2 @ highBudget(Z, X, A), oldCustomer(Z, X, B) <=>
    preferred(X, C), C is (A+B).
policyrule3 @ goodUniversity(X, A), highMarks(X, Y, B) <=>
    brightStudent(Y, C), C is (A+B).
policyrule4 @ accredited(X, Y, A) <=> goodUniversity(Y, A).

1 ?- accredited(aBU, stateU, 9), highMarks(stateU, alice, 8),
    highBudget(eOrg, alice, 6), oldCustomer(eOrg, alice, 7),
    discount(alice, 25).

access(alice)

2 ?- accredited(aBU, stateU, 9), highMarks(stateU, alice, 8),
    highBudget(eOrg, alice, 6), discount(alice, 25).

brightStudent(alice, 17)
highBudget(eOrg, alice, 6)
discount(alice, 25)
```

Figure 5. Deduction with CHR rules: solving the example in Tab. 1.

In Fig. 5 we implement the deduction process by translating the RT_0^W program in Tab. 1. Given the five simplification rules (i.e., *policy1*, ..., *policy4*), which represent the access policy, and the first query, (i.e. 1 ?-), that is *accredited(aBU, stateU, 9)*, *highMarks(stateU, alice, 8)*, *highBudget(eOrg, alice, 6)*, *oldCustomer(eOrg, alice, 7)*, *discount(alice, 25)*, which represents the set of presented credentials and the discount request (i.e. *discount(alice, 25)*), we are able to deduce that Alice can have discount. In fact the system answers with *access(alice)*. The solver can fire the first rule (i.e. *policyrule1*) because the sum of the trust level of the credentials, which is $= 30$ is greater than the requested threshold in the discount request, which is 25. As a remind, in this example (as for the one in Tab. 1) we use the weighted semiring, that is the $\times$ operator of the semiring is modeled with the arithmetic addition.

The second query (i.e. 2 ?-) in Fig. 5 is presented to show what happens when any of the credentials is not presented to the authorization system: without the *oldCustomer(eOrg, alice, 7)* credential missing in the query, the CHR rules in Fig. 5 are not able to compute the consequences in *policyrule2* rule, that is Alice is *preferred* for *eOrg*.

In this case, that is when the *access* constraint is not added to the store and, therefore, the access is not granted, the rules in Fig. 6 can be used to abduce the missing credentials with the (minimum) required level of trust. The query is now represented by the store obtained at the end of the failed access in Fig. 5, i.e. *brightStudent(alice, 17)*, *highBudget(eOrg, alice, 6)*, *discount(alice, 25)*. The program an-

```
:- use_module(library(chr)).

:- chr_constraint preferred/2, brightStudent/2, highBudget/3, oldCustomer/3,
    discount/2, sum/1.

abdrule1 @ discount(X, A), brightStudent(X, B) <=> C is (A-B),
    preferred(X, C).
abdrule2 @ discount(X, A) <=> preferred(X, A).
abdrule3 @ preferred(X, A), highBudget(Z, X, B) <=> C is (A-B),
    oldCustomer(Z, X, C).
abdrule4 @ preferred(X, A), oldCustomer(Z, X, B) <=> C is (A-B),
    highBudget(Z, X, C).
abdrule5 @ preferred(X, A) <=> oldCustomer(Z, X, B),
    highBudget(Z, X, C), sum(A).

1 ?- brightStudent(alice, 17), highBudget(eOrg, alice, 6), discount(alice, 25).

oldCustomer(eOrg, alice, 2)

2 ?- discount(alice, 25).

highBudget(_G88354, alice, _G88403)
oldCustomer(_G88354, alice, _G88356)
comb(25)
```

Figure 6. Abduction with CHR rules.

swers that we need the *oldCustomer* credential with a level of trust equal or greater than 2: the sum of the weights of the other two credentials is 23 and the discount is required with 25. Notice that, w.r.t. the policy in Tab. 1, in Fig. 6 we show only a subset of the abduction rules for sakes of brevity, i.e. the subset concerning the branch under *EPub.preferred*. This subset represent only the rules needed to answer to the first abduction query, since the *brightStudent* has been already deduced by the program in Fig. 5: the other (not shown here) abduction rules are not used in this example.

In the second query in Fig. 6 the program abduce both the missing credentials (i.e. *oldCustomer* and *highBudget*) needed to receive the discount. Moreover, it replays with a further constraint stating that the combination (i.e. the $\times$ semiring operator) of trust scores for these credentials must be 25 at least (i.e. *comb(25)*).

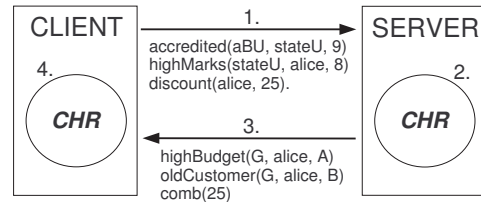


Figure 7. Trust negotiation with CHR solvers.

The deduction/abduction processes in CHR can be adopted to implement trust negotiation, as represented in the system architecture in Fig. 7: 1) the client sends its credentials in order to be authorized, 2) the server uses its CHR solver to deduce the authorization or to abduce the missing credentials, that 3) are sent back to the client; 4) the client then uses its solver to decide, for example, the appropriate level of trust to be assigned to each missing credential, in order to reach (by combining them) the required threshold (i.e. 25 for the

example in Fig. 7). In this way, the negotiation process is interactive.

5 Related Work

The importance of the deduction and abduction operations for policy-based management of autonomic networks has been already highlighted in [12]. An Autonomic Network crosses organizational boundaries and is provided by entities that see each other just as business partners. Policy-based network management already requires a paradigm shift in the access control mechanism, from identity-based access control to trust management and negotiation. In [12] the authors present an algorithm whose operations are expressed with logic, and so the access authorization procedure is not directly implemented with a tool in the paper, as we instead do in this work.

Among the most noticeable works concerning logical reasoning and (crisp) constraints, we need to cite Maher, e.g. [14] where he investigates abduction applied to fully-defined predicates, specifically linear arithmetic constraints over the real numbers. In [15], Maher and Huang address the problem of computing and representing answers of constraint abduction problems over the Herbrand domain. This problem is of interest when performing type inference involving generalized algebraic data types.

In [18] the authors present a CHR-based tool for detecting security policy inconsistencies, where policy are represented by CHR rules. With respect to [18], we use CHR rules to perform the deduction and abduction reasonings instead.

In [9] the authors consider configuration problems with preferences rather than just hard constraints, and we analyze and discuss the features that such preference-based configurators should have. In particular, these configurators, should provide explanations for the current state, implications of a future choice.

Concerning instead systems to implement abduction and deduction processes, abduction reasoning has been already realized in *HYPROLOG* [7] (available also in *SWI-Prolog*), which is an extension of *Prolog* and *CHR* with abduction and assumptions. The system is basically implemented by a compiler that translates the *HYPROLOG* syntax in a rather direct way into *Prolog* and *CHR*.

Another implementation of this kind of system is represented by *ACLP* [11]; *ACLP* is a system which combines abductive reasoning and constraint solving by integrating the frameworks of *Abductive Logic Programming* and *Constraint Logic Programming* (CLP). The *ACLP* system is currently implemented on top of the CLP language of *ECLiPSe* as a meta-interpreter exploiting its underlying constraint solver for finite domains.

6 Conclusions

In this paper we have proposed how two important logical reasoning processes, i.e. deduction and abduction can be modeled and solved with soft constraints [1]. We mainly focused on the abduction phase, since deduction was already available in the framework with the classical (soft) constraint entailment. Abduction can be implemented through the use of the $\oplus$ operator (see Sec. 2).

Deduction and abduction are presented with the help of a running example based on policy-based access control, in order to immediately describe a possible application of these operators and their importance to derive knowledge. We manage trust-based access authorization with weighted credentials, i.e. credentials where the associated level of trust can be used to enrich the authorization process. The proposed framework can be used to improve logical reasoning processes in autonomic networks of nodes [12], since these services (i.e. deduction and abduction) help the self-managing characteristics

of distributed computing resources: a node sometimes needs to independently produce the missing information (through an abduction process).

As future work, we plan to extend in *SWI-Prolog* the existent *CHR* module implementing soft constraints [2], by adding the $\div$ operator inside the soft module. In this way we could implement a stand-alone policy-based authorization system based on constraint programming. The distributed negotiation architecture shown in Fig. 7 can be enriched in order to model the mutual disclosure of credentials between strangers when credentials are sensitive.

REFERENCES

- [1] S. Bistarelli, *Semirings for Soft Constraint Solving and Programming*, volume 2962 of *LNCS*, Springer, 2004.
- [2] S. Bistarelli, T. Frühwirth, and M. Marte, ‘Soft constraint propagation and solving in CHRs’, in *SAC ’02: Proc. of the ACM symposium on Applied computing*, pp. 1–5. ACM Press, (2002).
- [3] S. Bistarelli and F. Gadducci, ‘Enhancing constraints manipulation in semiring-based formalisms’, in *ECAI ’06: European Conference on Artificial Intelligence*, pp. 63–67, (2006).
- [4] S. Bistarelli, F. Martinelli, and F. Santini, ‘A semantic foundation for trust management languages with weights: An application to the RT family’, in *Autonomic and Trusted Computing, 5th International Conference*, volume 5060 of *LNCS*, pp. 481–495. Springer, (2008).
- [5] S. Bistarelli, F. Martinelli, and F. Santini, ‘Weighted datalog and levels of trust’, in *ARES: Conference on Availability, Reliability and Security*, pp. 1128–1134. IEEE Computer Society, (2008).
- [6] S. Bistarelli, U. Montanari, and F. Rossi, ‘Semiring-based constraint satisfaction and optimization’, *J. ACM*, **44**(2), 201–236, (1997).
- [7] H. Christiansen and V. Dahl, ‘Hyprolog: A new logic programming language with assumptions and abduction’, in *Logic Programming, 21st International Conference, ICLP 2005*, volume 3668, pp. 159–173. Springer, (2005).
- [8] C. Codognot and P. Codognot, ‘Abduction and concurrent logic languages’, in *ECAI ’94: European Conference on Artificial Intelligence*, pp. 75–79. John Wiley and Sons, (1994).
- [9] E. C. Freuder, C. Likitvivatanavong, M. Moretti, F. Rossi, and R. J. Wallace, ‘Computing explanations and implications in preference-based configurators’, in *International Workshop on Constraint Solving and CLP*, volume 2627 of *LNCS*, pp. 76–92. Springer, (2002).
- [10] T. W. Frühwirth, ‘Constraint handling rules’, in *Selected Papers from Constraint Programming*, pp. 90–107, London, UK, (1995). Springer-Verlag.
- [11] A. C. Kakas, ‘ACLP: Integrating abduction and constraint solving’, *CoRR*, **cs.AI/0003020**, (2000).
- [12] H. Koshutanski and F. Massacci, ‘A negotiation scheme for access rights establishment in autonomic communication’, *J. Network Syst. Manage.*, **15**(1), 117–136, (2007).
- [13] N. Li, J. C. Mitchell, and W. H. Winsborough, ‘Design of a role-based trust-management framework’, in *SP ’02: Proc. of Security and Privacy*, pp. 114–130. IEEE Computer Society, (2002).
- [14] M. J. Maher, ‘Abduction of linear arithmetic constraints’, in *In ICLP ’05: Conf. on Logic Programming*, volume 3668, pp. 174–188. Springer, (2005).
- [15] M. J. Maher and G. Huang, ‘On computing constraint abduction answers’, in *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 5330 of *LNCS*, pp. 421–435. Springer, (2008).
- [16] T. Menzies, ‘Applications of abduction: knowledge-level modelling’, *Int. J. Hum.-Comput. Stud.*, **45**(3), 305–335, (1996).
- [17] H. E. Pople, ‘On the mechanization of abductive logic’, in *IJCAI’73: Proc. of the International Joint Conference on Artificial Intelligence*, pp. 147–152, San Francisco, CA, USA, (1973). Morgan Kaufmann Publishers Inc.
- [18] C. Ribeiro, A. Zuquete, P. Ferreira, and P. Guedes, ‘Security policy consistency’, *CoRR*, **cs.LO/0006045**, (2000).
- [19] M. Shanahan, ‘Prediction is deduction but explanation is abduction’, in *IJCAI’89: Proc. of the International Joint Conference on Artificial Intelligence*, pp. 1055–1060, (1989).
- [20] J. Wielemaker, ‘An overview of the *SWI-Prolog* programming environment’, in *Proc. of the 13th International Workshop on Logic Programming Environments*, pp. 1–16, Heverlee, Belgium, (2003). Katholieke Universiteit Leuven.

Finding the next solution in constraint- and preference-based knowledge representation formalisms

Ronen Brafman,¹ Francesca Rossi,² Domenico Salvagnin,² K. Brent Venable² and Toby Walsh³

Abstract. In constraint or preference reasoning, a typical task is to compute a solution, or an optimal solution. However, when one has already a solution, it may be important to produce the next solution following the given one in a linearization of the solution ordering, where more preferred solutions are ordered first. In this paper, we study the computational complexity of finding the next solution in some common preference-based representation formalisms. We show that this problem is hard in general CSPs, but it can be easy in tree-shaped CSPs and tree-shaped fuzzy CSPs. However, it is difficult in weighted CSPs, even if we restrict the shape of the constraint graph. We also consider CP-nets, showing that the problem is easy in acyclic CP-nets, as well as in constrained acyclic CP-nets where the (soft) constraints are tree-shaped and topologically compatible with the CP-net.

1 Introduction and motivation

In combinatorial satisfaction and optimization problems, the main task is finding a satisfying or optimal solution. There have been many efforts to develop efficient algorithms to perform such tasks, to study the computational complexity of this problem in general, and to find islands of tractability [7]. Another important task is to be able to compare two solutions and to say if one dominates another [2]. In this paper, we address another task that is crucial in many scenarios. When one has already a solution, it can be useful to be able to produce the next solution following the given one in the solution ordering where more preferred solutions are ordered first. If the solution ordering has ties or incomparability, the next solution could be any solution which is tied or incomparable to the given one. In general, however, the next solution is the solution following the given one in a linearization of the solution ordering. The problem of finding the next solution is related to the problem of enumerating all the solutions of a model [5] and to the ranking problem [8], although it is quite different from the latter because we assume only a reference solution in input, and not all previous ones.

In this paper we study the computational complexity of the problem of returning the next solution in some constraint and preference-based formalisms. We show that this is a hard problem in constraint satisfaction problems (CSPs), but it can be easy in tree-shaped CSPs [6] and tree-shaped fuzzy CSPs [10]. However, it is difficult in weighted CSPs, even if we restrict the shape of the constraint graph. Nevertheless, this hardness is only weak, since we give a pseudo-polynomial algorithm to find the next solution in weighted CSPs.

Moreover, we also show that this problem is easy in acyclic CP-nets [2], as well as in constrained acyclic CP-nets [3] where the (soft) constraints are tree-shaped and topologically compatible with the CP-net graph.

We intend to look for other tractable cases, and to investigate scenarios where the sufficient conditions for tractability, considered in this paper, naturally hold. We also plan to test experimentally how difficult it is in practice to find the next solution.

Parts of this paper appeared already in [4]. Due to lack of space, some formal proofs are omitted and others are only sketched.

2 Background

Hard and soft constraints. A soft constraint [10, 1] is a constraint [7] where each instantiation of its variables has an associated value from a (totally or partially ordered) set coming from a c-semiring. A c-semiring is defined by $\langle A, +, \times, 0, 1 \rangle$ where A is this set of values, $+$ is a commutative, associative, and idempotent operator, $\times$ is used to combine preference values and is associative, commutative, and distributes over $+$, 0 is the worst element, and 1 is the best element. The c-semiring induces a partial or total order $\leq$ over preference values where $a \leq b$ iff $a + b = b$.

A classical CSP [7] is just a soft CSP where the chosen c-semiring is $SCSP = \langle \{false, true\}, \vee, \wedge, false, true \rangle$. Fuzzy CSPs [10] are instead modeled with $FCSP = \langle [0, 1], max, min, 0, 1 \rangle$. That is, we maximize the minimum preference. For weighted CSPs, the c-semiring is $WCSP = \langle \mathbb{R}^+, min, +, +\infty, 0 \rangle$: preferences are interpreted as costs from 0 to $+\infty$, and we minimize the sum of costs.

Given an assignment s to all the variables of an SCSP P , its preference, written $pref(P, s)$, is obtained by combining the preferences associated by each constraint to the subtuples of s referring to the variables of the constraint. For example, in fuzzy CSPs, the preference of a complete assignment is the minimum preference given by the constraints. In weighted constraints, it is instead the sum of the costs given by the constraints. An optimal solution of an SCSP P is then a complete assignment s such that there is no other complete assignment s' with $pref(P, s) <_S pref(P, s')$.

Constraint propagation in classical CSPs reduces variable domains, and thus improves search performance. For some classes of constraints, constraint propagation is enough to solve the problem [6]. This is the case for tree-shaped CSPs, where directional arc-consistency, applied bottom-up on the tree shape of the problem, is enough to make the search for a solution backtrack-free. Given a variable ordering o , a CSP is directional arc-consistent (DAC) if, for any two variables x and y linked by a constraint c_{xy} , such that x precedes y in the ordering o , we have that, for every value a in the domain of

¹ Department of Computer Science Ben Gurion University Beer-Sheva, Israel
Email: brafman@cs.bgu.ac.il

² Dipartimento di Matematica Pura ed Applicata, University of Padova, Italy,
Email: {frossi,salvagni,kvenable}@math.unipd.it

³ NICTA and UNSW Sydney, Australia Email: Toby.Walsh@nicta.com.au

x there is a value b in the domain of y such that (a, b) satisfies c_{xy} .

Constraint propagation can be applied also to soft CSPs, and it maintains the usual properties, as in classical CSPs, if the soft constraint class is based on an idempotent semiring (that is, one where the combination operator is idempotent). This is the case for fuzzy CSPs, for example. As for classical CSPs, DAC is enough to find the optimal solution to a fuzzy CSP when the problem has a tree shape [10].

Fuzzy CSPs can also be solved via a cut-based approach. Given a fuzzy CSP P , an α -cut of P , where α is between 0 and 1, is a classical CSP with the same variables, domains, and constraint topology as the given fuzzy CSP, and where each constraint allows only the tuples that have preference above α in the fuzzy CSP. We will denote such a problem by $\text{cut}(P, \alpha)$. The set of solutions of P with preference greater than or equal to α coincides with the set of solutions of $\text{cut}(P, \alpha)$.

CP-nets. CP-nets [2] are a graphical model for compactly representing conditional and qualitative preference relations. CP-nets are sets of *ceteris paribus* (cp) preference statements. For instance, the statement *I prefer red wine to white wine if meat is served.* asserts that, given two meals that differ *only* in the kind of wine served *and* both containing meat, the meal with red wine is preferable to the meal with white wine. A CP-net has a set of features $F = \{x_1, \dots, x_n\}$ with finite domains $\mathcal{D}(x_1), \dots, \mathcal{D}(x_n)$. For each feature x_i , we are given a set of *parent* features $Pa(x_i)$ that can affect the preferences over the values of x_i . This defines a *dependency graph* in which each node x_i has $Pa(x_i)$ as its immediate predecessors. Given this structural information, the agent explicitly specifies her preference over the values of x_i for *each complete assignment* on $Pa(x_i)$. This preference is assumed to take the form of total or partial order over $\mathcal{D}(x_i)$. An *acyclic* CP-net is one in which the dependency graph is acyclic.

Consider a CP-net whose features are A, B, C , and D , with binary domains containing f and $\bar{f}$ if F is the name of the feature, and with the preference statements as follows: $a \succ \bar{a}, b \succ \bar{b}, (a \wedge b) \vee (\bar{a} \wedge \bar{b}) : c \succ \bar{c}, (a \wedge \bar{b}) \vee (\bar{a} \wedge b) : \bar{c} \succ c, c : d \succ \bar{d}, \bar{c} : \bar{d} \succ d$. Here, statement $a \succ \bar{a}$ represents the unconditional preference for $A=a$ over $A=\bar{a}$, while statement $c : d \succ \bar{d}$ states that $D=d$ is preferred to $D=\bar{d}$, given that $C=c$.

The semantics of CP-nets depends on the notion of a *worsening flip*. A *worsening flip* is a change in the value of a variable to a less preferred value according to the cp statement for that variable. For example, in the CP-net above, passing from $abcd$ to $ab\bar{c}d$ is a *worsening flip* since c is better than $\bar{c}$ given a and b . One outcome α is *better* than another outcome β (written $\alpha \succ \beta$) iff there is a chain of *worsening flips* from α to β . This definition induces a preorder over the outcomes, which is a partial order if the CP-net is acyclic.

In general, finding the optimal outcome of a CP-net is NP-hard. However, in acyclic CP-nets, there is only one optimal outcome and this can be found in linear time by sweeping through the CP-net, assigning the most preferred values in the preference tables. For instance, in the CP-net above, we would choose $A=a$ and $B=b$, then $C=c$, and then $D=d$.

3 Solution orderings and linearizations

Each of the constraint or preference-based formalisms recalled in the previous section generate a *solution ordering* over the variable assignments, where solutions dominate non-solutions, and more preferred solutions dominate less preferred ones. This solution ordering

can be a total order, a total order with ties, or even a partial order with ties. However, the problem of finding the next solution needs a strict linear order over the variable assignments, thus we will need to consider a linearization of the solution ordering.

CSPs generate a solution ordering which is total order with ties: all the solutions are in a tie (that is, they are equally preferred), and dominate in the ordering all the non-solutions, which again are in a tie. In soft constraints, the solution ordering is in general a partial order with ties: some assignments are equally preferred, others are incomparable, and others dominate each other. If we consider fuzzy or weighted CSPs, there can be no incomparability (since the set of preference values is totally ordered), so again we have a total order with ties, and a solution dominates another one if its preference value is higher. In this context, linearizing the solution ordering just means giving an order over the elements in each tie.

In acyclic CP-nets, the solution ordering is a partial order. In this scenario, any linearization of the solution ordering has to order every pair of incomparable assignments.

In the following, given a problem P and a linearization l of its solution ordering, we will denote with $\text{Next}(P, s, l)$ the problem of finding the solution just after s in the linearization l . Note that, while there is only one solution ordering for a problem P , there may be several linearizations of such a solution ordering.

It is not tractable to compute l explicitly, since it has an exponential length and it would mean knowing all the solutions and their relative order. For these reasons, we will assume the linearization is implicitly given to the Next procedure. For example, a lexicographic order on the variable assignments induces a linearization of the solution ordering of a problem, yet it is polynomially describable.

4 Finding the next solution in CSPs

Let P be a CSP with n variables, and let us consider any variable ordering $o = (x_1, \dots, x_n)$ and any value orderings $o_1, \dots, o_n$, where o_i is an ordering over the values in the domain of variable x_i . We will denote with O the set of orderings $\{o, o_1, \dots, o_n\}$. These orderings naturally induce a lexicographical linearization of the solution ordering, that we call $\text{lex}(O)$, where, given two variable assignments, say s and s' , we write $s \prec_{\text{lex}(O)} s'$ (that is, s precedes s') if either s is a solution and s' is not, or s precedes s' in the lexicographic order induced by O (that is, $s = (s_1, \dots, s_n)$, $s' = (s'_1, \dots, s'_n)$, and there exists $i \in [1, n]$ such that $s_i \prec_{o_i} s'_i$ and $s_j = s'_j$ for all $j < i$). We will now show that, if we take the linearization given by $\text{lex}(O)$, the problem of finding the next solution is NP-hard.

Theorem 1 *Computing $\text{Next}(P, s, \text{lex}(O))$, where P is a CSP, s is one of its solutions and O is a set of orderings, is NP-hard.*

The proof is based on a reduction from SAT.

This result can be extended to a wider class of orderings, as the following theorem states.

Theorem 2 *For each polynomially describable total order ω over complete variable assignments such that its top element does not depend on the constraints of the CSP and is polynomially computable, let us consider the linearization of the solution ordering induced by ω , say $l(\omega)$. Then there exists a solution s such that computing $\text{Next}(p, s, l(\omega))$, where p is a CSP, is NP-hard.*

5 Next on tree-shaped CSPs

We know that finding an optimal solution becomes easy if we restrict the constraint graph of the problem to have the shape of a tree. It is

therefore natural to consider this class to see whether also the Next problem becomes easy under this condition. We will see that this is indeed so: if the CSP is tree-shaped, it can be easy to find the next solution.

In this section we focus on tree-shaped CSPs. However, the same results hold for bounded tree-width. For a tree-shaped CSP with variable set $X = \{x_1, \dots, x_n\}$, let us consider the linearization $tlex(O)$, which is the same as $lex(O)$ defined in the previous section, with the restriction that the variable ordering o respects the tree shape: each nodes comes before its children. For example, let us consider the tree-shaped CSP shown in Figure 1, and assume that $o = (x_1, x_2, x_3, x_4, x_5)$ and that in all domains $a \prec_{o_i} b \prec_{o_i} c$. The solutions of the CSP are then ordered by $tlex(O)$ as follows: $(a, b, a, b, b) \prec (a, b, a, c, b) \prec (b, a, b, a, a) \prec (b, a, b, a, b) \prec (b, a, b, c, a) \prec (b, a, b, c, b) \prec (b, b, b, b, b) \prec (b, b, b, c, b)$.

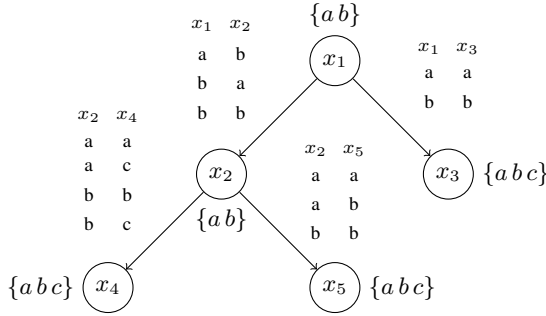


Figure 1. A tree-shaped CSP.

We will now describe an algorithm that, given as input a directionally arc consistent tree-shaped CSP P and a solution s for P , it either returns the consistent assignment following s according to $tlex(O)$, or it detects that s is the last consistent assignment in such an ordering. The algorithm works bottom-up in the tree, looking for new

Algorithm 1: CSP-Next

Input: tree-shaped and DAC CSP P , orderings $o, o_1, \dots, o_n$, assignment s
Output: an assignment s' , or “no more solutions”
for $i=n$ **to** 1 **do**
 Search $D(x_i)$ for the next value w.r.t. o_i which is consistent with $s_{f(i)}$, say v' ;
 if v' **exists then**
 $s_i \leftarrow v'$
 Reset-succ(s, i)
 return s
return “no more solutions”

variable values that are consistent with the value assigned to their father (denoted by $f(i)$ in Algorithm 1) and successive to the ones assigned in s in the domain orderings. As soon as it finds a variable for which such a value exists, it resets all the following variables (according to the variable ordering o) to their smallest compatible values w.r.t. the domain orderings (via procedure Reset-succ).

For example, if we run CSP-Next giving in input the CSP of Figure 1 and solution $s=(b, a, b, a, b)$, the algorithm first tries to find a value for x_5 consistent with $x_2 = a$ and following b in the domain ordering of x_5 . Since no such value exists, it moves to x_4 and performs a similar search, that yields $x_4 = c$. Procedure Reset-succ then sets $x_5 = a$, the first value in the ordering for x_5 consistent with $x_2 = a$.

Theorem 3 Consider a tree-shaped and DAC CSP P and the ordering $tlex(O)$ defined above. If s is not the last solution in ordering $tlex(O)$, the output of $CSP\text{-}next(P, s)$ is the successor of s according to $tlex(O)$; otherwise, the output of $CSP\text{-}next(P, s)$ is “no more solutions”.

If $|D|$ is the cardinality of the largest domain, it is easy to see that the worst case complexity of CSP-next is $O(n|D|)$, since both looking for consistent assignments and resetting to the earliest consistent assignment takes $O(|D|)$, and such operations are done $O(n)$ times.

From Theorem 3 we can thus conclude that $Next(P, s, tlex(O))$ is polynomial, since it can be computed by applying DAC to P and then CSP-Next to P and s , both of which are polynomial-time algorithms.

Note also that the choice of the linearization is crucial for the complexity of the algorithm. Indeed, a different choice for l may turn $Next(P, s, l)$ into an NP-hard problem, even on tree-shaped CSP, as proved in the following theorem.

Theorem 4 Computing $Next(P, s, l)$, where P is a tree-shaped CSP, s is one of its solutions, and l is an arbitrary linearization, is NP-hard.

The proof is based on a reduction from the subset sum problem.

6 Next on weighted CSPs

Theorem 5 Computing $Next(P, s, l)$, where P is a weighted CSP and s is one of its solutions, is NP-hard, for any linearization l .

The proof of Theorem 4 can be easily adapted for the purpose of this statement.

Note that theorems 4 and 5, while very similar in proof, have quite a different implication. Indeed, while for tree-shaped CSPs computing Next is NP-hard only for some choices of the linearization l , for weighted CSPs computing Next is *always* NP-hard, irrespective of the linearization.

However, we will now show that, if we consider the lexicographical ordering, then this NP-hardness is weak, since we can provide a pseudo-polynomial algorithm to find the next solution in weighted CSPs. Thus, it is only the possibility to use large utilities that makes the problem intractable in general.

In the context of a weighted CSP, finding the next solution means that, given a solution, we want to return the next assignment in lexicographical order with the same utility or, if there is no such assignment, the first assignment in lexicographical order with the next smaller utility.

Theorem 6 It is weakly NP-hard to compute the next solution in a weighted CSP.

To show that finding the next solution is only weakly NP-hard, we give a pseudo-polynomial algorithm which is a simple generalization of the dynamic programming algorithm for deciding subset sum [9, 11]. Given a weighted CSP with no constraints, and an assignment $a = (a_1, \dots, a_n)$ to its variables x_i with $1 \leq i \leq n$, with utility $U(a) = s$, we define $Q_a(j, t)$ as the lexicographically smaller assignment $b = (b_1, \dots, b_j)$ with utility t , that involves only the first j variables, and such that $b \succ_{lex} a$. If no such assignment exists, then $Q_a(j, t) = nil$. A simple recursion can be used to compute $Q_a(j, t)$, for any j from 1 to n , and any utility level t from 0 to s . In particular, we can initialize $Q_a(1, t)$ as $Q_a(1, t) = \text{lexmin}_{b_1 \in D_1} \{(b_1) : u_1(b_1) = t \wedge b_1 \succ_{lex} a_1\}$ and then compute recursively $Q_a(j, t) = \text{lexmin}_{b_j \in D_j} \{(Q_a(j-1, t-v_j), b_j) :$

$(b_1, \dots, b_j) \succ_{lex} (a_1, \dots, a_j) \cup \{(a_1, \dots, a_{j-1}, b_j) : u_j(b_j) = u_j(a_j) \wedge b_j \succ_{lex} a_j\}$ where $v_j = u_j(b_j)$. To compute the next solution given an assignment a with utility s , we just need to check whether $Q_a(n, s)$ is not *nil*. If it is, $Q_a(n, s)$ contains the lexicographically next assignment with utility s . Otherwise, we use a very similar dynamic program to compute the lexicographically smallest assignment with the next smallest utility. The running time of both dynamic programs is $O(n \cdot s \cdot |D|)$, where $|D|$ is the size of largest domain. Thus we can compute the next solution in pseudo-polynomial time.

This algorithm works also when the weighted CSP problem has some constraints, provided that there is a polynomial number of such constraints and each has a bounded scope. In such weighted CSPs, we compute $Q_a(j, t)$ from $Q_a(j-1, t - u_i)$ where u_i is the additional utility added by assigning $x_j = b_j$. This is polynomial to compute given our assumption that there is a polynomial number of constraints, each involving a polynomial number of variables. Hence, there is a pseudo-polynomial time algorithm to compute the next solution also for any such weighted CSP.

Example: To make it simple, we consider a weighted CSP with no constraints and five Boolean variables x_1 to x_5 where $u_i(0) = 0$ and $u_i(1) = i$. Suppose we are given the solution $a = (0, 1, 1, 0, 0)$ of utility $s = 5$ and we want to compute the lexicographically next solution. The dynamic programming algorithm will build from the bottom-left corner the following table:

t	x_1	x_2	x_3	x_4	x_5
5	<i>nil</i>	<i>nil</i>	<i>nil</i>	(1,0,0,1)	(1,0,0,1,0)
4	<i>nil</i>	<i>nil</i>	(1,0,1)	(1,0,1,0)	(1,0,1,0,0)
3	<i>nil</i>	(1,1)	(1,1,0)	(1,1,0,0)	(1,1,0,0,0)
2	<i>nil</i>	<i>nil</i>	<i>nil</i>	<i>nil</i>	<i>nil</i>
1	(1)	(1,0)	(1,0,0)	(1,0,0,0)	(1,0,0,0,0)
0	<i>nil</i>	<i>nil</i>	<i>nil</i>	<i>nil</i>	<i>nil</i>

For example, $Q_a(j=3, t=4) = (1, 0, 1)$. This means that $(1, 0, 1)$ is the lexicographically smaller assignment with utility 4, that involves only the first 3 variables, and such that it follows $(0, 1, 1)$ (that is, the restriction of a to the first 3 variables) lexicographically. Since $Q_a(j=5, t=5)$ is not *nil*, then the lexicographically next assignment with utility 5 is $(1, 0, 0, 1, 0)$.

7 Next on tree-shaped fuzzy CSPs

Turning our attention to fuzzy CSPs, we will show that Next on tree-like fuzzy CSPs can be easy. Let P be a fuzzy tree-shaped CSP with variable set $X = \{x_1, \dots, x_n\}$ and set of constraints C , and let us consider a variable ordering $o = \{x_1, \dots, x_n\}$ which respects the tree shape. Moreover, let o_i be a total order over the values in the domain of x_i , for $i = 1, \dots, n$.

We will consider set $T = \{t = (x_i = v_i, x_j = v_j) | i < j, \exists c \in C, t \in c, \text{pref}_c(t) > 0\}$, where $\text{pref}_c(t)$ denotes the preference assigned to t by constraint c that is, the set of all pairs of variables assignments appearing in P with preference greater than 0. The preferences assigned to tuples by the constraints in C and the orderings $o, o_i, \dots, o_n$ induce the following ordering o_T over T : $(x_i = v, x_j = w) \prec_{o_T} (x_h = z, x_k = u)$ if

- the preference associated to tuple $(x_i = v, x_j = w)$ by its constraint is higher than the preference associated to tuple $(x_h = z, x_k = u)$ by its constraint, or
- they have the same preference, and the variable pair (x_i, x_j) lexicographically precedes the variable pair (x_h, x_k) according to o , or

- they have the same preference, $i = h, j = k$ and the value pair (v, w) lexicographically precedes the value pair (z, u) according to domain orderings o_i and o_j .

We will now use this strict total order over the set of tuples of P to define a strict total order over the set of solutions of P . Given two complete assignments to X , say s and s' , let $t_s = \min_{o_T} \{t \text{ tuple of } s \text{ with preference } \text{pref}(s)\}$ and $t_{s'} = \min_{o_T} \{t \text{ tuple of } s' \text{ with preference } \text{pref}(s')\}$. Let $\text{opt}(P)$ denote the optimal preference of a fuzzy CSP P . We write $s \prec_f s'$ (that is, s precedes s' in ordering $\prec_f$), if

- $\text{pref}(s) > \text{pref}(s')$, or
- $\text{pref}(s) = \text{pref}(s') = \text{opt}(P)$ and s precedes s' in the lexicographic order induced by o and the domain orderings $o_1, \dots, o_n$, or
- $\text{pref}(s) = \text{pref}(s') < \text{opt}(P)$ and $t_s \prec_{o_T} t_{s'}$,
- $\text{pref}(s) = \text{pref}(s') < \text{opt}(P)$, $t_s = t_{s'}$ and s precedes s' in the lexicographic order induced by o and the domain orderings $o_1, \dots, o_n$.

It is possible to show that $\prec_f$ is a linearization of the solution ordering.

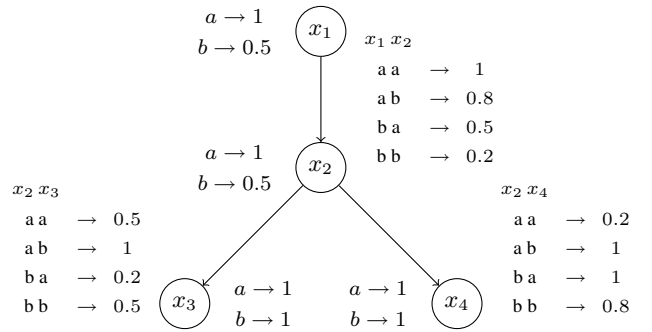


Figure 2. A tree-shaped DAC fuzzy CSP.

For example, let us consider the tree-shaped DAC Fuzzy CSP shown in Figure 2. Assume that $o = (x_1, x_2, x_3, x_4)$ and that $a \prec_{o_i} b$ for $i = 1, \dots, 4$. Then, if we consider solutions $s = (b, a, a, b)$ and $s' = (a, b, b, b)$, we have that $s \prec_f s'$ since $\text{pref}(s) = \text{pref}(s') = 0.5 < 1 = \text{opt}(P)$, $t_s = (x_1 = b, x_2 = a)$, $t_{s'} = (x_2 = b, x_3 = b)$, and thus $t_s \prec_{o_T} t_{s'}$. If instead we consider solutions $s = (b, b, a, a)$ and $s' = (b, b, a, b)$, we have again that $s \prec_f s'$, since $\text{pref}(s) = \text{pref}(s') = 0.2 < 1 = \text{opt}(P)$, $t_s = (x_1 = b, x_2 = b) = t_{s'}$, and s precedes s' lexicographically.

As with CSPs, we provide a polynomial time algorithm that solves the Next problem for tree-shaped fuzzy CSPs. The main idea that we exploit is that, in a fuzzy CSP, a solution can have preference p only if it includes a tuple that has preference p . In Algorithm 2:

- procedure $\text{fix}(P, t)$ takes in input a fuzzy CSP P and one of its tuples, $t = (x_i = v, x_j = w)$ and returns the fuzzy CSP obtained from P by removing from the domains of variables x_i and x_j all values except v and w ;
- procedure $\text{cut}(P, p)$ takes in input a fuzzy CSP P and a preference p and returns the CSP corresponding to the p -cut of P as defined in Section 2;
- procedure $\text{Solve}(P)$ takes in input a CSP P and returns the first solution in a lexicographic order given the variable and the domain orderings.

Algorithm 2: FuzzyCSP-Next

Input: tree-shaped and DAC Fuzzy CSP P , orderings $o, o_1, \dots, o_n, o_T$, assignment s with preference p
Output: an assignment s' , or “no more solutions”

if $p = \text{opt}(P)$ **then**
 $P' \leftarrow \text{cut}(P, p)$
 if $\text{CSP-next}(P', s) \neq \text{“no more solutions”}$ **then**
 return $\text{CSP-next}(P', s)$

if $p \neq \text{opt}(P)$ **then**
 compute tuple t_s
 $t^* = t_s$

else
 let t^* be the first tuple s.t. $\text{pref}(t^*) = \text{next}(p)$
 $p^* = \text{pref}(t^*)$
 $P' \leftarrow \text{cut}(\text{fix}(P, t^*), p^*)$
 if $\text{CSP-next}(P', s) \neq \text{“no more solutions”}$ **then**
 return $\text{CSP-next}(P', s)$

$\text{pref}(t) \leftarrow 0, \forall t \in T$ such that $\text{pref}(t) = p^*$ and $t \leq_{o_T} t^*$
 $\text{cpref} \leftarrow p^*$
for each tuple $t >_{o_T} t^*$ **following order** o_T **until** $\text{pref}(t) > 0$ **do**
 if $\text{pref}(t) < \text{cpref}$ **then**
 reset all preferences, previously set to 0, to their original values
 if $\text{Solve}(\text{cut}(\text{fix}(P, t), \text{pref}(t))) = \text{pref}(t)$ **then**
 return $\text{Solve}(\text{cut}(\text{fix}(P, t), \text{pref}(t)))$
 $\text{cpref} \leftarrow \text{pref}(t)$
 $\text{pref}(t) \leftarrow 0$

return “no more solutions”

Intuitively, when solution s with preference p is given in input, if s is optimal, we look for the next solution in the CSP obtained from P by performing a cut at level p and running CSP-next. If no solution is returned, then s must have been the last solution with optimal preference in the ordering and its successor must be sought for at lower preference levels.

If s is not optimal, we consider its tuples and we identify the smallest tuple of s , say t_s , according to ordering o_T , that has preference p in the corresponding constraint. We fix such a tuple, via $\text{fix}(P, t_s)$, and we cut the obtained fuzzy CSP at level p . We then look for the solution lexicographically following s in such a CSP by calling CSP-next. If no such solution exists, s must be the last solution with preference p among those that get their preference from t_s .

The next solution may have preference p or lower. However, if it does have preference p , such a preference must come from a tuple with preference p which follows t_s in the ordering o_T . In order to avoid finding solutions with preference equal to p that come from tuples with preference p preceding t_s according to o_T , we set the preference of all such tuples to 0. If none of the tuples with preference p following t_s generate solutions with preference p , we move down one preference level, restoring all modified preference values to their original values. This search continues until a solution is found or all tuples with preference greater than 0 have been considered.

Theorem 7 *Given a tree-shaped DAC fuzzy CSP P and a solution s , algorithm FuzzyCSP-Next computes the successor of s according to $\prec_f$ if s is not the last solution with preference greater than 0, and outputs “no more solutions” otherwise. The worst case time complexity of algorithm FuzzyCSP-Next is $O(|T||D|n)$, where $|T|$ is the number of tuples of P and $|D|$ the cardinality of the largest domain.*

The correctness of FuzzyCSP-Next follows directly from the description of the algorithm. For the complexity, we notice that the

complexity of FuzzyCSP-Next is bounded by that of running $|T|$ times the CSP-next algorithm. $\square$

Given a tree-shaped fuzzy CSP P , one of its solutions s , and the solution ordering $\prec_f$, $\text{Next}(P, s, \prec_f)$ can be therefore computed in polynomial time: we just need to achieve DAC (which is polynomial) and then run algorithm FuzzyCSP-Next. Again, it is not difficult to prove that the choice of the order is crucial for the complexity of the algorithm, and that $\text{Next}(P, s, l)$ is in general NP-hard even on tree-shaped fuzzy CSPs. Indeed, since tree-shaped fuzzy CSPs admit tree-shaped CSPs as a special case, the result is a direct consequence of Theorem 4.

8 Next on acyclic CP-nets

We now consider the complexity of the Next operation in acyclic CP-nets. It turns out that Next is easy on such CP-nets, if we consider a certain linearization of the solution ordering. We first define the concept of *contextual lexicographical linearization* of the solution ordering. Let us consider any ordering of the variables where, for any variable, its parents are preceding it in the ordering. Let us also consider an arbitrary total ordering of the elements in the variable domains. For sake of simplicity, let us consider Boolean domains. Given an acyclic CP-net with n variables, we can associate a Boolean vector of length n to each complete assignment, where element in position i corresponds to variable i (in the variable ordering), and it is a 0 if this variable has its most preferred value, given the values of the parents, and 1 otherwise. Therefore, for example, the optimal solution will correspond to a vector of n zeros.

To compute such a vector from a complete assignment, we just need to read the variable values in the variable ordering, and for each variable we need to check if its value is the most preferred or not, considering the assignment of its parents. This is polynomial if the number of parents of all variables is bounded. Given a vector, it is also easy to compute the corresponding assignment.

Let us now consider a linearization of the ordering of the solutions where incomparability is linearized by a lexicographical ordering over the vectors associated to the assignments. We will call such a linearization a *contextual lexicographical linearization*. Note that there is at least one of such linearizations for every acyclic CP-net.

Theorem 8 *Computing $\text{Next}(N, s, l)$, where N is an acyclic CP-net, s is one of its solutions, and l is any contextual lexicographical linearization of its solution ordering, is in P .*

Given any solution s and its associated vector, as defined above, the vector of the next solution in l can be easily obtained by a standard Boolean vector increment operation. Therefore, given any solution s , it is possible to obtain the next solution by 1) computing the vector associated to s , 2) incrementing it, and 3) computing the solution associated to the new vector. Since each of these steps is polynomial, the overall process is polynomial. The same proof can be easily extended to non-Boolean domains. $\square$

Figure 3 shows an acyclic CP-net, with features A, B, and C, and its solution ordering. It is assumed that the variables have each two values: for A we have a and $\bar{a}$, and similarly for B and C. Also, the variable ordering is $A \prec B \prec C$. Given solution abc (that is, $A=a, B=b, C=c$), the associated Boolean vector (as described above) is 000, since a is the most preferred value for A, b is the most preferred value for B given $A=a$, and c is the most preferred value for C. Instead, the vector associated to solution $\bar{a}\bar{b}c$ is 100, and the vector associated to $\bar{a}b\bar{c}$ is 111. Given vector 101, the associated solution is $\bar{a}b\bar{c}$. In Figure 3 it is possible to see the CP-net, the solution ordering, and the vector for each solution. Also, if we order the solutions

according to a standard lexicographical order over their vectors, we get a linearization of the partial solution ordering.

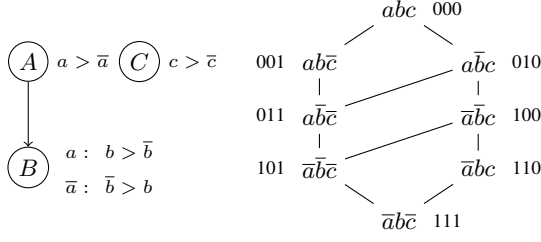


Figure 3. An acyclic CP-net and its solution ordering.

9 Next on constrained CP-nets

Some statements are better expressed via constraints, other via preferences. Moreover, some preferences are better modeled via soft CSPs, other via CP-nets. However, usually in a real-life problem we may have statements of all these kinds, thus requiring to use all the above considered formalisms in the same problem. It is therefore useful to consider problems where CP-nets and CSPs, or soft CSPs, coexist [3].

We thus consider here the notion of a *constrained CP-net*, which is just a CP-net plus some (soft) constraints [3]. Given a CP-net N and a constraint problem P , we will write (N, P) to denote the constrained CP-net given by N and P . For sake of simplicity, in the following we will assume that the CP-net and the CSP involve the same variables. Nevertheless, our results hold also for the more general setting.

Given a constrained CP-net (N, P) , its solution ordering, written $\prec_{np}$, is that given by the (soft) constraints, where ties can be broken by the CP-net preferences. More precisely, solution s dominates solution s' (that is, $s \prec_{np} s'$) if s dominates s' according to the constraints in P , or s and s' are equally preferred according to the constraints in P , but s dominates s' according to the CP-net N .

We now ask consider the complexity of computing the next solution in a linearization of this ordering. The first results says that the problem is difficult if we take the lexicographical linearization (given o , which is an ordering over the variables) of $\prec_{np}$, denoted with $lex(o, \prec_{np})$.

Theorem 9 Computing $Next((N, P), s, lex(o, \prec_{np}))$, where (N, P) is a constrained CP-net and s is one of its solutions, is NP-hard.

The statement can be proven by reducing Next on a CSP to Next on a constrained CP-net. The same proof applies also to constrained CP-nets where the CP-net is acyclic.

Next becomes easy if we consider acyclic CP-nets, tree-shaped CSPs, and we add a compatibility condition between the acyclic CP-net and the constraints. This compatibility condition is related to the topology of the CP-net dependency graph and of the constraint graph.

Consider two variables in an acyclic CP-net, say x and y . We say that x depends on y if there is a dependency path from y to x in the acyclic DAG of the CP-net. Given an acyclic CP-net N and a tree-shaped CSP P , we say that N and P are *compatible* if there exists a variable of the CSP, say r , such that: for any two variables x and y such that x is the father of y in the r -rooted tree, we have that x does not depend on y in the CP-net. Informally, this means that it is possible to take a tree of the constraints where the top-down father-child links, together with the CP-net dependency structure, do not create cycles. If the compatibility holds for any root taken from a set S , then we will write that N and P are S -compatible.

Figure 4 shows an example of a CP-net DAG and two trees, of which the one in Fig. 4 (b) is compatible with the CP-net: if we choose A as the root, the father-child relationship is not contradicted by the CP-net dependencies. Instead, the tree in Fig. 4 (c) is incompatible with the CP-net: whatever root is chosen, some tree links are contradicted by the CP-net dependencies.

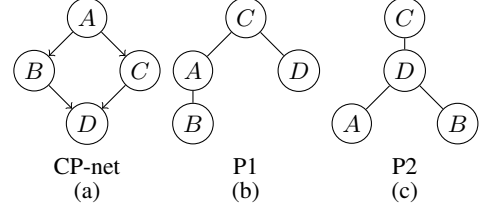


Figure 4. A CP-net dependency graph and two trees.

Theorem 10 Consider an acyclic CP-net N and a tree-shaped CSP P , and assume that N and P are S -compatible, where S is a subset of the variables of P . Taken a solution s for (N, P) , and a variable ordering o which respects the tree shape of P with root an element of S , we have that $Next((N, P), s, lex(o, \prec_{np}))$ is in P .

To compute $Next((N, P), s, lex(o, \prec_{np}))$, we use algorithm CSP-Next, except that we dynamically order each variable domains according the CP-net: for any variable, we order its domain according to the row of its CP table associated to the fixed assignment to the parent variables.

Under these same conditions, Next remains easy even if we consider CP-nets constrained by fuzzy CSPs rather than hard CSPs. We just need to adapt in a similar way algorithm FuzzyCSP-Next.

ACKNOWLEDGEMENTS

Research partially supported by the Italian MIUR PRIN project 20089M932N: “Innovative and multi-disciplinary approaches for constraint and preference reasoning”.

REFERENCES

- [1] Bistarelli, Montanari, and Rossi, ‘Semiring-based constraint satisfaction and optimization’, *Journal of the ACM*, **44**, 201–236, (1997).
- [2] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole, ‘CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements’, *Journal of Artificial Intelligence Research*, **21**, 135–191, (2004).
- [3] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole, ‘Preference-based constrained optimization with CP-nets’, *Computational Intelligence*, **20**(2), 137–157, (2004).
- [4] R. Brafman, F. Rossi, D. Salvagnin, B. Venable, and T. Walsh, ‘Finding the next solution in constraint- and preference-based knowledge representation formalisms’, in *Proc. KR 2010*, (2010).
- [5] A. A. Bulatov, V. Dalmau, M. Grohe, and D. Marx, ‘Enumerating homomorphisms’, in *Proc. STACS 2009*, (2009).
- [6] R. Dechter, ‘Tractable structures for CSPs’, in *Handbook of Constraint Programming*, eds., P. Van Beek F. Rossi and T. Walsh, Elsevier, (2005).
- [7] Rina Dechter, *Constraint Processing*, Morgan Kaufmann, 2003.
- [8] Terence Kelly and Andrew Bayde, ‘Generating k-best solutions to auction winner determination problems’, *SIGecom Exch.*, **6**(1), 23–34, (2006).
- [9] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, 1990.
- [10] P. Meseguer, F. Rossi, and T. Schiex, ‘Soft constraints’, in *Handbook of Constraint Programming*, eds., P. Van Beek F. Rossi and T. Walsh, Elsevier, (2005).
- [11] D. Pisinger, ‘Linear time algorithms for knapsack problems with bounded weights’, *ALGORITHMS: Journal of Algorithms*, **33**, (1999).

Factors Influencing User Motivation for Giving Online Preference Feedback

Joost Broekens and Alina Pommeranz and Pascal Wiggers and Catholijn M. Jonker¹

Abstract. Preference elicitation is important for any computerized system advising users about choices. Recommender systems aim to propose interesting material to users. Therefore, they must first gather user preferences. Negotiation support systems can only give meaningful bidding advice based on users' preferences regarding negotiable issues and interests. In general, the more detail users are willing to give the better the support will be. We investigated how four factors influence users' willingness to give detail in an online preference elicitation experiment. 18 users rated 60 items (pictures/songs) with 5 levels of detail (from 3-point scale over affective feedback to free text). For each item, users could choose the desired detail level. Our results show that, of the four factors investigated (having an opinion about an item, content type of the item, familiarity with, and ownership of that item), mainly having an opinion about an item makes users give significantly more detailed feedback. Further, users with an opinion about an item use qualitatively rich affective feedback in 30% of the cases. Our findings indicate that adaptive preference elicitation interfaces can conditionally hide and show fine grained feedback providing a simpler interface, which can be important for smaller interfaces. Further, the fact that 30% of the cases rated with an opinion included affective detail indicates that users are willing to give rich affective feedback when they have an opinion.

1. INTRODUCTION

Preference elicitation is an important aspect of any computerized system that is supposed to give personalized advice to users about choices of opinion or behavior. Preference elicitation is about extracting a user's opinion about items (e.g., music, pictures, negotiable issues, interests) in such a way that the extracted opinion actually reflects what the user thinks about that item. Common criteria to evaluate such feedback are reliability and validity. The feedback must be reliable: when the user is asked again for feedback on the same or a similar item, one can assume, all else being equal, that the feedback is similar to the earlier feedback. The feedback must be valid: the feedback as extracted and used by the computerized system should reflect the actual opinion of the user; if a user rates a piece of music 4 out of 5 stars, than that should mean the user likes the piece of content more than average but less than perfect, and other items rated 5 or 3 should be liked more and less respectively.

For example, in recommender systems, the aim is to build a user model for the purpose of proposing interesting and novel material to that user. To achieve this, the recommender system must first gather user preferences that reflect actual interest. There are two main

approaches: content-based recommendation and collaborative filtering [12]. The content-based approach goes as follows. The system presents a user with the opportunity to give feedback about an item in the form of a rating mechanism (usually stars or values). Based on this feedback (called content-based feedback, as the user directly rates the content), the system can calculate how much the user likes this content. Given a measure of content similarity (e.g., documents containing the same text) the system can then deduce what other content the user would like and recommend this to the user. The collaborative filtering approach roughly works as follows [13]. Subjects give feedback about items. Sometimes the feedback is implicit such as whether or not a user bought a particular item. Differing from the first approach, the system does not assume anything about items (the content) and can therefore not deduce which other items to propose to the user based on item similarity. Instead, an item-item similarity measure is deduced from a person-person similarity measure, e.g., by analyzing in how far two different users are alike with respect to the items they bought or viewed. In any case, whenever user feedback is used, preference elicitation becomes an issue because the system wants to capture actual interest (or likes and dislikes) to build a user model. For a recent review from the perspective of preference elicitation see [22].

The same holds for negotiation support [23]. If a negotiation support system is supposed to give personalized advice about bids that is meaningful and useful to the user, the system must first extract that user's preferences to build a user model representing the user's possible issues, interests and options involved in the negotiation. When the system knows how important these issues and preferences are to that user, it can start calculating bids that match the user and the opponent best in the negotiation; for example, bids that try to maximize a pareto optimum [16].

In this paper, we address a third issue with respect to preference elicitation, beside validity and reliability: the level of detail of the feedback. In general, the more detail a user is willing to give, the better the bidding support or recommendation will work (assuming it is also valid and reliable). However, one does not want to bother users with an endless interface measuring all kinds of opinions and ratings about items such as persons, songs, pictures, bids, or issues. User motivation is an important aspect to consider when developing systems that need user feedback. For example, when collecting user tags (labels users associate to content items), it is important to consider why users tag the content, for example, to portray themselves, to help categorize the content or to help others find content [1][18]. Ideally a system should motivate users to give feedback, for instance by making the feedback system socially relevant or fun to use [11]. In our work, we took a slightly different approach. We wanted to know how a user's default motivation for giving feedback varies un-

¹ Technical University Delft, The Netherlands, email: d.j.broekens, a.pommeranz, p.wiggers, c.m.jonker@tudelft.nl

der the influence of different variables in a relatively neutral setting. With neutral setting we mean that we did not specifically motivate people to give preference feedback by making the system fun, social or otherwise rewarding. It is important to know this default motivation regarding the detail level for at least three reasons. First, up until now there is no such thing as a baseline level of detail users are willing to give as feedback. From a research point of view this is important to have as it allows comparison of different methods to elicit user feedback. Second, if this default motivation varies depending on the user, the type of content or other variables, then it is worthwhile to adaptively give a user the opportunity to add more detail based on these variables. Doing so will increase the level of detail of the extracted preference and hence increase the amount of available information for subsequent user modeling, without bothering the user because the user's intrinsic motivation already triggers him or her to give more detail. Third, most preference feedback is given by means of 5-point Likert scales. Affective feedback is currently being investigated as an additional way of asking feedback. This is being tested in different domains, such as interactive television [14] and music annotation and retrieval [17]. Such studies show that adding affective information indeed can be beneficial to user models. The question is however, do users want to give additional levels of affective feedback, and how far do they go in giving detail?

We investigate the following hypothesis: the level of detail persons are willing to give in their feedback depends on content type, familiarity, ownership and opinion. To be more precise: when users are free to choose the level of detail they want to give about a particular piece of content, how far would they go in giving feedback, and does this depend on the content type, whether they own or are familiar with the content and whether they have an opinion about the content. We present results of an online experiment in which 18 users rated 60 items (30 pictures of well-known persons, 30 popular songs) with 5 different levels of detail: 3-point scale, 5-point likert scale, affective feedback using the *AffectButton* [3], emotion words and free text feedback. For each item, the user chose the desired level of detail. Our results show that the main factor influencing the amount of detail users are willing to give is whether or not they have an opinion about the item at all, while content type, familiarity with, and ownership of item did not have an important influence. Further, users with an opinion include affective (emotional) feedback in their feedback.

2 RELATED WORK

People's preferences have been the interest of researchers in many different fields. These include psychology, behavioral science, consumer research, e-commerce, recommender systems as well as negotiation and decision support. In the introduction to our work we mentioned the importance of preference elicitation for computerized support systems, such as recommender systems or negotiation support systems. In this section we dig deeper into related work in the area of preference elicitation interfaces. First, however, we give a short overview of studies from psychology and behavioral sciences that focus on how people come to have preferences.

2.1 Constructed Preferences

Carenini and Poole [6] describe two very influential conceptual shifts for classical decision theory: constructive preferences [21] and value-focused thinking [16], and their implications for preference elicitation. We will focus here on the first shift which has occurred in behavioral decision making. Many studies have confirmed that prefer-

ences are not stable but constructive. This means that people do not have well-defined preferences in most situations but rather construct them when necessary, i.e., in the decision making context. There are different views on how people construct their preferences. Simon et al. [29] for instance found in their experiments that while people processed the decision task, their preferences of attributes in the option that was chosen increased whereas those for attributes of rejected options decreased. Similar effects have been found in negotiation settings reported by [8]. It is also in line with Bettman and Luce's [2] idea of trying to maximize the ease of justifying a decision while making it. Another aspect of constructing preferences has been brought forward by [10] focusing on the goals of the decision task in relation to a so-called prominence effect. This effect occurs when people prefer an alternative that is superior only on the most prominent, i.e. the most important, attribute. They confirmed in three studies that the prominent attribute will be more heavily weighted when the goal was making a choice between alternatives than when the goal was to arrive at a matching value.

The research above implies that we have to think carefully about the way we pose a preference elicitation task to our users, in order to avoid unwanted effects. Another view on constructing preferences comes from [30] and states that people construct preferences from memory. The authors present the so-called PAM (preferences-as-memory) framework, which assumes that "decisions (or valuation judgments) are made by retrieving relevant knowledge (attitudes, attributes, previous preferences, episodes, or events) from memory in order to determine the best (or a good) action." They also emphasize that this is not an entirely cognitive view on preference construction since affect determines what the person recalls first. Information consistent with emotions is more available in memory. For our study this is interesting for two reasons: (1) if people construct preferences based on their memories, familiarity of an item might influence the detail of the given preference and (2) if emotional factors play a role in the construction of preferences, then how likely are people to use affective feedback methods.

Other psychological effects have been found when people construct their preferences. For example, anchoring effects and effects that occur when complicated numbers or information are presented in the choice task [15]. Different ways to measure preferences can lead to different results, which is usually not the intention of eliciting preferences. To help people to construct their preferences in health care scenarios, Johnson and colleagues [15] suggest to present default choices that have lead to the best outcome for most patients and present information in a way that helps the patient to understand the outcomes of each choice.

Consumer research looked at the interplay between affect and cognition on decision making [28]. They investigated the influence of available processing resources when confronted with a decision task. In cases people have only few resources available affective reactions tend to have a greater impact on choice, whereas with high availability of resources cognitions related to the consequences of the choice are more dominant. They noted that this finding is influenced also by personality and by the representation of the choice alternatives. Other interesting insights from this research area come from [31] looking at affective and cognitive factors in preferences. Among other things they report that mere exposure of a stimulus can positively influence a person's preference for this stimulus. Conscious recognition, however, was not the dominant factor since people were generally not aware that they had seen the same stimulus earlier.

Concluding this section, we can record that there are many factors influencing preference elicitation. Payne and coworkers [20] have

made an attempt to develop guidelines for measuring preferences taking people’s behavior into account. Similar work focusing on the user side has been presented in [24] and [25]. Our work is complementary in the sense that we study how four factors, item content type, familiarity with, opinion about, and item ownership influence the level of *preference detail users are willing to give*.

2.2 Preference Elicitation Interfaces

Most literature on preference elicitation interfaces focuses on the technical implementations and underlying formal structures instead of on the user input itself. Therefore, it is often not clear how a method facilitates the user to construct preferences. In addition, many systems are based on quantitative methods using utility functions that require special input and assume stable preferences. As the previous section shows, this view is debatable, since people construct their preferences and this construction has many influential factors that need to be carefully considered when designing an interface. We now review several preference elicitation interface approaches. For a recent more detailed review see [22].

Chen and Pu [7] give an overview over existing systems that elicit user preferences. They mention different techniques used in the systems, like knowledge-based find-me techniques [4], example critiquing and tweaking [9], active decisions and clustering or collaborative filtering techniques [26]. There are also hybrid systems combining different approaches [5]. In knowledge-based systems, preferences are elicited by example-similarity; the user rates a given item and requests similar items. Tweaking can be used to limit the similar items to only those satisfying the tweak. In example-critiquing approaches the user is presented with a set of candidates that can be critiqued. The user can either choose one of them or critique some of their attributes. An interesting interface has been developed by [27] called the Apt Decision Agent. In this system people initially provide a small number of criteria for an apartment. Based on those they get a number of sample apartments. They can react to any attributes of any apartment. Interesting here is that the preference feedback by the user gets more and more detailed during the interaction. At the same time the user is not *forced* to go into more detail, but is free to give only the feedback the user wants to give. A similar approach is called Active Decisions and used in many online shopping environments, where people go through a two stage process. First, they screen a large set of products from which they choose a promising subset. Second, they evaluate these in further detail.

As already mentioned, most approaches are focused on techniques to implement the preference elicitation process. Less attention has been given to concrete interface elements used in these approaches and in particular the question ‘how much detail a user wants to give using these interface elements’. Most approaches assume users that want to interact with the interface and give ratings or critique examples.

3 EXPERIMENTAL SETUP

As mentioned in the previous section, most research in preference elicitation pays little attention to the motivation that drives people to enter their preferences into a system and how much detail they are willing to give for each preference. We investigate the hypothesis that the level of detail persons are willing to give in their feedback depends on content type of the item, familiarity with the item, ownership of the item and opinion about the item. We have set up a content

rating experiment with content type, ownership and opinion as independent variables and detail as dependent variable. The content was preselected by the experimenters (see Materials and Procedure), and subjects could indicate their familiarity with the content, ownership of, and opinion about each content item. As we had no control over these three variables, we could not do a real 2x2x2 setup: some of the cells in the experiment design would have been (and indeed are) almost empty. As such we will consider the three variables as independent factors (we will not test for dependencies). For the influence of familiarity on detail a standard correlation study was performed, based on the same data, with familiarity as independent variable and level of detail as dependent variable.

Figure 1. Example of the interface for testing the level of detail. In this figure the levels are all shown on the left side of the screen. From top to bottom: 3-point, 5-point, AffectButton, emotion words and free text. On the right side the current stimulus and the fields to fill in familiarity and ownership. In this figure, all levels are shown, but during the experiment a subject sees a level only when he/she pressed the “DETAIL” button situated next to the “next song/pic” button (not shown here, because all levels have been filled in).

3.1 Participants

We selected participants from a broad range of people that indicated earlier that they would like to take part in experiments. We tested 18 participants, 12 male and 6 female between the age of 21 and 65 (avg=30, stdev=10). Participants have different cultural backgrounds as well as nationality and education.

3.2 Material and Procedure

Participants received an email with the invitation to participate including a link to the application needed for the study. The study was done online, so subjects did the experiment at a place of their own choice when they wanted. The participants were told that the experiment was about creating an alternative top-40 of famous people and popular music. The real goal of the experiment was kept secret, so that user motivation was not influenced by the instructions. The email contained detailed instructions about how to use the application. After the participants started the application, they were asked to fill in

some standard information (age, gender, and education). After filling in this information, they were presented with 30 songs and 30 pictures of famous people (one at the time, at random). For each picture/song they were asked to fill in their familiarity with the song or person (5-point scale) and whether or not they owned the song or media concerning the person (yes/no) (see right side of the window in Figure 1). Then they were asked to give their opinion about the picture/song. For each stimulus they had to give at least a thumbs-down/neutral/thumbs-up opinion (3-point scale). Neutral was interpreted as no opinion, thumbs-down and thumbs-up were interpreted as having an opinion. After that, they had the choice to enter more detail to their opinion (click on a button: DETAIL) or go to the next picture/song (click on a button: next song/pic). The position of the two buttons was randomly interchanged for each new content item to avoid a bias for clicking one or the other. There were 5 levels of detail and each level had to be filled in before the participant could go to the next to make sure the user takes an active decision in whether to give more feedback or not. However, at every level subjects could stop giving feedback and go to the next stimulus, except at the obligatory first level. With these levels we wanted to test (a) for more detail with respect to granularity (level 1 - level 2), and (b) more detail with respect to additional affective dimensions (level 2- level 3). Free labels (level 4) were added to add expressive power to level 3. Free text input (level 5) was added to give the user full expressive freedom. The levels are:

1. Thumbs-down/no opinion/thumbs-up. All subjects had to rate their opinion about each item using this input level.
2. A 5-point scale ranging from awful to great. This was a more detailed version of the first level, introducing 5 options: Awful, Bad, Neutral, Good, Great.
3. Affective feedback using the AffectButton, an interactive button that can be used to give affective (emotional) feedback based on three dimensions: pleasure, arousal and dominance [19]. It is a dynamically changing selectable emotion expression. The expression changes based on the position of the mouse in the button. Previous research has shown that the AffectButton is useable and produces reliable and valid affective feedback [3].
4. Emotion words that match the feedback filled-in with the AffectButton. Based on a subject's feedback for an item at level 3, five emotion words out of a total of 31 were presented that best matched the feedback. The complete list of words spanned the pleasure-arousal-dominance space. If a user entered a happy face, words such as happy, content, etc. would be presented in the option list.
5. Free text input. This option enables subjects to give free text input about an item as a last level of detail.

All data were collected at a central server including the actual level of detail (1-5) entered per item. We took each rated item as statistical unit of analysis resulting in a total of $18 \times 60 = 1080$ cases. Excel and StatistiXL were used to analyze the data.

4 RESULTS AND DISCUSSION

Before analyzing the data related to our hypotheses, we first did several checks to find out if the experiment went as intended. The correlation between sequence number of item presented and detail was not significant, indicating that there were no wear or boredom effects on the subjects during the course of the experiment. Correlations between level-1 (thumbs), level-2 (5-point) and the continuous pleasure scale of level-3 (AffectButton) were all high and significant (all $r > 0.8$, sign. < 0.001), as would be expected. This indicates subjects are consistent in their ratings and therefore we can assume

subjects participated seriously in the experiment.

4.1 Opinion influences amount of feedback

With regards to our hypotheses, we found a significant, but not very large, influence of content type and ownership on the level of detail subjects would enter for an item. Music is rated with more detail ($mean_{level\ of\ detail} = 2.0$) than pictures of famous people ($mean = 1.8$; $F(1, 1078) = 5.3$; $p = 0.02$). Items owned by a person are rated with more detail ($mean = 2.2$) than items not owned ($mean = 1.9$; $F(1, 1078) = 8.1$; $p < 0.01$). We found a significant and large influence of opinion on the level of detail entered. Items about which a person has an opinion are rated with much more detail ($mean = 2.1$) than items about which a person has no opinion ($mean = 1.4$; $F(1, 1078) = 65$; $p = 0$). In fact, in a backwards stepwise linear regression analysis (to check which of these factors has what responsibility for the variation in the level of detail), only opinion came out as a significant model for predicting the level of detail. These results tell us that people who have an opinion about something feel the need to give more detailed feedback about that item. This influence has been found for both content types, and there was only a small interaction effect between content type and opinion (2×2 ANOVA, $F(1) = 4.5$; $p = 0.03$). The regression and ANOVA analyses indicate that, for practical purposes, of the four factors investigated, the user's opinion using a 3-point feedback scale is the important factor that influences the amount of feedback given.

With regards to the effects of familiarity we found an interesting pattern that depends on opinion. Familiarity was correlated positively with level of detail when taking all cases into account ($n = 1080$; $p < 0.001$; $r = 0.136$); however, the correlation is rather small. When we repeated the analysis with only those cases where a subject used at least the 5-point scale (level 2) the correlation disappeared. However, when we again repeated the analysis but now with only the cases where a subject used at least the affective feedback (level 3), the correlation was negative and much stronger ($n = 262$; $p < 0.001$; $r = -0.353$). This indicates a nonlinear relationship between familiarity and level of detail. It seems that in general familiarity with the item does not play a major role in the level of detail one wants to give as feedback, unless one has an opinion about the item and one is unfamiliar with the item. In that case, subjects used more detail to express their opinion.

4.2 Affective feedback used to express opinion

To gain a little more insight into the distribution of the use of feedback detail, consider Figure 2. Of all cases rated as "no-opinion", only about 22% will subsequently be scored using more detail than a three point thumbs-up/no opinion/thumbs down scale. Only in about 11% of those cases affective feedback is used. In contrast, if a case is rated as "thumbs-up" or "thumbs-down", these figures change dramatically. Now, 48% of the cases are rated using a 5-point scale and 31% of the items are rated using affective feedback.

4.3 Discussion of relevance

Our results clearly show that the amount of feedback detail, including affective feedback, strongly depends on whether users have an opinion. This is in itself not very surprising, but it is relevant for several reasons. *First*, the results show that an adaptive preference elicitation interface can take into account course grained feedback for predicting if fine grained feedback is needed. This means that the

interface can conditionally hide and show the fine grained feedback providing a simpler interface. This can be particularly important for smaller interfaces, such as those used for mobile devices. *Second*, the results show that users are willing to give affective feedback when they have an opinion. Affective feedback is qualitatively richer data than standard one-dimensional feedback and is therefore important to consider in preference elicitation interfaces. *Third*, our results give a first baseline of the amount of feedback that can be expected given that no effort has been done to motivate the user to give feedback (e.g., using social computing, added functionality). As we did not prime subjects to give any level of detail (for each item a user explicitly indicated by pushing a button the need for the next level), and as the type of content only marginally influences the amount of feedback given, our results seem valid as "baseline motivation". *Fourth*, future work is needed to investigate the exact influence of familiarity on the amount of detail given, as the influence of this factor seems to be dependent on opinion.

Of course our findings are limited. They relate to online, web-based, preference elicitation, and our experiment should be repeated in other domains, with larger groups of users, and different rating tasks, goals and contexts.

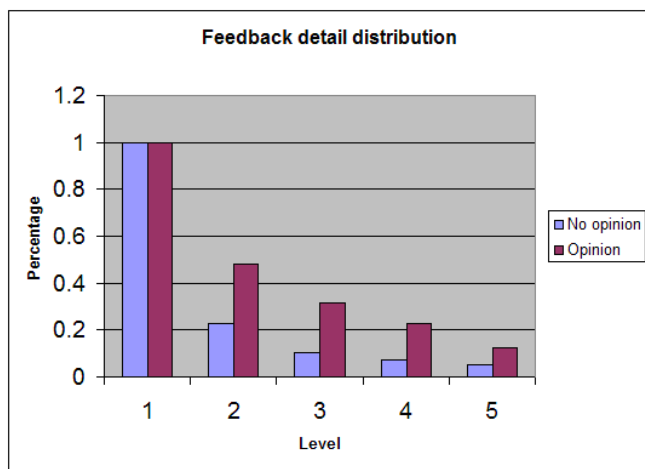


Figure 2. The figure shows the distribution of the levels of feedback given for all cases. Level 1 was always given (see experiment setup). For example, 48% of the people used the 5-point Likert scale when they had an opinion, while only 22% used it when they did not have an opinion.

5 CONCLUSIONS

We have argued that it is important to consider the level of detail a user is willing to give when asked for feedback about a content item. We have done a web-based preference elicitation experiment testing the influence of four factors on the amount of detail given. The four factors investigated are type of content, familiarity with the content, ownership of the content and, whether or not a subject has an opinion about the content. Opinion showed to be the most important factor. Subjects with an opinion about an item gave significantly more detail than when an opinion was absent. This provides opportunities for adaptive preference elicitation interfaces to conditionally hide and show fine grained feedback providing a simpler interface, which can be important for smaller interfaces. Further, 30% of the cases rated with an opinion included affective detail, indicating users' willingness to give rich affective feedback when they have an opinion.

ACKNOWLEDGEMENTS

We would like to thank our participants. This research is supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs, grant number VICI-project 08075.

REFERENCES

- [1] Morgan Ames and Mor Naaman, 'Why we tag: motivations for annotation in mobile and online media', in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 971–980, San Jose, California, USA, (2007). ACM.
- [2] James R. Bettman, Mary F. Luce, and John W. Payne, 'Constructive consumer choice processes', *Journal of Consumer Research*, **25**(3), 187–217, (1998).
- [3] Joost Broekens and Willem-Paul Brinkman, 'Affectbutton: Towards a standard for dynamic affective user feedback', in *ACII 2009*. IEEE, (2009).
- [4] R. Burke, 'Knowledge-based recommender systems', In A. Kent (ed.), *Encyclopedia of Library and Information Systems*, **69**(Supplement 32), (2000).
- [5] R. Burke, 'Hybrid recommender systems: Survey and experiments', *User Modeling and User-Adapted Interaction*, **12**(4), 331–370, (2002).
- [6] G. Carenini and D. Poole, 'Constructed preferences and value-focused thinking: Implications for ai research on preference elicitation', in *AAAI-02 Workshop on Preferences in AI and CP: symbolic approaches*, Edmonton, Canada, (2002).
- [7] L. Chen and P. Pu, 'Survey of preference elicitation methods', Technical report, Swiss Federal Institute of Technology in Lausanne (EPFL), (2004).
- [8] Jared R. Curhan, Margaret A. Neale, and Lee Ross, 'Dynamic valuation: Preference changes in the context of face-to-face negotiation', *Journal of Experimental Social Psychology*, **40**(2), 142–151, (2004).
- [9] B. Faltings, P. Pu, and M. Torrens, 'Designing example-critiquing interaction', in *IUI*, Portugal, (2004). ACM.
- [10] Gregory W. Fischer, Ziv Carmon, Dan Ariely, and Gal Zauberman, 'Goal-based construction of preferences: Task goals and the prominence effect', *Management Science*, **45**(8), 1057–1075, (1999).
- [11] Jenneke Fokker, Huib de Ridder, Piet Westendorp, and Johan Pouwelse, 'Psychological backgrounds for inducing cooperation in peer-to-peer television', in *Interactive TV: a Shared Experience*, pp. 136–145, (2007).
- [12] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry, 'Using collaborative filtering to weave an information tapestry', *Commun. ACM*, **35**(12), 61–70, (1992).
- [13] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl, 'Evaluating collaborative filtering recommender systems', *ACM Trans. Inf. Syst.*, **22**(1), 5–53, (2004).
- [14] Shang Hsu, Ming-Hui Wen, Hsin-Chieh Lin, Chun-Chia Lee, and Chia-Hoang Lee, 'Aimed- a personalized tv recommendation system', in *Interactive TV: a Shared Experience*, pp. 166–174, (2007).
- [15] E.J. Johnson, M. Steffel, and D.G. Goldstein, 'Making better decisions: From measuring to constructing preferences', *Health Psychology*, **24**(4), S17–S22, (2005).
- [16] R. Keeney and H. Raiffa, *Decisions with multiple objectives: preferences and value tradeoffs*, Cambridge University Press, Cambridge, 1993.
- [17] Micheline Lesaffre, Marc Leman, and Jean-Pierre Martens, 'A user-oriented approach to music information retrieval', in *Dagstuhl Seminar Proceedings 06171*, (2006).
- [18] Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis, 'Ht06, tagging paper, taxonomy, flickr, academic article, to read', in *Proceedings of the seventeenth conference on Hypertext and hypermedia*, pp. 31–40, Odense, Denmark, (2006). ACM.
- [19] A. Mehrabian, *Basic Dimensions for a General Psychological Theory*, OG&H Publishers, 1980.
- [20] John W. Payne, James R. Bettman, and David A. Schkade, 'Measuring constructed preferences: Towards a building code', *Journal of Risk and Uncertainty*, **19**(1), 243–270, (1999).
- [21] J.W. Payne, J.R. Bettman, and E. J. Johnson, *The Adaptive Decision Maker*, Cambridge University Press, 1993.

- [22] Bart Peintner, Paolo Viappiani, and Neil Yorke-Smith, 'Preferences in interactive systems: Technical challenges and case studies', *2009*, **29**(4), 13–24, (2009).
- [23] A. Pommeranz, W-P Brinkman, P. Wiggers, Joost Broekens, and Catholijn M. Jonker, 'Design guidelines for negotiation support systems: An expert perspective using scenarios', in *ECCE 2009*. ACM, (2009).
- [24] P. Pu, B. Faltings, and M. Torrens, 'User-involved preference elicitation', in *In IJCAI-03 Workshop on Configuration*, (2003).
- [25] Pearl Pu and Li Chen, 'User-involved preference elicitation for product search and recommender systems', *AI Magazine*, **29**(4), 93–103, (2009).
- [26] A.M. Rashid, I. Albert, D. Cosley, S.K. Lam, S.M. McNee, J.A. Konstan, and J. Riedl, 'Getting to know you: Learning new user preferences in recommender systems', in *IUI*, San Francisco, California, (2002). ACM.
- [27] S. Shearin and H. Lieberman, 'Intelligent profiling by example', in *Proceedings of the 6th international conference on Intelligent user interfaces*, pp. 145–151, Santa Fe, New Mexico, United States, (2001). ACM.
- [28] Baba Shiv and Alexander Fedorikhin, 'Heart and mind in conflict: The interplay of affect and cognition in consumer decision making', *Journal of Consumer Research: An Interdisciplinary Quarterly*, **26**(3), 278–92, (1999).
- [29] D. Simon, D. C. Krawczyk, and K. J. Holyoak, 'Construction of preferences by constraint satisfaction', *Psychological Science*, **15**, 331–336, (2004).
- [30] E. U. Weber and E. J. Johnson, 'Constructing preferences from memory', in *The Construction of Preference*, eds., S. Lichtenstein and P. Slovic, pp. 397–410, New York, NY, (2006). Cambridge University Press.
- [31] Robert B. Zajonc and Hazel Markus, 'Affective and cognitive factors in preferences', *Journal of Consumer Research*, **9**(2), 123, (1982).

Synthesizing Preference Information of Multiple Decision Makers in Terms of Collective Decision Rules

Salem Chakhar¹ and Inès Saad²

Abstract. We propose an approach based on DRSA (Dominance-based Rough Set Approach) method for synthesizing preference information of multiple decision makers in a multicriteria classification problem. The proposed approach takes as input a common information table and generates a set of collective decision rules representing a generalized description of the preference information of the decision makers.

1 INTRODUCTION

DRSA (Dominance-based Rough Set Approach) [3] is an extension of rough sets theory [5] to deal with multicriteria classification problems. It takes as input a *decision table* describing the *decision objects* and generates as output a set of *decision rules*. DRSA is a single decision maker oriented method. However, there are some proposals to extend DRSA to group decision making [7][4][1]. But these proposals have several shortcomings as discussed in Section 7.

The objective of this paper is to introduce a DRSA-based approach for synthesizing preference information of multiple decision makers in a multicriteria classification problem. The proposed approach takes as input a common information table and generates a set of collective decision rules representing a generalized description of the preference information of the decision makers.

The paper goes as follows. Section 2 presents the background. Section 3 introduces the approach. Section 4 presents the aggregation procedure. Section 5 deals with collective decision rules generation. Section 6 illustrates the approach through an application. Section 7 discusses some related work. Section 8 concludes the paper.

2 BACKGROUND

DRSA [3][4] is a rough sets-based multicriteria classification method. This method has been developed to overcome the shortcomings of rough set [5] in multicriteria classification problems. The idea of DRSA is to replace *indiscernibility* relation in rough approximations by *dominance* relation.

2.1 Basic notations and assumptions

Information about decision objects are often represented in terms of an *information table* where rows correspond to *objects* and columns correspond to *attributes*. The information table S is a 4-tuple $\langle U, Q, V, f \rangle$ where: U is a finite set of objects, Q is a finite set

of attributes, $V = \bigcup_{q \in Q} V_q$, V_q is a domain of attribute q , and $f : U \times Q \rightarrow V$ is an *information function* defined such that $f(x, q) \in V_q, \forall q \in Q, \forall x \in U$. The set of attributes Q is often divided into a sub-set C of *condition attributes* and a sub-set D of *decision attributes*. In this case, S is called *decision table*.

A series of assumptions are established first. The domain of condition attributes are supposed to be ordered to decreasing or increasing preference. Such attributes are called *criteria*. We assume that the preference is increasing with the value of $f(\cdot, q)$ for every $q \in C$. We also assume that the set of decision attributes D is a singleton $\{d\}$. Decision attribute d makes a partition of U into a finite number of decision classes $\mathbf{CI} = \{Cl_t, t \in T\}$, $T = \{0, \dots, n\}$, such that each $x \in U$ belongs to one and only one class in $\mathbf{CI}$. Further, we suppose that the classes are preference-ordered, i.e. for all $r, s \in T$, such that $r > s$, the objects from Cl_r are preferred to the objects from Cl_s .

The idea of rough set approach is the approximation of knowledge generated by the decision attributes by “*granules of knowledge*” generated by condition attributes. The sets to be approximated are:

$$Cl_t^{\geq} = \bigcup_{s \geq t} Cl_s, Cl_t^{\leq} = \bigcup_{s \leq t} Cl_s, t = 0, \dots, n.$$

Set $Cl_t^{\geq}$ is called the *upward union*. The assertion $x \in Cl_t^{\geq}$ means that “ x belongs to at least class Cl_t ”. Set $Cl_t^{\leq}$ is called the *downward union*. The assertion $x \in Cl_t^{\leq}$ means that “ x belongs to at most Cl_t ”.

2.2 Approximation of unions of classes

In DRSA the represented knowledge is a collection of upward and downward unions of classes and the “*granules of knowledge*” are sets of objects defined using a (weak) dominance relation. The dominance relation Δ_P , where $P \subseteq C$, is defined for each pair of objects x and y as follows:

$$x \Delta_P y \Leftrightarrow f(x, q) \geq f(y, q), \forall q \in P.$$

The “*granules of knowledge*” used for approximation in DRSA with respect to a set of criteria $P \subseteq C$ and object $x \in U$ are:

- $\Delta_P^+(x) = \{y \in U : y \Delta_P x\}$: the set of objects that dominate x ,
- $\Delta_P^-(x) = \{y \in U : x \Delta_P y\}$: the set of objects dominated by x .

Δ_P^+ and Δ_P^- are respectively called *P-dominating* set and *P-dominated* set. For each set of criteria $P \subseteq C$, the *P-lower* and *P-upper* approximations of $Cl_t^{\geq}$ are defined as follows:

- $\underline{P}(Cl_t^{\geq}) = \{x \in U : \Delta_P^+(x) \subseteq Cl_t^{\geq}\}$,
- $\bar{P}(Cl_t^{\geq}) = \bigcup_{x \in Cl_t^{\geq}} \Delta_P^+(x) = \{x \in U : \Delta_P^-(x) \cap Cl_t^{\geq} \neq \emptyset\}$.

¹ University of Paris-Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France, email: salem.chakhar@dauphine.fr

² MIS, University of Picardie Jules Verne, 33 Rue Saint Leu, 80039 Amiens Cedex 01, France; and Amiens Business School, 18, Place Saint-Michel, 80000 Amiens Cedex 01, France, email: ines.saad@u-picardie.fr

P -lower approximation of $Cl_t^{\geq}$ contains all the objects with P -dominating set are assigned with certitude to classes at most as good as Cl_t . P -upper approximation of $Cl_t^{\geq}$ contains all the objects with P -dominating set is assigned to a class at least as good as Cl_t .

Similarly, the P -lower and P -upper approximations of $Cl_t^{\leq}$ are defined as follows:

- $\underline{P}(Cl_t^{\leq}) = \{x \in U : \Delta_P^-(x) \subseteq Cl_t^{\leq}\},$
- $\bar{P}(Cl_t^{\leq}) = \bigcup_{x \in Cl_t^{\leq}} \Delta_P^-(x) = \{x \in U : \Delta_P^+(x) \cap Cl_t^{\leq} \neq \emptyset\}.$

P -lower approximation of $Cl_t^{\leq}$ contains all the objects with P -dominated set are assigned with certitude to a class at most as good as Cl_t . P -upper approximation of $Cl_t^{\leq}$ contains all the objects with P -dominated set is assigned to a class at least as good as Cl_t .

We also define the P -boundary sets of $Cl_t^{\geq}$ and $Cl_t^{\leq}$ as follows:

- $Bn_P(Cl_t^{\geq}) = \bar{P}(Cl_t^{\geq}) - \underline{P}(Cl_t^{\geq}),$
- $Bn_P(Cl_t^{\leq}) = \bar{P}(Cl_t^{\leq}) - \underline{P}(Cl_t^{\leq}).$

$Bn_P(Cl_t^{\geq})$ contains all the objects which are assigned both to a class better than Cl_t and to one or several classes worse than Cl_t . In other words, it contains objects with P -dominating set cannot be assigned with certitude to classes at least as good as Cl_t . Similarly, $Bn_P(Cl_t^{\leq})$ is the set of objects with P -dominated set cannot be assigned with certitude to classes at most as good as Cl_t .

2.3 Decision rules

The approximations of upward and downward unions of classes can serve to induce a set of “if ..., then ...” decision rules relating condition and decision attributes. An object $x \in U$ supports a decision rule if its description matches both the condition part and the decision part of this rule. A decision rule covers object x if the description of x matches at least the condition part of the rule. The strength of a decision rule is the number of objects supporting this rule.

2.4 Quality of classification

The quality of classification is defined by the following ratio:

$$\gamma_P = \frac{\text{card}(U - (\bigcup_{t=1, \dots, n} Bn_P(Cl_t^{\geq})))}{\text{card}(U)} \quad (1)$$

$$= \frac{\text{card}(U - (\bigcup_{t=0, \dots, n-1} Bn_P(Cl_t^{\leq})))}{\text{card}(U)}.$$

It expresses the pourcentage of objects that are assigned with certitude to a given class.

3 COLLECTIVE DECISION RULES CONSTRUCTION APPROACH

The proposed approach is composed of three phases: individual classification, aggregation, and generation of collective decision rules. The main input of the approach is a common information table I defined as $\langle U, C, V, f \rangle$ with a finite set U of objects and a finite set C of criteria. The output is a collection of collective decision rules representing a generalized description of the preference information provided by the decision makers. Let $H = \{1, \dots, i, \dots, h\}$ be a finite set of decision makers corresponding to h decision attributes $D_1, \dots, D_i, \dots, D_h$. Further, we suppose that decision attributes are defined on the same domain. We also assume that each decision maker $i \in H$ has a preference order on the universe U and that this preference order is represented by a finite set of preference ordered classes:

$$Cl_i = \{Cl_{t,i}, t \in T_i\}, T_i = \{0, \dots, n_i\},$$

such that $\bigcup_{t=1}^{n_i} Cl_{t,i} = U, Cl_{t,i} \cap Cl_{r,i} = \emptyset, \forall r, t \in T_i, r \neq t$, and if $x \in Cl_{r,i}, y \in Cl_{s,i}$ and $r > s$, then x is better than y for decision maker i .

3.1 Phase 1: Individual classification

In this first phase, each decision maker uses the common information table I to construct its own decision table S_i defined as $\langle U, C \cup D_i, V, f_i \rangle$ where D_i is a new decision attribute and f_i is an information function, both associated with decision maker i . Then, each decision maker runs the DRSA method using its decision table S_i as input. In terms of this phase, the classification conducted by each decision maker is characterized, among others, by:

- the P -lower approximation and P -boundary of $Cl_{t,i}^{\leq}$ and $Cl_{t,i}^{\geq}$, for each $t \in T_i$, and
- the quality of classification γ_P^i defined in similar way to Eq. (1).

3.2 Phase 2: Aggregation

The objective of this phase is to combine the outputs of the first phase in order to assign to each object $x \in U$ a collective assignment interval by using an aggregation procedure detailed in Section 4. First, we design by Cl the collective preference order obtained by the union of individual preference orders:

$$Cl = \{Cl_t, t \in T\}, T = \{0, \dots, n\},$$

such that each $x \in U$ belongs to one and only one class $Cl_t \in Cl$. This operation is correct since, as stated before, decision attributes are defined on the same domain. According to this definition, we have: $x \in Cl_{t,i} \Leftrightarrow x \in Cl_t, \forall x \in U, \forall t \in T$, and $\forall i \in H$.

The aggregation procedure can be represented as follows:

$$\begin{aligned} U &\rightarrow Cl \times Cl \\ x &\rightarrow I(x) = [l(x), u(x)] \end{aligned}$$

It is a mapping from U to $Cl \times Cl$ that associates to each $x \in U$ a collective assignment interval $I(x) = [l(x), u(x)]$, where $l(x)$ and $u(x)$ are respectively the lower and upper classes to which object x can be assigned. Details are given in Section 4.3.

3.3 Phase 3: Generation of collective decision rules

The objective of this phase is to use the DRSA method to infer a set of collective decision rules representing a generalized description of the preference information provided by the different decision makers. The application of DRSA method requires that the decision attribute be mono-valued. Thus, some simple rules are first used to construct a collective decision table with a mono-valued decision attribute (Section 5.1). Then, the DRSA method may be applied using the obtained collective decision table as input (Section 5.2).

4 AGGREGATION PROCEDURE

The aggregation procedure is composed of three steps.

4.1 Step 2.1: Normalization

The objective of this first step is to standardize the quality of classifications γ_P^i ($\forall i \in H$) using the following formula:

$$^i\gamma'_P = \frac{1}{h} \cdot \sum_{i=1}^h \gamma_P^i, \quad (i = 1, \dots, h). \quad (2)$$

4.2 Step 2.2: Computing the concordance and discordance powers

The aggregation procedure is based on the majority principle which is defined through the concordance and discordance powers. The semantic interpretation of these powers is similar to the same concepts employed in ELECTRE family of multicriteria methods; see[2]. However, they are defined, computed and used differently in the present paper.

4.2.1 Concordance power

First, we define the sets $L(x, Cl_t^{\leq})$ and $L(x, Cl_t^{\geq})$ as follows:

- $L(x, Cl_t^{\leq}) = \{i : i \in H \wedge x \in \underline{P}(Cl_{t,i}^{\leq})\}$,
- $L(x, Cl_t^{\geq}) = \{i : i \in H \wedge x \in \underline{P}(Cl_{t,i}^{\geq})\}$.

The first set represents the decision makers for which object x belongs to the lower approximation of $Cl_t^{\leq}$. The second one represents the decision makers for which object x belongs to the lower approximation of $Cl_t^{\geq}$. Next, the *concordance powers* for the assignment of x to $Cl_t^{\leq}$ and to $Cl_t^{\geq}$ are computed as follows:

$$L^+(x, Cl_t^{\leq}) = \sum_{i \in L(x, Cl_t^{\leq})} ^i\gamma'_P. \quad (3)$$

$$L^+(x, Cl_t^{\geq}) = \sum_{i \in L(x, Cl_t^{\geq})} ^i\gamma'_P. \quad (4)$$

$L^+(x, Cl_t^{\leq}) \in [0, 1]$ measures the power of coalition of decision makers that assign x to the lower approximation of $Cl_t^{\leq}$. $L^+(x, Cl_t^{\geq}) \in [0, 1]$ measures the power of coalition of decision makers that assign x to the lower approximation of $Cl_t^{\geq}$.

4.2.2 Discordance power

First, we define the sets $B(x, Cl_t^{\leq})$ and $B(x, Cl_t^{\geq})$ as follows:

- $B(x, Cl_t^{\leq}) = \{i : i \in H \wedge x \in Bn_P(Cl_{t,i}^{\leq})\}$,
- $B(x, Cl_t^{\geq}) = \{i : i \in H \wedge x \in Bn_P(Cl_{t,i}^{\geq})\}$.

The first set represents the decision makers for which object x belongs to the boundary of $Cl_t^{\leq}$. The second one represents the decision makers for which object x belongs to the boundary of $Cl_t^{\geq}$. Then, the *discordance powers* for the assignment of x to the boundary of $Cl_t^{\leq}$ and $Cl_t^{\geq}$ are computed as follows:

$$B^+(x, Cl_t^{\leq}) = \sum_{i \in B(x, Cl_t^{\leq})} ^i\gamma'_P. \quad (5)$$

$$B^+(x, Cl_t^{\geq}) = \sum_{i \in B(x, Cl_t^{\geq})} ^i\gamma'_P. \quad (6)$$

$B^+(x, Cl_t^{\leq}) \in [0, 1]$ measures the power of coalition of decision makers that assign x to the boundary of $Cl_t^{\leq}$. $B^+(x, Cl_t^{\geq}) \in [0, 1]$ measures the power of coalition of decision makers that assign x to the boundary of $Cl_t^{\geq}$.

4.3 Step 2.3: Definition of assignment intervals

Let $\theta \in [.5, 1.0]$ be a majority threshold and $\theta' \in [0, .5]$ be a veto threshold. Based on the concordance and discordance powers, we may distinguish four situations for the assignment of x to $Cl_t^{\leq}$:

	$B^+(x, Cl_t^{\leq}) < \theta'$	$B^+(x, Cl_t^{\leq}) \geq \theta'$
$L^+(x, Cl_t^{\leq}) \geq \theta$	$x \in Cl_t^{\leq}$	$x \notin Cl_t^{\leq}$
$L^+(x, Cl_t^{\leq}) < \theta$	$x \notin Cl_t^{\leq}$	$x \notin Cl_t^{\leq}$

These situations are summarized by the following assignment rule:

if $L^+(x, Cl_t^{\leq}) \geq \theta \wedge B^+(x, Cl_t^{\leq}) < \theta'$, **then** $x \in Cl_t^{\leq}$
else $x \notin Cl_t^{\leq}$ (*rule 1*)

This assignment rule can be explained as follows. An object x is assigned to $Cl_t^{\leq}$ if and only if:

- there is a “sufficient” majority of decision makers (in terms of their quality of classification) that assign x to $Cl_t^{\leq}$, and
- when the first condition holds, none of the minority of decision makers shows an “important” opposition to the assignment of x to $Cl_t^{\leq}$.

In similar way, four situations can be distinguished for the assignment of x to $Cl_t^{\geq}$:

	$B^+(x, Cl_t^{\geq}) < \theta'$	$B^+(x, Cl_t^{\geq}) \geq \theta'$
$L^+(x, Cl_t^{\geq}) \geq \theta$	$x \in Cl_t^{\geq}$	$x \notin Cl_t^{\geq}$
$L^+(x, Cl_t^{\geq}) < \theta$	$x \notin Cl_t^{\geq}$	$x \notin Cl_t^{\geq}$

These situations are summarized by the following assignment rule:

if $L^+(x, Cl_t^{\geq}) \geq \theta \wedge B^+(x, Cl_t^{\geq}) < \theta'$, **then** $x \in Cl_t^{\geq}$
else $x \notin Cl_t^{\geq}$ (*rule 2*)

This assignment rule can be explained as follows. An object x is assigned to $Cl_t^{\geq}$ if and only if:

- there is a “sufficient” majority of decision makers (in terms of their quality of classification) that assign x to $Cl_t^{\geq}$, and
- when the first condition holds, none of the minority of decision makers shows an “important” opposition to the assignment of x to $Cl_t^{\geq}$.

The application of these assignment rules on the set of objects U permits to associate to each object x a collective assignment interval $I(x) = [l(x), u(x)]$ where:

$$l(x) = \begin{cases} \operatorname{argmax}_{Cl_t} N_1(x), & \text{if } N_1(x) \neq \emptyset, \\ Cl_0, & \text{otherwise.} \end{cases} \quad (7)$$

$$u(x) = \begin{cases} \operatorname{argmin}_{Cl_n} N_2(x), & \text{if } N_2(x) \neq \emptyset, \\ Cl_n, & \text{otherwise.} \end{cases} \quad (8)$$

where $N_1(x) = \{Cl_t : x \in Cl_t^{\leq}\}$ and $N_2(x) = \{Cl_t : x \in Cl_t^{\geq}\}$. Set $N_1(x)$ contains the set of classes to which x is assigned by applying *rule 2*, while set $N_2(x)$ contains the set of classes to which x is assigned by applying *rule 1*.

The aggregation procedure is summed up in Algorithm 1. This algorithm runs in $O(|U| \cdot n \cdot h)$ where $|U|$ is the cardinality of U , n is the number of classes and h is the number of decision makers.

Algorithm 1 AggregationProcedure

Input: $P(Cl_{t,i}^{\leq}), P(Cl_{t,i}^{\geq})$: P -lower approx. ($i \in H; t \in T_i$)
 $Bn_P(Cl_{t,i}^{\leq}), Bn_P(Cl_{t,i}^{\geq})$: P -boundary ($i \in H; t \in T_i$)
 γ_P^i : quality of classification ($i \in H$)
Output: $I(x)$: Collective assignment interval ($\forall x \in U$)
1. Normalize γ_P^i ($i \in H$)
2. **for** each $x \in U$
3. **for** each $t \in T$
4. compute $L(x, Cl_t^{\leq}), B(x, Cl_t^{\leq}), L(x, Cl_t^{\geq}), B(x, Cl_t^{\geq})$
5. compute $L^+(x, Cl_t^{\leq}), B^+(x, Cl_t^{\leq}), L^+(x, Cl_t^{\geq}), B^+(x, Cl_t^{\geq})$
6. **if** $L^+(x, Cl_t^{\leq}) \geq \theta$ and $B^+(x, Cl_t^{\leq}) < \theta'$, **then** $x \in Cl_t^{\leq}$
7. **else** $x \notin Cl_t^{\leq}$ **end if**
8. **if** $L^+(x, Cl_t^{\geq}) \geq \theta$ and $B^+(x, Cl_t^{\geq}) < \theta'$, **then** $x \in Cl_t^{\geq}$
9. **else** $x \notin Cl_t^{\geq}$ **end if**
10. **end for**
11. $N_1(x) \leftarrow \{Cl_t : x \in Cl_t^{\leq}\}$
12. $N_2(x) \leftarrow \{Cl_t : x \in Cl_t^{\geq}\}$
13. **if** $N_1(x) \neq \emptyset$, **then** $l \leftarrow \text{argmax}_{Cl_t} N_1(x)$
14. **else** $l \leftarrow Cl_0$ **end if**
15. **if** $N_2(x) \neq \emptyset$ **then** $u \leftarrow \text{argmin}_{Cl_t} N_2(x)$
16. **else** $u \leftarrow Cl_n$ **end if**
17. $I(x) \leftarrow [l, u]$
18. **end for**

5 INFERENCE OF DECISION RULES

5.1 Construction of a collective decision table

The objective of this step is to construct a collective decision table S defined as $\langle U, C \cup D, V, g \rangle$ where D is a collective decision attribute and g is a collective information function defined as follows:

$$g(x, q) = \begin{cases} f(x, q), & \text{if } q \in C, \\ g(x, D), & \text{if } q = D. \end{cases} \quad (9)$$

Two cases may be distinguished for the definition of $g(x, D)$. The first holds when $l(x) = u(x)$. Here, object x is assigned to a single class and consequently we can set $g(x, D) = l(x)$ (or similarly $g(x, D) = u(x)$). The second case holds when $l(x) < u(x)$. This corresponds to the situation where object x is assigned to more than one class. To define $g(x, D)$ we may apply one of the following rules to reduce the collective assignment interval $I(x)$ to a single class:

- use the “min” operator on the collective assignment interval $I(x)$. This leads to $g(x, D) = l(x)$. (rule 3)
- use the “max” operator on the collective assignment interval $I(x)$. This leads to $g(x, D) = u(x)$. (rule 4)
- use the “median” operator on $l', \dots, u'$, where $l', \dots, u'$ is an ordered list issued from $l(x), \dots, u(x)$. (rule 5)

The proposed approach assumes an ordinal measurement scale. Hence, the median value may correspond to no decision class (when there is an even number of values). To avoid this problem, rule 5 can be subdivided into two rules:

- use the “floor” of the median value: $g(x, D) = \lfloor \mu(l', \dots, u') \rfloor$. (rule 5.1)
- use the “ceil” of the median value: $g(x, D) = \lceil \mu(l', \dots, u') \rceil$. (rule 5.2)

Function $\mu(\cdot)$ returns the median value. The collective assignment interval reduction step is formalized in Algorithm 2. **OrderedList**

in Algorithm 2 returns an ordered list from $(l(x), \dots, u(x))$. Algorithm 2 runs in $O(|U| \cdot k \log k)$ where $|U|$ is the cardinality of U and k is the number of values in $(l(x), \dots, u(x))$.

Algorithm 2 AssignmentIntervalReduction

Input: $I(x)$: Collective assignment interval ($\forall x \in U$)
 $rule$: Interval reduction rule
Output: $g(x, D), \forall x \in U$
1. **for** each $x \in U$
2. $l \leftarrow l(x)$
3. $u \leftarrow u(x)$
4. **if** $l = u$, **then** $g(x, D) \leftarrow l$
5. **else if** $rule$ is “min”, **then** $g(x, D) \leftarrow l$
6. **else if** $rule$ is “max”, **then** $g(x, D) \leftarrow u$
7. **else** $(l', \dots, u') \leftarrow \text{OrderedList}(l(x), \dots, u(x))$
8. $m \leftarrow \text{median}(l', \dots, u')$
9. **if** $rule$ is “floor”, **then** $g(x, D) \leftarrow \lfloor m \rfloor$ **end if**
10. **if** $rule$ is “ceil”, **then** $g(x, D) \leftarrow \lceil m \rceil$ **end if**
11. **end if**
12. **end if**
13. **end for**
14. **end for**

5.2 Inference of collective decision rules

The objective here is to apply DRSA using the collective decision table S as input. The application of DRSA at this level is the same as for a single decision maker. The output is a collection of decision rules synthesizing the preference information of the different decision makers. These rules can then be included in a knowledge-based decision support system [6] and used as basis for decision making.

6 APPLICATION

The problem considered concerns post-accident nuclear risk management in the southern France in which one of the authors was implied. For the purpose of this paper, only a subset of data is used. Further decision objects and names of decision makers are codified (confidentiality reasons). The problem considered involves 10 decision objects, 7 evaluation criteria, 3 decision makers (CM, PP and CAL), and six decision classes (Cl_0 to Cl_5). Decision objects correspond to a subset of the districts of the study area. The list of evaluation criteria is given in Table 1 and decision classes are given in Table 2.

Table 1. List of evaluation criteria

Code	Description
C_1	Radioecological vulnerability of agricultural area
C_2	Radioecological vulnerability of forest area
C_3	Radioecological vulnerability of urban area
C_4	Real estate vulnerability
C_5	Tourism vulnerability
C_6	Economic vulnerability of companies
C_7	Employment vulnerability

Table 2. Decision classes

Level	Class	Name
0	Cl_0	Normal situation
1	Cl_1	Very minor
2	Cl_2	Minor
3	Cl_3	Moderate
4	Cl_4	Major
5	Cl_5	Major and long-lasting

6.1 Phase 1: Individual classification

First, each decision maker runs the DRSA³ method using its own decision table obtained by adding a new decision attribute to the common information table. Decision tables used here are given in Table 3 where decision attributes D_1 , D_2 and D_3 correspond to decision makers CM, PP and CAL. The obtained quality of classifications are $\gamma_P^1 = 0.61$ (CM), $\gamma_P^2 = 0.33$ (PP), and $\gamma_P^3 = 0.33$ (CAL).

Table 3. Decision tables

Object	C_1	C_2	C_3	C_4	C_5	C_6	C_7	D_1	D_2	D_3
x_1	4	5	5	5	4	1	1	4	4	5
x_2	4	5	5	5	4	2	2	4	4	5
x_3	4	5	5	5	4	2	1	4	4	5
x_4	4	5	5	5	4	3	1	5	4	5
x_5	3	2	2	4	4	2	0	3	2	3
x_6	1	1	1	2	4	1	0	0	0	1
x_7	2	2	1	2	4	1	0	3	2	2
x_8	1	2	1	2	2	1	0	0	0	1
x_9	3	2	2	4	4	2	0	3	2	2
x_{10}	3	3	3	4	4	1	0	3	2	3

6.2 Phase 2: Aggregation

6.2.1 Step 2.1: Normalization

First, Eq. (2) is used to normalize the quality of classifications γ_P^1 , γ_P^2 , and γ_P^3 , which leads to: $^1\gamma'_P = .48$, $^2\gamma'_P = .26$, and $^3\gamma'_P = .26$.

6.2.2 Step 2.2: Computing the concordance/discordance powers

Concordance power For illustration, we only show the computing of $L^+(x_5, Cl_3^{\leq})$. The lower approximations for $Cl_3^{\leq}$ according to decision makers CM, PP and CAL are as follows:

- $\underline{P}(Cl_3^{\leq}) = \{x_5, x_6, x_7, x_8, x_9, x_{10}\}$. (CM)
- $\underline{P}(Cl_3^{\leq}) = \{x_5, x_6, x_7, x_8, x_9, x_{10}\}$. (PP)
- $\underline{P}(Cl_3^{\leq}) = \{x_8\}$. (CAL)

Hence, we have $L(x_5, Cl_3^{\leq}) = \{1, 2\}$. This means that only decision makers CM and PP assign x_5 to the lower approximation of $Cl_3^{\leq}$. Now, Eq. (3) can be used to compute the concordance power for object x_5 with respect to $Cl_3^{\leq}$:

$$L^+(x_5, Cl_3^{\leq}) = \sum_{i \in L(x_5, Cl_3^{\leq})} i \gamma'_P = 1 \gamma'_P + 2 \gamma'_P = .48 + .26 = .74.$$

The concordance powers of decision object x_5 with respect to $Cl_t^{\leq}$ ($t = 0, \dots, 4$) and $Cl_t^{\geq}$ ($t = 1, \dots, 5$) are given in Table 4.

Discordance power For illustration, we only show the computing of $B^+(x_5, Cl_4^{\geq})$. The boundaries for $Cl_4^{\geq}$ according to decision makers CM, PP and CAL are as follows:

- $Bn_P(Cl_4^{\geq}) = \emptyset$. (CM)
- $Bn_P(Cl_4^{\geq}) = \emptyset$. (PP)
- $Bn_P(Cl_4^{\geq}) = \{x_5, x_6, x_7, x_9, x_{10}\}$. (CAL)

Then, we get $B(x_5, Cl_4^{\geq}) = \{3\}$. This means that only decision maker CAL assigns x_5 to the boundary of $Cl_4^{\geq}$. By Eq. (6), the discordance power for the assignment of object x_5 to $Cl_4^{\geq}$ is:

$$B^+(x_5, Cl_4^{\geq}) = \sum_{i \in B(x_5, Cl_4^{\geq})} i \gamma'_P = 3 \gamma'_P = .26$$

The boundary powers of decision object x_5 with respect to $Cl_t^{\leq}$ ($t = 0, \dots, 4$) and $Cl_t^{\geq}$ ($t = 1, \dots, 5$) are summed up in Table 4.

³ Using 4eMKA, which is a stand-alone and free software implementing the DRSA method. See: <http://idss.cs.put.poznan.pl/site/4emka.html>.

Step 2.3: Definition of assignment intervals Here, assignment rules *rule 1* and *rule 2* given in Section 4.3 are used to associate to each object $x \in U$ a collective assignment interval $I(x)$. The majority and veto thresholds used in this application are $\theta = .5$ and $\theta' = .25$, respectively. Then, assignment *rule 1* and *rule 2* become:

$$\text{if } L^+(x, Cl_t^{\leq}) \geq .5 \wedge B^+(x, Cl_t^{\leq}) < .25, \text{ then } x \in Cl_t^{\leq} \\ \text{else } x \notin Cl_t^{\leq}$$

$$\text{if } L^+(x, Cl_t^{\geq}) \geq .5 \wedge B^+(x, Cl_t^{\geq}) < .25, \text{ then } x \in Cl_t^{\geq} \\ \text{else } x \notin Cl_t^{\geq}$$

The application of these rules to x_5 is summarized in Table 4 (fourth row). According to this table, it is easy to see that the first assignment rule is verified only for $Cl_3^{\leq}$ and $Cl_4^{\leq}$ while the second assignment rule is verified only for $Cl_1^{\geq}$ and $Cl_2^{\geq}$. In conclusion, we obtain: $x_5 \in Cl_3^{\leq}$, $x_5 \in Cl_4^{\leq}$, $x_5 \in Cl_1^{\geq}$ and $x_5 \in Cl_2^{\geq}$.

Table 4. Application of assignment rules (*rule 1* and *rule 2*) to object x_5

Cl_t	$Cl_0^{\leq}$	$Cl_1^{\leq}$	$Cl_2^{\leq}$	$Cl_3^{\leq}$	$Cl_4^{\leq}$	$Cl_1^{\geq}$	$Cl_2^{\geq}$	$Cl_3^{\geq}$	$Cl_4^{\geq}$	$Cl_5^{\geq}$
$L^+(x_5, Cl_t)$	0	0	0	.74	1	1	1	.48	0	0
$B^+(x_5, Cl_t)$	0	0	.52	.26	0	0	0	.52	0.26	0
Decision	No	No	No	Yes	Yes	Yes	Yes	No	No	No

Now, to define the assignment interval $I(x_5) = [l(x_5), u(x_5)]$, we use Eqs. (7) and (8) to define $l(x_5)$ and $u(x_5)$. Based on Table 4, we get: $N_1(x_5) = \{Cl_t : x_5 \in Cl_t^{\leq}\} = \{Cl_1, Cl_2\}$, and $N_2(x_5) = \{Cl_t : x_5 \in Cl_t^{\geq}\} = \{Cl_3, Cl_4\}$. Then, Eqs. (7) and (8) lead to:

- $l(x_5) = \arg\max_{Cl_t} N_1(x_5) = \arg\max_{Cl_t} \{Cl_1, Cl_2\} = Cl_2$.
- $u(x_5) = \arg\min_{Cl_t} N_2(x_5) = \arg\min_{Cl_t} \{Cl_3, Cl_4\} = Cl_3$.

Finally, the assignment interval for decision object x_5 is $I(x_5) = [Cl_2, Cl_3]$. For convenience, the assignment intervals for all decision objects are given in Table 5 (second column).

6.3 Phase 3: Generation of collective decision rules

Step 3.1: Construction of a collective decision table The objective here is to construct the collective decision table $\langle U, C \cup D, V, g \rangle$. The definition of $g(x, D)$, $\forall x \in U$ is summarized in Table 5 where columns “min”, “max”, “floor” and “ceil” refer to interval reduction rules *rule 3*, *rule 4*, *rule 5.1* and *rule 5.2*.

Table 5. The definition of $g(x, D)$ for different interval reduction rules

x_i	$I(x_i)$	min	max	floor	ceil
x_1	Cl_4, Cl_4	Cl_4	Cl_4	Cl_4	Cl_4
x_2	Cl_4, Cl_4	Cl_4	Cl_4	Cl_4	Cl_4
x_3	Cl_4, Cl_4	Cl_4	Cl_4	Cl_4	Cl_4
x_4	Cl_5, Cl_5	Cl_5	Cl_5	Cl_5	Cl_5
x_5	Cl_2, Cl_3	Cl_2	Cl_3	Cl_2	Cl_3
x_6	Cl_0, Cl_0	Cl_0	Cl_0	Cl_0	Cl_0
x_7	Cl_3, Cl_3	Cl_3	Cl_3	Cl_3	Cl_3
x_8	Cl_0, Cl_0	Cl_0	Cl_0	Cl_0	Cl_0
x_9	Cl_3, Cl_3	Cl_3	Cl_3	Cl_3	Cl_3
x_{10}	Cl_2, Cl_3	Cl_2	Cl_3	Cl_2	Cl_3

Step 3.2: Inference of collective decision rules The quality of classifications according to different interval reduction rules are given in Table 6. As it is shown in this table, interval reduction using the “max criterion” (*rule 4*) leads to the highest quality of classification (.83). The quality of classifications obtained by *rule 5.1* (floor) and *rule 5.2* (ceil) are equal to .72. In the three cases, we can conclude that the number of objects assigned with certitude to a given

class is acceptable. In the contrary, the quality of classification obtained by the “min” criterion (*rule 3*) is relatively low. Hence, the use of *rule 3* is not recommended in this illustrative application.

Table 6. The classification quality for different interval reduction rules

Rule	min	max	floor	ceil
γ_P	.28	0.83	.72	.72

A selection of collective decision rules generated using *rule 5.1* for interval reduction is given in Table 7. The first column in this table contains the decision rule. The second column contains objects supporting the rule. The last column indicates the strength of the rule. The description of these rules is straightforward. For illustration, we briefly comment two ones:

- **Rule 4:** if $f(x, q_5) \leq 3$, then $Cl_2^{\leq}$
- **Rule 20:** if $f(x, q_2) \leq 2 \wedge f(x, q_5) \leq 4$, then $Cl_2^{\geq}$

Rule 4 means that an object x is assigned to $Cl_2^{\leq}$ if its evaluation with respect to “Tourism vulnerability” criterion (q_5) is less or equal to 3. Rule 4 is supported only by decision objects x_8 and x_{15} . Its strength is equal to 40%.

Rule 20 says that object x is assigned to $Cl_2^{\geq}$ once (i) its evaluation with respect to “Radioecological vulnerability of forest area” criterion (q_2) is less or equal to 2, and (ii) its evaluation with respect to “Tourism vulnerability” criterion (q_5) is less or equal to 4. The decision objects supporting Rule 20 are: $x_1, x_2, x_3, x_4, x_5, x_7, x_9$, and x_{10} . The strength of Rule 20 is equal to 92.86%.

Table 7. A selection of collective decision rules

Rule	Supporting objects	Strength
Rule1: if $f(x, q_5) \leq 2$, then $Cl_0^{\leq}$	x_8	100%
Rule2: if $f(x, q_1) \leq 1$, then $Cl_2^{\leq}$	x_6, x_8	80%
Rule4: if $f(x, q_5) \leq 3$, then $Cl_2^{\leq}$	x_8, x_{15}	40%
Rule13: if $f(x, q_6) \leq 3$, then $Cl_5^{\leq}$	x_4	100%
Rule19: if $f(x, q_1) \leq 2$, then $Cl_2^{\geq}$	$x_1, x_2, x_3, x_4, x_5, x_7, x_9, x_{10}$	100%
Rule20: if $f(x, q_2) \leq 2 \wedge f(x, q_5) \leq 4$, then $Cl_2^{\geq}$	$x_1, x_2, x_3, x_4, x_5, x_7, x_9, x_{10}$	92.86%
Rule22: if $f(x, q_2) \leq 2 \wedge f(x, q_5) \leq 3$, then $Cl_2^{\geq}$	$x_1, x_2, x_3, x_4, x_5, x_7, x_9, x_{10}$	100%

7 RELATED WORK

In [7], the authors propose a DRSA-based methodology to group decision making with application to knowledge management. It contains four steps. First, a common decision table is constructed. Second, decision rules for each assignment example determined in the first step are inferred. The obtained results are checked for inconsistencies problems. Third, each decision maker solves the eventual inconsistency problems. Fourth, the analyst identifies collectively accepted decision rules. The main shortcoming of [7]’s methodology is its time consuming. In fact, the methodology requires, in conflicting situations, that the analyst conducts an in-depth discussion with the different decision makers in order to solve the conflicts. This is a time-consuming and difficult task.

The authors in [1] propose an argumentative multi-agent model based on a mediator agent in order to automate the resolution of conflicts between decision makers in [7]’s methodology. This approach allows the mediator agent to elicit preference of decision makers while exploiting and managing their points of view. Although this multi-agent system-based approach permits to automatize conflict resolution, it has one major shortcoming. In fact, the aggregation rule used in [1] is defined as a weighted-sum of four criteria: the number

of agents, the quality of classification, the number of rules and the average strength of rules. However, we think that the second and fourth criteria are similar, which may lead to over-evaluation.

Another extension of DRSA to support multiple decision makers is reported in [4] where the authors extend the lower and upper approximations and boundary concepts. More specifically, they introduce the concepts of downward and upward multi-union and mega-union. These concepts are then used to define lower and upper approximation for unions of classes. We think that this extension has three main shortcomings. First, it is difficult for decision makers to understand the aggregation mechanism adopted in [4]. Second, [4]’s approach is expensive in computational time. Third, there is no dialogue between the different decision makers.

8 CONCLUSION

We proposed a three-phase DRSA-based approach for group multi-criteria classification problems. The proposed approach takes as input a common information table and generates a set of collective decision rules representing a generalized description of the preference information of the decision makers. The paper detailed the approach and illustrates it through a real-world application. The proposed approach has several merits. First, as it is based on DRSA, the approach: (i) does not require any preference parameter, (ii) is able to deal with lack of information, and (iii) is able to detect and handle inconsistency problems in the decision table. Second, the approach uses the majority rule which is characterized by (i) its simplicity, anonymity and neutrality, and (ii) its low-demanding in terms of computational time. Third and in contrary to [7][1] (which are very demanding in terms of dialogue) and [4] (which requires no dialogue), the proposed approach is not very demanding in terms of dialogue between the different decision makers.

Several topics need to be investigated in the future. The first one concerns the use of decision rules-related information to define the assignment rules. The second one is related to the use of other classification methods that accept interval-based assignment for decision objects. The third one concerns the use of input level aggregation-oriented schema.

REFERENCES

- [1] I. Brigui-Chitioui and I. Saad, ‘A multi-agent approach for collective decision making in knowledge management’, *Group Decision and Negotiation*, (2010). to appear.
- [2] J.R. Figueira, S. Greco, and M. Ehrgott, eds., *Multiple criteria decision analysis: State of the art surveys*, chapter ELECTRE methods, 133–162, Springer-Verlag, New York, 2005.
- [3] S. Greco, B. Matarazzo, and R. Slowinski, ‘Rough sets theory for multicriteria decision analysis’, *European Journal of Operational Research*, **129**(1), 1–47, (2001).
- [4] S. Greco, B. Matarazzo, and R. Slowinski, ‘Dominance-based rough set approach to decision involving multiple decision makers’, in *Proceedings of the 5th International conference Rough sets and current trends in computing (RSCTC)*, Kobe, Japan, November 6-8, eds., S. Greco, Y. Hata, S. Hirano, M. Inuiguchi, S. Miyamoto, H.S. Nguyen, and R. Slowinski, volume 4259 of *LNAI*, 306–317, Springer-Verlag, Berlin Heidelberg, (2006).
- [5] Z. Pawlak, *Rough set. Theoretical aspects of reasoning about data*, Kluwer Academic Publishers, Dordrecht, 1990.
- [6] I. Saad and S. Chakhar, ‘A decision support for identifying crucial knowledge requiring capitalizing operation’, *European Journal of Operational Research*, **195**(3), 889–904, (2009).
- [7] I. Saad, C. Rosenthal-Sabroux, and M. Grundstein, ‘Improving the decision making process in the design project by capitalizing on company’s crucial knowledge’, *Group Decision and Negotiation*, **14**(2), 131–145, (2005).

Multi-agent soft constraint aggregation: a sequential approach

Giorgio Dalla Pozza, Francesca Rossi, K. Brent Venable
University of Padova, Italy.

Email: gdallapo@studenti.math.unipd.it, frossi@math.unipd.it, kvenable@math.unipd.it

Abstract. We consider a scenario where several agents express their preferences over a common set of variable assignments, by means of a soft constraint problem for each agent, and we propose a procedure to compute a variable assignment which satisfies the agents' preferences best. Such a procedure considers one variable at a time and, at each step, asks all agents to express its preferences over the domain of that variable. Based on such preferences, a voting rule is used to decide on which value is the best for that variable. At the end, the values chosen constitute the returned variable assignment.

We study several properties of this procedure and we show that the use of soft constraints allows for a great flexibility on the preferences of the agents, compared to similar work in this setting where agents model their preferences via CP-nets, where several restrictions on the agents' preferences need to be imposed to obtain similar properties.

1 Introduction

In many real-life situations, we need to express our preferences over a set of objects, in order to select the most preferred one. This may be useful, for example, when activating a search over the web, or when looking for a product in a catalogue, or when trying to decide on the best resource allocation or timetabling.

Expressing our preferences over a set of objects means giving an ordering to this set, where more preferred objects appear earlier in the ordering. This ordering need not be total, that is, we may have incomparable objects. Indifference may be present as well, since some objects may be equally preferable to us.

When the set of objects is small, we may just present them ordered to express our preferences over them. However, when the set of objects is very large, this is unfeasible. Unfortunately, this is usually the case in real-life situation, since often objects have a combinatorial structure, that is, each object can be seen as the combination of certain features, where each feature has a set of possible instances. Consider for example a PC: usually it is not seen as a single item, but as a combination of a features, such as its processor, its hard disk, its RAM, and its monitor. Each of these features has some possible instances, and a PC is the combination of such feature instances.

Usually the instances for each feature are not many, but because of the combination the set of objects can still be very large. Consider an example, where we need to choose a menu for a meal, composed of a first course, a main course, a dessert, and a wine. Assuming to have 5 choices for each of these, we should choose among $5^4 = 625$ different meals. Fortunately, when the set of objects has a combinatorial structure, we may describe our preference in a compact way, using one of the several formalisms available in the literature, such

as soft constraints [7] and CP-nets [2]. The aim of such structures is to allow one to express in time and space polynomial in the number of features an ordering over the set of all objects, which may be exponential in such a number.

A structure such as a CP-net or a soft constraint problem automatically induces a preference ordering over the set of objects. For such formalisms, the literature tells us how expensive it is to find the most preferred object, or to test if an object is the most preferred, or also to test whether one object is more preferred to another one. For example, for both types of structures, it is NP-hard to find the most preferred object, although it becomes linear in the number of features when some topological restrictions hold.

Often, we cannot take decisions by ourselves, but we need to compromise our preferences with those of other people. To come back to the dinner example above, we may think of a set of friends who want to organize a dinner and need to agree on a common menu. This is the kind of scenarios we consider in this paper. More precisely, we consider a set of agents, each expressing its preferences over a common set of objects with a combinatorial structure. The goal is now to combine such preferences and to find the object which is the most preferred according to this combination. Of course, the resulting object will depend on the chosen rule to combine the preferences of the agents.

Voting theory [1] is a very wide research area, between economic theory and operations research, that considers exactly these scenarios: in an election, voters (that we would call agents) vote by expressing their preferences over a set of candidates (that we would call objects), and a voting rule decides who the winner candidate is. Thus it is natural to look in this area for possible approaches to multi-agent preference aggregation.

Voting theory provides many rules to aggregate preferences. If there are only two objects, the best rule, according to many criteria, is majority voting. When there are more than two objects, there are many voting rules one could use (Plurality, Borda, STV, approval, etc.), each with its pros and cons. Each one of such rules takes in input (a part of) the preference orderings of the agents and gives as output the "winner" object, that is, the object that is considered to be the best according to the rule.

The most naive way to use voting rules in our context consists of choosing a voting rule and giving to it what it needs to know about the preference orderings of the agents, then running the voting rule and see what result comes out. This is however not feasible in general. In fact, if the chosen voting rule needs to know a large part of the preference ordering from the agents, it may take exponential time only to give the input to the rule.

A valid alternative is to use the voting rule several times, on each feature of the object set. That is, the rule asks the agents to provide their preferences on each feature at a time, and at each step a winner instance for a certain feature will be returned. At the end, all the winner instances will constitute the winning object.

This approach is certainly more attractive computationally, since usually the number of instances of each feature is small. Also, in special cases, it may return useful results. For example, if the preferences of the agents on the various features are independent of each other, then the winning object will be optimal for all agents. Unfortunately, this is a very rare case, since usually there are dependencies among the various features. What can be done in this situation is to study when, even in presence of dependencies among features, voting can be performed on each single feature at a time rather than on complete objects. This study has been done already for CP-nets [6], showing that a sequential single-feature voting protocol can find a winner object in polynomial time, and have several other desirable properties, when the CP-nets satisfy certain conditions on their dependencies. In particular, the CP-nets must be acyclic, and their dependency graphs must all be *compatible* with a given graph ordered according to the feature ordering in the voting procedure. We are now interested to undergo the same study for soft constraints.

More precisely, in our setting there is a common set of variables and domains, and each agent expresses its preferences over all variable assignments via a soft constraint problem. Given a variable ordering, say $v_1, \dots, v_n$, and a sequence of n voting rules $r_1, \dots, r_n$, the sequential voting procedure is simply a sequence of as many steps as the number of variables, where at each step i agents are required to report their preferences over the domain of variable v_i . To do this, each agent needs to compute the influence of the whole soft constraint problem over v_i . Voting rule r_i is then applied to such preferences over the domain of v_i , and the result is a value d_i for v_i . Step i ends by adding constraint $v_i = d_i$ to the soft constraint problems of all agents. After all n steps have been executed, the collection of all computed values is the variable assignment chosen by the sequential procedure.

To make each step (and the whole procedure) polynomial in the number of agents and variables, we have to assume that each r_i is polynomial to compute, and also that each agent can easily compute its preferences over a single variable. For example, we may assume that agents model their preferences via tree-shaped soft constraints.

Our results are similar to those obtained in [6], where agents express their preferences with CP-nets. However, the use of soft constraints allows us to avoid imposing many restrictions on the preferences of the agents. In fact, contrarily to CP-nets, soft constraints are not directional, and thus information can flow from one variable of a constraint to another one without a predefined ordering between them. This allows us to not tie the variable ordering used by the sequential procedure to the topology of the constraint graph of each agent. This makes the approach much more generally applicable. In fact, the tractability assumption over the constraint graphs is similar to the assumptions that CP-nets are acyclic. However, we do not need to impose that the constraint graphs are mutually compatible and with a graph structure based on the variable ordering.

A viable alternative to the sequential voting approach, that we don't consider in this paper, would be to define voting rules whose input is not a preference ordering, but a compact structure representing it. In this way, the drawbacks outlined above of the one-step approach would disappear, since the rule would directly work on a small structure rather than on a large ordered set.

Another alternative aggregation strategy is that of a direct aggrega-

tion of all the soft constraint networks at once. This path has been followed in [8, 5] among others. With respect to this approach, the one presented in this paper has the advantage of having less restrictive assumptions that guarantee tractability and that it allows for the use of several voting rules instead of a single preference combination criterion.

2 Background

2.1 Soft Constraints

A soft constraint [7] is a constraint [4] where each instantiation of its variables has an associated value from a (totally or partially ordered) set which has the structure of a c-semiring. A c-semiring is defined by $\langle A, +, \times, 0, 1 \rangle$ where A is this set of values, $+$ is a commutative, associative, and idempotent operator, $\times$ is used to combine preference values and is associative, commutative, and distributes over $+$, 0 is the worst element, and 1 is the best element. The c-semiring S induces a partial or total order $\leq_S$ over preference values, where $a \leq_S b$ iff $a + b = b$. A Soft Constraint Satisfaction Problem (SCSP) is a tuple $\langle V, D, C, A \rangle$ where V is a set of variables, D is the domain of the variables and C is a set of constraints over V associating values from c-semiring A .

A classical CSP [4, 9] is just a soft CSP where the chosen c-semiring is $S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$. Fuzzy CSPs [7] are instead modeled with $S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle$. That is, we want to maximize the minimum preference. For weighted CSPs, the c-semiring is $S_{WCSP} = \langle R^+, min, +, +\infty, 0 \rangle$: preferences are interpreted as costs from 0 to $+\infty$, and we want to minimize the sum of costs.

As an example, consider the following fuzzy CSP where $V = \{X, Y\}$, $D = \{a, b\}$ and $C = \{c_Y, c_{XY}\}$. Soft constraint c_Y is defined over the values of Y and associates preference 0.4 to a and to 0.7 to b . Constraint c_{XY} , instead, is defined over X and Y and associates 0.9, 0.8, 0.7, 0.6 to, respectively, tuples $(X = a, Y = a)$, $(X = a, Y = b)$, $(X = b, Y = a)$ and $(X = b, Y = b)$.

Two main operations have been defined on soft constraints: combination, denoted with $\otimes$, and projection, denoted with $\downarrow$. In words, combining two constraints means building a new constraint involving all the variables of the original ones, and which associates to each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associated by the original constraints to the appropriate sub tuples. Considering the example of fuzzy CSP above, we have that $c_Y \otimes c_{XY}$ is a constraint on X and Y associating 0.4, 0.7, 0.4 and 0.6 to, respectively, tuples $(X = a, Y = a)$, $(X = a, Y = b)$, $(X = b, Y = a)$ and $(X = b, Y = b)$.

Projecting a constraint on a subset of variables means eliminating the others by associating to each tuple over the remaining variables a semiring element which is the sum of the elements associated by the original constraint to all the extensions of this tuple over the eliminated variables. For example, constraint $c_Y \otimes c_{XY} \downarrow_X$ is a constraint defined only over X , i.e. the variable on which we are projecting, and associates 0.7 to a and 0.6 to b , since 0.7 is the highest preference of any tuple of $c_Y \otimes c_{XY}$ where $X = a$ and 0.6 is the highest preference of any tuple of $c_Y \otimes c_{XY}$ where $X = b$.

To obtain the solution of an SCSP, we just combine all constraints. Thus, solving an SCSP means inducing an ordering over the set of complete assignments. In the case of fuzzy and weighted CSPs, such and ordering is total with ties. In the example above, the induced ordering has $(X = a, Y = b)$ at the top with a preference of 0.7,

$(X = b, Y = b)$ just below with 0.6 and $(X = b, Y = a)$ and $(X = a, Y = b)$ tied at the bottom with 0.4. An optimal solution of an SCSP is then a complete assignment with an undominated preference. The optimal solution of the example is $(X = a, Y = b)$. Finding an optimal solution in a set of soft constraints is an NP-hard problem.

Constraint propagation in SCSPs may be very helpful in improving the behavior of the search. For some classes of constraints, constraint propagation is enough to solve the problem [3]. This is the case for tree-shaped fuzzy CSPs, where directional arc-consistency (DAC), applied bottom-up on the tree shape of the problem, is enough to make the search for an optimal solution backtrack-free. DAC is also enough to compute the preferences over the values of the root variable, in dependence of the rest of the problem. That is, DAC is equivalent to combining all constraints and projecting over the root variable.

For the purpose of this paper we note that, if we project the solution of a SCSP over a single variable, we obtain a total order with ties over the values of that variable, where each value is associated to the preference of the best solution of the SCSP having such variable instantiated to such a value. Given an SCSP P and one of its variables v , we will denote as $top(v, P)$ the set of values of v that are assigned the highest preference in such an ordering. In our running example, if we consider $c_Y \otimes c_{XY} \downarrow_X$, the induced ordering over the values of the domain of X is $a > b$.

2.2 Voting Rules

A voting rule allows a set of voters to choose one among a set of candidates. Voters need to submit their vote, that is, their preference ordering over the set of candidates, and the voting rule aggregates such votes to yield a final result, usually called the winner.

In the classical setting [1], given a set of candidates C , a *profile* is a sequence of orderings over the set of candidates, one for each voter. Usually, such orderings are total orders, however several extensions have been studied such as when the orderings are partial orders or total orders with ties. Given a profile, a *voting rule*, also known as a social choice function, maps it onto a single winning candidate.

In this paper, we will often use a terminology which is more familiar to multi-agent settings, and we will therefore sometimes call "agents" the voters, "solutions" the candidates, and "decision" or "best solution" the winning candidate.

Some examples of widely used voting rules are:

- Plurality: each voter states who the preferred candidate is, and the candidate who is preferred by the largest number of voters wins;
- Majority: like plurality, but with only two candidates;
- Borda: given m candidates, each voter gives a ranking of all candidates and the i^{th} ranked candidate scores $m - i$; the candidate with the greatest sum of scores wins;
- Approval: each voter approves between 1 and $m - 1$ candidates on m total candidates; the candidate with most votes of approval wins;
- Cup (or sequential majority voting): a sequence of pairwise majority elections is performed, according to a tree which sets the agenda for the pairwise elections; at each step, a candidate faces another candidate; the loser is eliminated and the winner goes on to the next step; the process continues until only one candidate remains.
- Copeland: the winner is the candidate that wins the most pairwise competitions against all the other candidates.

The research on voting theory has mainly been concerned with the definition of properties of voting systems that are desirable. These properties are desirable also in automated contexts, so we will recall a few of them, since we will later be interested in studying their presence (or absence) in the preference aggregation system we propose:

- Condorcet-consistency: A voting rule is Condorcet-consistent if, when a candidate who beats every other in pairwise elections (namely, a Condorcet winner) exists, that candidate is always elected; Condorcet winners are unique and may not exist.
- Anonymity: A voting rule is anonymous when the results of an election are the same even under a permutation on the voters' set. This means that the result is independent of who are the voters, or in which order they are considered, but it depends only on their set.
- Neutrality: A voting rule is neutral when, for any permutation on the set of candidates, the permuted results of an election are identical to the results of the same election where the permutation on the candidates' set occurs before applying the voting rule. This is the analogous of anonymity for the candidates: the result does not depend on the names of the candidates.
- Consistency: A voting rule is consistent if, when considering preferences of 2 disjoint sets of voters - who decide over the same issues and have identical final results - the result obtained by a vote of the joint set of voters is the same as the ones obtained by the disjoint set of voters.
- Participation: A voting rule is participative if, given any profile, and given a new vote over a set of issues by a new voter, the result obtained from the new profile is equally or more preferred by the new voter. This means that a voter has an incentive to participate in the voting process, since by doing so he can get a better result.
- Efficiency (or unanimity): A voting rule is efficient if, given a winner over an election, there's no candidate who is preferred to the winner by all voters. This means that, when all voters agree that a certain candidate should win, he will be declared the winner.

These properties are present in some of the voting rules defined above. More precisely, all such rules are anonymous and neutral. Moreover, all but Cup are efficient, only Cup and Copeland are Condorcet consistent, and all but Cup and Copeland are consistent and participative.

3 Soft profiles

As noted above, a soft constraint problem is a way to express our preferences over a set of objects with a combinatorial structure. The set of objects over which preferences are expressed is the set of all assignments of all variables of the problem. If we have n variables with m values in each variable domain, this set contains m^n objects. In the fuzzy example, each variable assignment is given a preference between 0 and 1, and assignments with a higher preference value are more preferred.

Assume we have a set of agents, each one expressing his preferences over a common set of objects via a soft constraint problem. Since the objects are common to all agents, this means that all the soft constraint problems have the same set of variables and the same variable domains. However, they may have different constraints, as well as different preferences over the variable domains. We will call a *soft profile* the preference of a set of m agents, identified by a triple (V, D, P) : a set of variables V , a sequence D of $|V|$ domains, and a sequence P of m soft constraint problems over variables in V with

domains in D . A *fuzzy profile* is the same as a soft profile, except that the preferences of each agent are modelled via a fuzzy constraint problem.

4 Sequential preference aggregation

The idea is to sequentially vote on each variable via a voting rule. We do not restrict ourselves to use always the same voting rule for all variables, so we will have a sequence of as many voting rules as the variables in V .

Given a profile (V, D, P) , assume $|V| = n$, and consider an ordering of such variables $O = \langle v_1, \dots, v_n \rangle$ and a sequence of voting rules $R = \langle r_1, \dots, r_n \rangle$. The sequential voting procedure we propose is a sequence of n steps, where at each step i :

1. We ask all agents to report their preference ordering over the domain of variable v_i . If we have m agents, let us call such preference orderings $\langle po_i, \dots, po_m \rangle$.
2. We apply voting rule r_i to this profile, returning a winning assignment for variable v_i , say d_i . If there are ties in the result, the first one following a lexicographical order will be taken.
3. We add the constraint $v_i = d_i$ to the preferences of each agent.

After all n steps have been executed, we collect the winning assignments in the tuple $\langle d_1, \dots, d_n \rangle$, which will be reported as the chosen assignment for the variables in V . We write $Seq_{O,R}(V, D, P) = \langle d_1, \dots, d_n \rangle$.

This short description of the sequential voting procedure does not say what it means for an agent to report their preference ordering over the domain of variable v_i . In general, since we do not make any assumption on the voting rules r_i , the agent needs to provide the rule with a preference ordering over the whole domain of v_i . Since this variable can be connected to other parts of the agent's soft constraint problem, in order to report the correct preferences over the domain of v_i , the agent needs to consider the influence of the rest of the problem over v_i . This means that, in general, the agent needs to compute the projection over v_i of its whole soft constraint problem. This is a soft unary constraint over v_i , which is the desired preference ordering, to be given to the voting rule. Once this is provided by each agent, the rule is applied to the profile and the winner is computed. We note that all the rules we consider in this paper are polynomial to compute.

On the other hand the task of solving the SCSP is in general NP-hard, so it may require exponential time to accomplish it, unless the class of constraint problems used by the agent is tractable. This is for example the case of tree-like shaped soft constraint problems, which are polynomial to solve. Thus, if agents decide to express their preferences via tree-shaped soft constraint problems, at each step of the sequential voting procedure it is polynomial to provide the input to the voting rule relevant for that step. While being tractable, tree-shaped SCSPs can represent a wide variety of interconnections between preferences over issues.

We note that all the rules we have considered are polynomial to compute. Thus

5 Properties of the sequential voting procedure

We will now prove several properties of this sequential preference aggregation procedure.

5.1 Order independence

The first thing we may wonder is how much the choice of the ordering O influences the result of our procedure. Of course, in general,

different orderings will lead to different results. However, some ordering will produce the same result. Intuitively, these are orderings that differ just for the relative position of variables that are independent of each other, according to all agents.

More precisely, given an ordering O over V , this can be used to direct all edges in the constraint graph of each agent, by setting the direction from v_1 to v_2 if v_1 is earlier than v_2 in O . Let us call $G_i(O)$ this directed graph relative to agent i . If two variables v and v' are not connected in the transitive closure of this graph, then they are said to be independent.

Theorem 1 *Consider any two orderings O_1 and O_2 over V , that differ for the relative position of two variables v_1 and v_2 , and assume these two variables are independent in $G_i(O)$, for all i . Then $Seq_{O_1,R}(V, D, P) = Seq_{O_2,R}(V, D, P)$.*

To prove this statement, we just need to prove that, if two variables v_i and v_j are independent for an agent, its preference ordering over v_i , given that v_j has already been instantiated, is the same as the one when v_i has still its whole domain. This is certainly so, since the two variables are independent in the agents' constraint graph.

Given a soft profile, we may wonder if there exist a *best* variable ordering to use for the sequential voting procedure, where by *best* we mean that the resulting variable assignment reflects the preferences of the agents as much as possible. If v_i depends on v_j for all agents, then we should choose an ordering where v_i comes before v_j . If we order such two variables in the opposite way, agents could regret what they reported as their preferences over v_j when v_i will be instantiated.

5.2 Condorcet consistency

It is now natural to ask ourselves if the result returned by the sequential voting procedure has some relation with what is considered to be most preferred by the agents. In this respect, it is natural to consider the notion of Condorcet winner, which is classical in voting theory.

As defined above, a Condorcet winner (CW) is a candidate which is strictly or equally preferred to any other candidate by a majority of agents. Given a totally ordered profile, as in classical voting theory, there can be zero or exactly one Condorcet winner. In our context, however, since we may have ties in the preference orderings of the agents, there could be more than one Condorcet winner, since several variable instantiations could be considered optimal for a majority of agents. For any voting rule, it is very desirable that it is Condorcet-consistent, that is, that it returns a Condorcet winner if there is one.

First, we define the notion of sequential Condorcet winner (SCW). Given an SCSP Q , we will denote as $Q|_{v_1=d_1, \dots, v_h=d_h}$ the problem obtained from Q by fixing variables $v_1, \dots, v_h$ to the corresponding values. Let P_i denote the fuzzy constraint problem of agent i . Given a soft profile (V, D, P) with m agents and n variables, and an ordering O over V , $\langle d_1, \dots, d_n \rangle$ is a SCW iff, for all $j = 1, \dots, n$, $|\{i | d_j \in \text{top}(v_j, P_i|_{v_1=d_1, \dots, v_{j-1}=d_{j-1}})\}| > m/2$. In words, a sequential Condorcet winner is the combination of local Condorcet winners.

If all the local rules are Condorcet consistent, the sequential voting procedure returns a SCW by definition. However, to conclude that Seq is Condorcet consistent, we need to prove that SCW = CW. The following results shows that a CW is always an SCW, but unfortunately the opposite does not hold.

Theorem 2 *Given a soft profile (V, D, P) and an ordering O over V , if d is a CW for (V, D, P) , then it is a SCW for (V, D, P) . Thus, if Seq is Condorcet consistent, all local voting rules are so.*

In fact, if $d = (d_1, \dots, d_n)$ is a CW, then a majority of voters prefers it to all other candidates. Thus, at each step i of the sequential voting procedure, the same majority prefers d_i to all other values in the domain of v_i given the values already chosen for the previous variables.

The opposite does not hold in general, even if all voting rules are Condorcet consistent.

Theorem 3 *If all local voting rules are Condorcet consistent, the sequential voting procedure may be not Condorcet consistent.*

In fact, consider a fuzzy profile (V, D, P) where: $V = \{X, Y\}$, $D_X = D_Y = \{a, b\}$ and there are 5 agents. The fuzzy SCSPs of all agents have a single constraint over $\{X, Y\}$. For two agents we have: $\text{def}(X = a, Y = b) = 0.9$, $\text{def}(X = b, Y = b) = 0.8$, $\text{def}(X = a, Y = a) = 0.7$, $\text{def}(X = b, Y = a) = 0.6$; for one agent: $\text{def}(X = a, Y = a) = 0.9$, $\text{def}(X = a, Y = b) = 0.8$, $\text{def}(X = b, Y = a) = 0.7$, $\text{def}(X = b, Y = b) = 0.6$; for the other two agents: $\text{def}(X = b, Y = a) = 0.9$, $\text{def}(X = b, Y = b) = 0.8$, $\text{def}(X = a, Y = a) = 0.7$, $\text{def}(X = a, Y = b) = 0.6$. When each agent solves the problem and projects on variable X , for the first two agents we have $\text{pref}_X(a) = 0.9$ and $\text{pref}_X(b) = 0.8$; for the third agent $\text{pref}_X(a) = 0.9$ and $\text{pref}_X(b) = 0.7$; and for the last two agents $\text{pref}_X(a) = 0.7$ and $\text{pref}_X(b) = 0.9$. Thus, 3 over 5 agents agree that $X = a$ is optimal. Since the voting rule r_X is Condorcet-consistent, this value will be chosen for X . Given $X = a$, the preferences of the agents for Y are: for the first two agents $\text{pref}_Y(a) = 0.7$ and $\text{pref}_Y(b) = 0.9$; for the third agent $\text{pref}_Y(a) = 0.9$ and $\text{pref}_Y(b) = 0.7$; for the last two agents $\text{pref}_Y(a) = 0.7$ and $\text{pref}_Y(b) = 0.6$. Thus $Y = a$ will be chosen, since r_Y is Condorcet consistent, and $(X = a, Y = a)$ will be the SCW. However, $(X = a, Y = a)$ is not a CW, since the majority of the agents prefers $(X = b, Y = b)$. This means that there could be results of the sequential voting procedure that are not CWs. To make sure that the procedure is Condorcet consistent, we need to impose some restrictions on the profile, similarly to what is done in [6].

5.3 Anonymity

It is also important to make sure that a preference aggregation system does not depend on the names or the order of the agents. This corresponds to saying that the rule is anonymous.

In our setting, a permutation of voter set corresponds, basically, to a permutation of the soft constraint problems. It is easy to see that if the sequential voting rule respects anonymity, then also all the local voting rules do so, and, vice versa, if all the local voting rules are anonymous, so is the resulting sequential rule.

5.4 Neutrality

Neutrality is a property that requires for a rule to be insensitive to permutations of the candidates. This means that the result does not depend on the names of the candidates, but only on their position in the preference orderings. It is thus very important that a voting system is neutral.

We note that the candidates of the local voting rules are the values in the variable domains, while the candidates of the sequential voting rule are the complete assignments to all variables.

While a permutation of the values in the domains always corresponds to a permutation of the variable assignments, not all of the permutations of variable assignments can be obtained via permutations of domain values. For example, if we have assignments: $s_1 =$

$(X = a, Y = b)$, $s_2 = (X = a, Y = b)$, $s_3 = (X = b, Y = a)$, and $s_4 = (X = b, Y = b)$, the permutation that swaps s_1 and s_2 , leaving s_3 and s_4 fixed, cannot be modelled by a permutation of domain values.

From this observation it derives that neutrality of the local voting rules does not imply neutrality of the sequential voting rule, while the neutrality of the sequential voting rule implies the neutrality of each local voting rule.

5.5 Consistency

As defined above, a voting rule r is consistent if, when considering two profiles P_1 and P_2 with disjoint sets of voters, who vote over the same candidates, such that $r(P_1) = r(P_2)$, we have $r(P_1 \cup P_2) = r(P_1)$.

Theorem 4 *All the local voting rules in R are consistent if and only if $\text{Seq}_{O,R}$ is consistent.*

In fact, if all the local rules are consistent, at every step i of the sequential procedure, applied to profile $P_1 \cup P_2$, the result for variable v_i is the same as the result in profile P_1 (and also in profile P_2), so the overall result $(d_1, \dots, d_n)$ will be the same as the result obtained by the sequential procedure in profile P_1 and in profile P_2 . On the other hand, if one of the local rules, say r_i , is not consistent, then the result of the sequential procedure cannot be the same.

5.6 Participation

Theorem 5 *If the sequential voting procedure is participative, then each local voting rule is so.*

Let us assume that there is a local rule, say r_i that is not participative. This means that there is a profile over the values of variable v_i , p_i and an agent h with preference $p_{i,h}$ over the values of v_i , such that agent h strictly prefers $r_i(p_i)$ to $r_i(p_i \cup p_{i,h})$.

Let us now consider the soft profile (V, D, P) , defined as follows:

- for each agent $j = 1, \dots, m$, the preferences over the values of v_i are as in p_i ;
- all other unary constraints are the same for all agents and associate preference 1 to exactly one value per variable and 0 to all other variables. There are no other constraints.

Let us assume that the SCSP P_h of agent h is defined as follows:

- his preference over variable v_i is $p_{i,h}$,
- all other unary constraints associate preference 1 to exactly one value per variable and 0 to all other variables. There are no other constraints.

Note that for each agent j , including h , his preference for any assignment corresponds with his preference on the value assigned to the first variable. Since we have that $\text{Seq}_{O,R}(V, D, P) \downarrow v_i = r_i(p_i)$ and $\text{Seq}_{O,R}(V, D, (P \cup P_h)) \downarrow v_i = r_i(p_i \cup p_{i,h})$, agent h strictly prefers $\text{Seq}_{O,R}(V, D, P)$ to $\text{Seq}_{O,R}(V, D, (P \cup P_h))$. Therefore the sequential voting procedure is not participative.

On the other hand, it is possible that all local voting rules are participative, but the sequential voting procedure is not so.

Theorem 6 *If all the local voting rules are participative, the sequential voting procedure may not be participative.*

To see this, consider the profile where $V = \{x, y\}$, $D = (\{a, b, c\}, \{a, b\})$, and P is a sequence of two fuzzy SCSPs which coincide and contain a unary constraint on x (associating preference 1 to a , 0.8 to b , and 0.6 to c), a binary constraint on x and y (associating preference 1 to (a, b) , 0.9 to (a, a) , 0.8 to (b, a) , 0.7 to (b, b) , 0.6 to (c, a) , and 0.5 to (c, b)), and a unary constraint over y (associating preference 1 to both a and b). It is easy to see that this SCSPs are DAC. Assume also that variables are ordered $x \prec_O y$ and that r_1 is the scoring rule with score vector $(3, 2, 0)$ and r_2 is the majority rule. In this profile, $Seq_{O,R}(V, D, P) = (x = a, y = b)$. We now consider a third voter, with a fuzzy SCSP with a unary constraint on x (associating preference 0.8 to a , 1 to b , and 0.9 to c), a binary constraint on x and y (associating preference 0.8 to (a, b) , 0.5 to (a, a) , 0.7 to (b, a) , 1 to (b, b) , 0.6 to (c, a) , and 0.5 to (c, b)), and a unary constraint over y (associating preference 1 to both a and b). In this new profile P' , $Seq_{O,R}(V, D, P') = (x = b, y = a)$. However, the third voter prefers $(x = a, y = b)$ to $(x = b, y = a)$. Thus the third voter would be better off not participating to the sequential voting process.

5.7 Efficiency

Theorem 7 *If the sequential voting procedure is efficient, then each local voting rule is so.*

Let us assume that there is a local rule, say r_i that is not efficient. This means that there is a profile over the values of variable v_i , p_i , such that $r_i(p_i) = d$ but all agents prefer another value d' .

Let us now consider the soft profile (V, D, P) , defined as follows:

- for each agent j the preferences over the values of v_i are as in p_i ;
- all other unary constraints are the same for all agents and associate preference 1 to exactly one value per variable and 0 to all other variables. There are no other constraints.

We will have that $Seq_{O,R}(V, D, P) = (d_1, d_2, \dots, d_{i-1}, d, \dots, d_n)$. For each agent j his preference for $Seq_{O,R}(V, D, P)$ corresponds with his local preference for d . Thus each agent will prefer $(d_1, d_2, \dots, d_{i-1}, d', \dots, d_n)$ to $Seq_{O,R}(V, D, P)$.

On the other hand, it is possible that all local voting rules are efficient, but the sequential voting procedure is not so.

In fact, let us consider the case where we have two variables X and Y with domain $D(X) = \{a, b\}$ and $D(Y) = \{c, d, e\}$ and three voters. The SCSP of each voter has a single binary constraint on X and Y where tuples $(X = a, Y = c)$, $(X = a, Y = d)$, $(X = a, Y = e)$, $(X = b, Y = c)$, $(X = b, Y = d)$ and $(X = b, Y = e)$ are associated respectively to preferences:

- 1, 0.2, 0.2, 1, 0.2, 0.2 by the first voter;
- 0.2, 1, 0.2, 1, 0.2, 0.2 by the second voter;
- 0.2, 0.2, 1, 1, 0.2, 0.2 by the third voter.

Assume we have that $X < Y$ in the variable ordering, that we are using plurality for all the issues, and that ties are broken lexicographically. Then, when X is considered all voters will vote for a . However, there is no complete assignment with $X = a$ that is optimal for all voters, while $(X = b, Y = c)$ is optimal for all voters.

However, if we add the condition that there is a single candidate which is optimal for all agents, then the sequential voting procedure is efficient as well.

Theorem 8 *If all the local voting rules are efficient, and there is a single candidate which is strictly preferred to all other candidates for all voters, then the sequential voting procedure is efficient.*

In fact, if there is a single candidate, say $d = (d_1, \dots, d_n)$ that is optimal for all agents, then we have that, after DAC, for each agent i and for each variable j , $top(v_j, P_i|_{v_1=d_1, \dots, v_{j-1}=d_{j-1}}) = d_j$. Thus since each rule is efficient it will elect the value of d assigned to its variable. Thus $Seq_{O,R}(V, D, P) = d$.

We note that for profiles where there is a unique candidate that is optimal for all agents, efficiency coincides with Condorcet consistency. Thus, given such profiles, the Condorcet consistency of the local rules implies the Condorcet consistency of the sequential rule.

6 Conclusions and Future Work

We have proposed a sequential preference aggregation procedure for settings where several agents express their preferences over a common set of variable assignments via soft constraints. We have then studied some properties of this procedure. These properties are similar to those obtained in a setting where agents use CP-nets. However, soft constraints allow us to impose no restrictions on the profiles to obtain such properties, while CP-net profiles need to be severely restricted.

We just considered a few properties of the sequential voting procedure. We plan to study many others in order to better characterize the result of the procedure in terms of the preferences of the agents. We also plan to develop heuristics for an efficient computing of such a result even when the soft constraint problems are not from a tractable class.

Finally we will conduct an experimental study for comparing the performance of sequential vs. classical preference aggregation when a single rule is used for all the variables.

7 Acknowledgments

Research partially supported by the Italian MIUR PRIN project 20089M932N: "Innovative and multi-disciplinary approaches for constraint and preference reasoning".

REFERENCES

- [1] K. J. Arrow and A. K. Sen and K. Suzumara, *Handbook of Social Choice and Welfare.*, North-Holland, Elsevier, 2002.
- [2] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole, 'CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements', *J. Artif. Intell. Res. (JAIR)*, **21**, 135–191, (2004).
- [3] R. Dechter, 'Tractable structures for CSPs', in *Handbook of Constraint Programming*, eds., P. Van Beek F. Rossi and T. Walsh, Elsevier, (2005).
- [4] Rina Dechter, *Constraint Processing*, Morgan Kaufmann, 2003.
- [5] Christophe Gonzales, Patrice Perny, and Sergio Queiroz, 'Preference aggregation with graphical utility models', in *AAAI*, pp. 1037–1042. AAAI Press, (2008).
- [6] Jerome Lang and Lirong Xia, 'Sequential composition of voting rules in multi-issue domains', *Mathematical social sciences*, **57**, 304–324, (2009).
- [7] P. Meseguer, F. Rossi, and T. Schiex, 'Soft constraints', in *Handbook of Constraint Programming*, eds., P. Van Beek F. Rossi and T. Walsh, Elsevier, (2005).
- [8] Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh, 'Incompleteness and incomparability in preference aggregation', in *IJCAI*, (2007).
- [9] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of Constraint Programming*, Elsevier, 2006.

Local search for stable marriage problems with ties and incomplete lists

Mirco Gelain¹, Maria Silvia Pini¹, Francesca Rossi¹, Kristen Brent Venable¹
and Toby Walsh²

Abstract. The stable marriage problem has a wide variety of practical applications, ranging from matching resident doctors to hospitals, to matching students to schools, or more generally to any two-sided market. We consider a useful variation of the stable marriage problem, where the men and women express their preferences in the form of a preference list with ties over a subset of the members of the other sex. Matchings are permitted only with people who appear in these preference lists. In this setting, we study the problem of finding a stable matching that marries as many people as possible. Stability is an envy-free notion: no man and woman who are not married to each other would both prefer each other to their partners or to being single. This problem is NP-hard. We propose to tackle it via a local search approach, which exploits properties of the problem to reduce the size of the neighborhood and to make local moves efficiently. Experimental results show that this approach is able to solve large problems, quickly returning stable matchings of large and often optimal size.

1 Introduction

The stable marriage problem [3] is a well-known problem of matching men to women to achieve a certain type of “stability”. Each person expresses a strict preference ordering over the members of the opposite sex. The goal is to match men to women so that there are no two people of opposite sex who would both rather be matched with each other than with their current partners. Surprisingly such a stable marriage always exists and one can be found in polynomial time. Gale and Shapley give a quadratic time algorithm to solve this problem based on a series of proposals of the men to the women (or vice versa) [1]. The stable marriage problem has a wide variety of practical applications, ranging from matching resident doctors to hospitals, sailors to ships, primary school students to secondary schools, as well as in market trading.

There are many variants of this classical formulation of the stable marriage problem. Some of the most useful in practice include incomplete preference lists (SMI), that allow us to model unacceptability for certain members of the other sex, and preference lists with ties (SMT), that model indifference in the preference ordering. With a SMI problem, we have to find a stable marriage in which the married people accept each other. It is known that all solutions of a SMI problem

have the same size (that is, number of married people). In SMT problems, instead, solutions are stable marriages where everybody is married. Both of these variants are polynomial to solve. In real world situations, both ties and incomplete preference lists may be needed. Unfortunately, when we allow both, the problem becomes NP-hard [7]. In a SMTI (Stable Marriage with Ties and Incomplete lists) problem, there may be several stable marriages of different sizes, and solving the problem means finding a stable marriage of maximum size.

In this paper we investigate the use of a local search approach to tackle this problem. Our algorithm starts from a randomly chosen marriage and, at each step, moves to a neighbor marriage which is obtained by removing one blocking pair, that is, a man-woman pair who are not married in the current marriage but who prefer to be married with each other rather than with their current partners. Stable marriages have no blocking pairs, so the aim of such a move is to pass to a marriage which is closer to stability. Among the neighbor marriages, the evaluation function chooses one with the smallest number of blocking pairs and of singles. Since there may be several stable marriages with different sizes, we look for the one with maximum size (that is, the smallest number of singles). Random moves are also used, to avoid stagnation in local minima. The algorithm stops when a perfect matching (that is, a stable marriage with no singles) is found, or when a given limit on the number of steps is reached.

This basic local search approach works well with problems of limited size, but does not scale. With large sizes, it fails to find good solutions and sometimes even stable marriages. One of the main reasons is that the neighborhood can be very large, since a marriage may have a large number of blocking pairs. Many such blocking pairs can be ignored since they are “dominated” by others, whose removal will also eliminate all the dominated blocking pairs. By considering only undominated blocking pairs, we can solve SMTI problems of much larger size in a small amount of time. The marriages returned by our local search method are stable and contain very few single people. Experiments on randomly generated SMTI problems of size 100 show that our algorithm is able to find stable marriages with at most two singles on average in tens of seconds at worst.

The SMTI problem has been tackled also in [2], where the problem is modelled in terms of a constraint optimization problem and a constraint solver is employed to solve it. This systematic approach is guaranteed to find always an optimal solution. However, our experimental results show that our lo-

¹ University of Padova

² NICTA and UNSW

cal search algorithm in practice always finds optimal solutions. Moreover, it scales well to sizes much larger than those considered in [2]. Another approach is to use approximation. Unfortunately the SMTI problem cannot be approximated within $33/29$ unless $P=NP$ [11]. On the other hand, McDermid recently gave a three phase algorithm for the SMTI problem that has an approximation ratio of $3/2$ [8] which beats the previous best known guarantee of $5/3$.

2 Background

In this section we give the basic notions about stable marriage problems and local search.

2.1 Stable marriage problems with ties and incompleteness

A stable marriage (SM) problem [3] consists of matching members of two different sets, usually called men and women. When there are n men and n women, the SM problem is said to have size n . Each person strictly ranks all members of the opposite sex. The goal is to match the men with the women so that there are no two people of opposite sex who would both rather marry each other than their current partners. Such a marriage is called *stable*. At least one stable marriage exists for every SM problem. In fact, the set of stable marriages forms a lattice. Gale and Shapley give a polynomial time algorithm to find the stable marriage at the top (or bottom) of this lattice [1].

In this paper we consider a variant of the SM problem where preference lists may include ties and may be incomplete. This variant is denoted by SMTI [6]. Ties express indifference in the preference ordering, while incompleteness models unacceptability for certain partners.

Definition 1 (SMTI marriage) *Given a SMTI problem with n men and n women, a marriage M is a one-to-one matching between men and women such that partners accept each other. If a man m and a woman w are matched in M , we write $M(m) = w$ and $M(w) = m$. If a person p is not matched in M we say that he/she is single.*

Definition 2 (Marriage size) *Given a SMTI problem of size n and a marriage M , its size is the number of men (or women) that are married.*

An example of a SMTI problem with four men and women is shown in Table 1. A SMTI problem is described by giving, for each man and woman, the corresponding preference list over members of the other sex. For example, by writing $2 : 2 (3\ 4)$ among the men’s preference lists we mean that man m_2 strictly prefers woman w_2 to women w_3 and w_4 , that are equally preferred.

Definition 3 (blocking pairs in SMTIs) *Consider a SMTI problem P , a marriage M for P , a man m and a woman w . A pair (m, w) is a blocking pair in M if m and w accept each other and m is either single in M or he strictly prefers w to $M(m)$, and w is either single in M or she strictly prefers m to $M(w)$.*

men’s preference lists	women’s preference lists
1: 2 1	1: 3 1 (2 4)
2: 2 (3 4)	2: 1 4 2
3: (1 2 3 4)	3: (1 2) (4 3)
4: (3 2) 1 4	4: (3 2 4)

Table 1. An example of a SMTI problem of size 4.

Definition 4 (Weakly Stable Marriages) *Given a SMTI problem P , a marriage M for P is weakly stable if it has no blocking pairs.*

As we will consider only the weak stability criterion, we will simply call them stable marriages. Given a SMTI problem, there may be several stable marriages of different sizes. If the size of a marriage coincides with the size of the problem, it is said to be a perfect matching.

In the above example, the marriage $2\ 3\ 1\ 4$ (where the number in position i indicates the woman married to man m_i in that marriage) is stable and its size is 4, so it is a perfect matching.

Solving a SMTI problem means finding a stable marriage with maximal size. This problem is NP-hard [7].

2.2 Local search

Local search [4, 10] is one of the fundamental paradigms for solving computationally hard combinatorial problems. Local search methods in many cases represent the only feasible way for solving large and complex instances. Moreover, they can naturally be used to solve optimization problems.

Given a problem instance, the basic idea underlying local search is to start from an initial search position in the space of all solutions (typically a randomly or heuristically generated candidate solution, which may be infeasible, sub-optimal or incomplete), and to improve iteratively this candidate solution by means of typically minor modifications. At each *search step* we move to a position selected from a *local neighborhood*, chosen via a heuristic evaluation function. The evaluation function typically maps the current candidate solution to a real number and it is such that its global minima correspond to solutions of the given problem instance. The algorithm moves to the neighbor with the smallest value of the evaluation function.

This process is iterated until a *termination criterion* is satisfied. The termination criterion is usually the fact that a solution is found or that a predetermined number of steps is reached, although other variants may stop the search after a predefined amount of time.

Different local search methods vary in the definition of the neighborhood and of the evaluation function, as well as in the way in which situations are handled when no improvement is possible. To ensure that the search process does not stagnate in unsatisfactory candidate solutions, most local search methods use randomization: at every step, with a certain probability a random move is performed rather than the usual move to the best neighbor.

3 Local search on SMTIs

We adapt the classical local search schema to SMTI problems as follows. Given a SMTI problem P , we start from a randomly generated marriage M for P . At each search step, we move to a new marriage in the neighborhood of the current one. For each marriage M , the neighborhood $N(M)$ is the set of all marriages obtained by removing one blocking pair from M . Consider a blocking pair $bp = (m, w)$ in M and assume $m' = M(w)$ and $w' = M(m)$. Then, removing bp from M (written $M \setminus bp$) means obtaining a marriage M' in which m is married with w and both m' and w' become single, leaving the other pairs in the marriage M unchanged. Notice that, if M is stable, its neighborhood is empty. Notice also that this notion of neighborhood is not symmetric.

To select the neighbor to move to, we use an evaluation function $f : \mathcal{M}_n \rightarrow \mathbb{Z}$, where $\mathcal{M}_n$ is the set of all possible marriages of size n , and $f(M) = nbp(M) + ns(M)$. For each marriage M , $nbp(M)$ is the number of blocking pairs in M , while $ns(M)$ is the number of singles in M which are not in any blocking pair. The algorithm moves to a marriage $M' \in N(M)$ such that $f(M') \leq f(M'') \forall M'' \in N(M)$.

During the search, the algorithm maintains the best marriage found so far, defined as follows: if no stable marriage has been found, then the best marriage is the one with the smallest value of the evaluation function; otherwise, it is the stable marriage with fewest singles.

To avoid stagnation in a local minimum of the evaluation function, at each search step we perform a random walk with probability p (where p is a parameter of the algorithm). In the random walk, we move to a randomly selected marriage in the neighborhood. If a stable marriage is reached, its neighborhood is empty and a random restart is performed.

The algorithm terminates if a perfect marriage (that is, a stable marriage with no singles) is found, or when a maximal number of search steps is reached. Upon termination, the algorithm returns the best marriage found during the search.

The pseudo-code of our algorithm, called LTI, is shown in Algorithm 1. In the pseudo-code, M_{best} is the best marriage found so far, and f_{best} its evaluation (number of blocking pairs plus number of singles). Function $best_neighborhood$ returns one of the best marriages in the neighborhood of the current marriage, according to the evaluation function.

In addition to this simple local search algorithm which directly applies standard local search approaches to SMTI problems, we have also designed a more sophisticated algorithm which has been tailored to exploit the specific features of SMTI problems. The main difference is in the definition of the neighborhood, which refers to the notion of *undominated* blocking pairs.

Definition 5 (dominance in blocking pairs) Let (m, w) and (m, w') be two blocking pairs. Then (m, w) dominates (from the men's point of view) (m, w') if m prefers w to w' . There is an equivalent concept from the women's point of view.

Definition 6 (undominated blocking pair) A *men-* (resp., *women-*) undominated blocking pair is a blocking pair such that there is no other blocking pair that dominates it from the men's (resp., women's) point of view. When the point of view (men or women) is clear or not important, we will omit it.

Algorithm 1: LTI

input : a SMTI problem P , an integer max_steps
output: a marriage

```

1  $M \leftarrow$  random marriage
2  $steps \leftarrow 0$ 
3  $M_{best} \leftarrow M$ 
4  $f_{best} \leftarrow f(M)$ 
5 repeat
6   if  $f(M) = 0$  then
7     return  $M$ 
8   if  $rand() \leq p$  then
9      $M \leftarrow RandomWalk(M)$ 
10  else
11     $PAIRS \leftarrow$  blocking pairs in  $M$ 
12    if  $PAIRS$  is empty then
13      perform a random restart
14    else
15       $M \leftarrow best\_neighborhood(M, PAIRS)$ 
16  if  $M$  is the first stable marriage found so far then
17     $f_{best} \leftarrow f(M)$ 
18     $M_{best} \leftarrow M$ 
19  if  $M_{best}$  is not stable and  $f_{best} > f(M)$  then
20     $f_{best} \leftarrow f(M)$ 
21     $M_{best} \leftarrow M$ 
22  if both  $M_{best}$  and  $M$  are stable and  $f_{best} > f(M)$  then
23     $f_{best} \leftarrow f(M)$ 
24     $M_{best} \leftarrow M$ 
25   $steps \leftarrow steps + 1$ 
26 until  $steps \leq max\_steps$  ;
27 return  $M_{best}$ 

```

For example, consider the SMTI problem in Table 1, the marriage 1 2 3 4, and two blocking pairs (m_1, w_2) and (m_4, w_2) . Using the definitions above, (m_1, w_2) dominates (m_4, w_2) from the women's point of view. If we remove (m_4, w_2) from the marriage, (m_1, w_2) will remain. On the other hand, removing (m_1, w_2) also eliminates (m_4, w_2) . Thus removing undominated blocking pairs may reduce the number of blocking pairs more than eliminating dominated pairs.

We call LTIU the algorithm LTI where the neighborhood is defined as the set of marriages obtained from the current one by removing any undominated blocking pair. More precisely, at each step we consider the undominated blocking pairs from the men's point of view which are also undominated from the women's point of view. Then, in the next step, we swap genders and do the same, to ensure gender neutrality in our algorithm.

Due to their ability to restart, our algorithms have the PAC (probabilistically approximate complete property) [5]. That is, as their runtime goes to infinity, the probability that the algorithm does not return an optimal solution goes to zero. If the algorithm starts at a stable marriage, the algorithms will perform a random restart, which will end up in an optimal solution with probability greater than zero. On the other hand, if the algorithm starts from a non-stable marriage, we perform one or more steps in which we remove a blocking

pair. This sequence of blocking pair removal has been shown to converge to a stable marriage with non-zero probability in the context of SMs with incomplete preference lists [9]. The proof of this result can be adapted to our context, although we have ties in the preference lists. Since a stable marriage can be reached with non-zero probability, and as we have argued above that from any stable marriage random restarting will reach an optimal solution with non-zero probability, the PAC property holds.

4 Experimental setting

Problems are generated using the same method as in [2]. The generator takes three parameters: the problem's size n , the probability of incompleteness p_1 and the probability of ties p_2 . Given a triple (n, p_1, p_2) , a SMTI problem with n men and n women is generated, as follows:

1. For each man and woman, we generate a random preference list of size n , i.e., a permutation of n persons;
2. We then iterate over each man's preference list: for a man m_i and for each woman w_j in his preference list, with probability p_1 we delete w_j from m_i 's preference list and delete m_i from w_j 's preference list. In this way we get a possibly incomplete preference list.
3. If any man or woman has an empty preference list, we discard the problem and go to step 1.
4. We iterate over each person's (men and women's) preference list as follows: for a man m_i and for each woman in his preference list, in position $j \geq 2$, with probability p_2 we set the preference for that woman as the preference for the woman in position $j - 1$ (thus putting the two women in a tie).

Note that this method generates SMTI problems in which the acceptance is symmetric. In fact, if a man m does not accept a woman w , m is removed from w 's preference list as well. This does not introduce any loss of generality because m and w cannot be matched together in any stable marriage.

Notice also that this generator will not construct a SMTI problem in which a man (resp., woman) accepts only women (resp., men) who do not find him (resp, her) acceptable. Such a man (resp., woman) will remain single in every stable matching. Therefore a simple preprocessing step can remove such men and women, giving a smaller problem of the form constructed by our generator.

We generated random SMTI problems of size 100, by letting p_2 vary in $[0, 1.0]$ with step 0.1, and p_1 vary in $[0.1, 0.8]$ with step 0.1 (above 0.8 the preference lists start to be empty). For each parameter combination, we generated 100 problem instances. Moreover, the probability of the random walk is set to $p=20\%$ and the search step limit is $s=50000$.

4.1 Experimental results

We first analyzed the behavior of the base algorithm, LTI. Unfortunately this algorithm fails to find a stable marriage in most of our test problems (see Figure 1). In fact, LTI can find a stable marriage only for easy problems, which are the ones with a large amount of ties (that is, p_2 high) and/or with a large amount of incompleteness (that is, p_1 high).

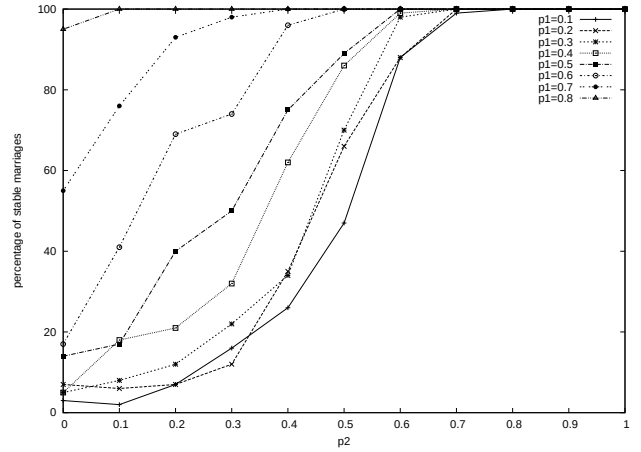


Figure 1. Average number of stable marriages found by LTI.

On the other hand, algorithm LTIU finds a stable marriage in 100% of the runs. Since stability is essential in our context, from now on we will only show the experimental results for algorithm LTIU.

We start by showing the average size of the marriages returned by LTIU. In Figure 2 we can see that LTIU finds a perfect marriage (that is, a stable marriage with no singles) almost always. Even in settings with a large amount of incompleteness (that is, $p_1 = 0.7 - 0.8$) the algorithm finds very large marriages, with only 2 singles in average.

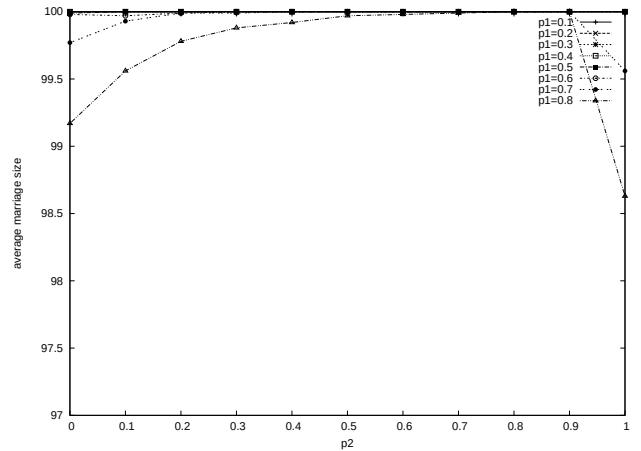


Figure 2. Average size of marriages with LTIU, varying p_2 for different values of p_1 .

We also consider the number of steps needed by our algorithm. From Figure 3, we can see that the number of steps is less than 2000 most of the time, except for problems with a large amount of incompleteness (i.e. $p_1 = 0.8$). As expected, with p_1 greater than 0.6 the algorithm requires more steps. In some cases, it reaches the step limit of 50000. Moreover, as the percentage of ties rises, stability becomes easier to achieve

and thus the number of steps tends to decrease slightly. From the results we can see that the case with complete indifference ($p_2=1$) is a special case. In fact, in this situation, the number of steps increases for almost every value of p_1 . This is because the algorithm makes most of its progress via random restarts. In these problems every person (if accepted) is equally preferred to all the others accepted. This means that the only blocking pairs are those involving singles who both accept each other. In this situation, after a few steps all singles that can be married are matched, stability is reached, and the neighborhood becomes empty. The algorithm therefore performs another random restart. In this situation it is very difficult to reach a perfect matching and the algorithm therefore often runs until the step limit.

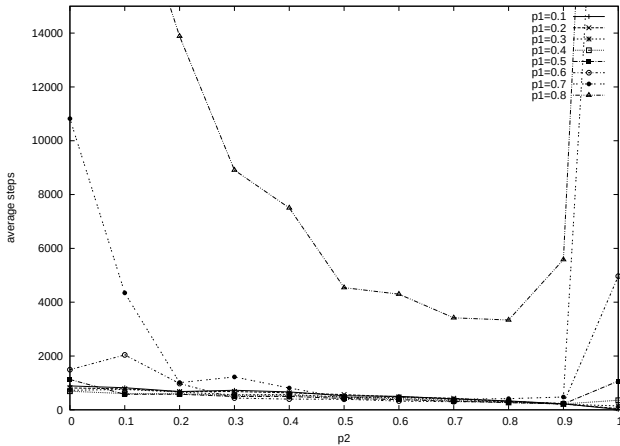


Figure 3. Average number of steps for LTIU, varying p_2 for different values of p_1 .

The algorithm is fast. It takes, on average, less than 40 seconds to give a result even for very difficult problems (see Figure 4). As expected, with $p_2 = 1$ the time increases for the same reason discussed above concerning the number of steps.

Re-considering Figure 2 and the fact that all the marriages the algorithm finds are stable, we notice that most of the marriages are perfect. From Figure 5 we see that the average percentage of matchings that are perfect is almost always 100% and this percentage only decreases when the incompleteness is large.

We compared our local search approach to the one in [2]. In their experiments, they measured the maximum size of the stable marriages in problems of size 10, fixing p_1 to 0.5 and varying p_2 in $[0,1]$. We did similar experiments, and obtained stable marriages of a very similar size to those reported in [2]. This means that although our algorithm is incomplete in principle, it always finds an optimal solution in practice, and for small sizes it behaves as a complete algorithm in terms of size of the returned marriage. However, we can also tackle problems of much larger sizes, still obtaining optimal solutions most of the time.

We also considered the runtime behavior of our algorithm. In Figure 6 we show the average normalized number of blocking pairs and, in Figure 7, the average normalized number of singles of the best marriage as the execution proceeds. Al-

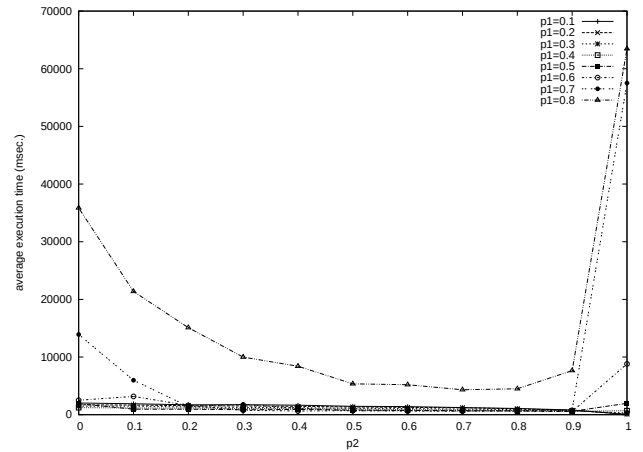


Figure 4. Average execution time for LTIU, varying p_2 for different values of p_1 .

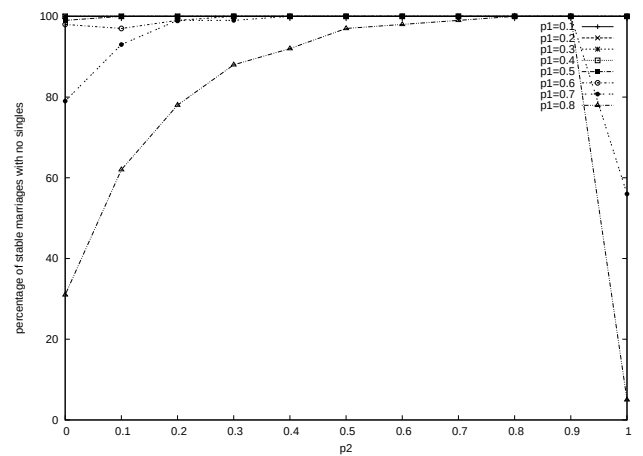


Figure 5. Percentage of perfect matchings, varying p_2 for different values of p_1 .

though the step limit is 50000, we only plot results for the first steps because the rest is a long plateau that is not very interesting. We show the results only for $p_2 = 0.5$. However, for greater (resp., lower) number of ties the curves are shifted slightly down (resp., up). From Figure 6 we can see that the average number of blocking pairs decreases very fast, reaching 5 blocking pairs after only 100 steps. Then, after 300-400 steps, we reach a stable marriage almost all the time for all values of p_1 . Considering Figure 7, we can see that the algorithm starts with more singles for greater values of p_1 . This happens because, with more incompleteness, it is more difficult for a person to be accepted. However, after 200 steps, the average number of singles becomes very small no matter the incompleteness in the problem.

Looking at both Figures 6 and 7, we observe that, although we set a step limit $s = 50000$, the algorithm reaches a very good solution after just 300-400 steps. In fact, after this number of steps, the best marriage found by the algorithm usually

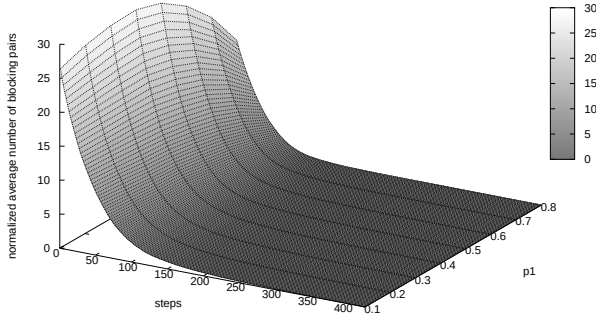


Figure 6. Average normalized number of blocking pairs ($p_2=0.5$).

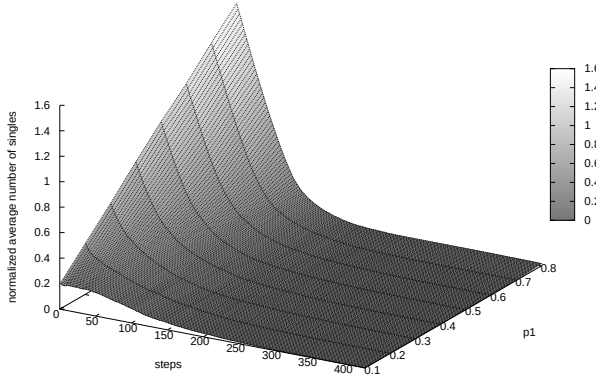


Figure 7. Average normalized number of singles ($p_2=0.5$).

has no blocking pairs nor singles. This appears largely independent of the amount of incompleteness and the number of ties in the problems. Hence, for SMTI problems of size 100 we could set the step limit to just 400 steps and still be reasonably sure that the algorithm will return a stable marriage with a large size, no matter the amount of incompleteness and ties.

5 Conclusions

We have presented a local search approach for solving stable marriage problems with ties and indifference. Experimental results show that our algorithm is both fast and effective at finding large stable marriages for problems of sizes not considered before in the literature. Moreover, the runtime behavior of the algorithms is not greatly influenced by the amount of incompleteness or ties in the problem. The algorithm was

usually able to obtain a very good solution after a very small amount of time.

ACKNOWLEDGEMENTS

Research partially supported by the Italian MIUR PRIN project 20089M932N: "Innovative and multi-disciplinary approaches for constraint and preference reasoning".

REFERENCES

- [1] D. Gale and L. S. Shapley, 'College admissions and the stability of marriage', *The American Mathematical Monthly*, **69**(1), 9–15, (1962).
- [2] Ian P. Gent and Patrick Prosser, 'An empirical study of the stable marriage problem with ties and incomplete lists', in *ECAI*, pp. 141–145, (2002).
- [3] D. Gusfield and R. W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Boston, MA, 1989.
- [4] Edward Tsang Holger H. Hoos, 'Local search methods', in *Handbook of Constraint Programming*, eds., F. Rossi, P. Van Beek, and T. Walsh, Elsevier, (2006).
- [5] H. Hoos, 'On the run-time behaviour of stochastic local search algorithms for sat', in *Proc. AAAI'99*, pp. 661–666, (1999).
- [6] Kazuo Iwama, David Manlove, Shuichi Miyazaki, and Yasufumi Morita, 'Stable marriage with incomplete lists and ties', in *ICALP*, pp. 443–452, (1999).
- [7] David Manlove, Robert W. Irving, Kazuo Iwama, Shuichi Miyazaki, and Yasufumi Morita, 'Hard variants of stable marriage', *Theor. Comput. Sci.*, **276**(1-2), 261–279, (2002).
- [8] E. McDermid, 'A 3/2-approximation algorithm for general stable marriage', in *Proc. 36th ICALP*, pp. 689–700, (2009).
- [9] Alvin E. Roth and John H. Vande Vate, 'Random paths to stability in two-sided matching', *Econometrica*, **58**(6), 1475–1480, (1990).
- [10] Thomas G. Stutzle, *Local Search Algorithms for Combinatorial Problems - Analysis, Improvements, and New Applications*, Ph.D. dissertation, Am Fachbereich Informatik der Technischen Universität Darmstadt, 1998.
- [11] Yanagisawa, *Approximation algorithms for stable marriage problems*, Ph.D. dissertation, Kyoto University, Graduate School of Informatics, 2007.

Dynamic House Allocation

Sujit Gujar¹ and James Zou² and David C. Parkes³

Abstract. We study a dynamic variant on the house allocation problem. Each agent owns a distinct object (a house) and is able to trade its house while present in the market. Agents have strict preferences over houses, and the market operates without payments. The goal is to enable an efficient reallocation of objects, along with strategyproofness and while satisfying participation constraints. We first establish conditions under which an online mechanism that allows an agent to influence the period in which it trades can be manipulated. This motivates *partition mechanisms* in which agents are divided online into disjoint feasible trading groups, with the *top trading cycle algorithm* (TTCA) run separately for each group. In particular, we demonstrate good rank-efficiency for a mechanism that adopts stochastic-optimization in determining how to partition agents.

1 Introduction

In the *house allocation problem*, each of a set of self-interested agents owns a distinct object (a house) and has strict preferences on houses [10]. The problem is to find a reallocation of objects amongst agents that is robust against misreports of preferences by agents while identifying beneficial trades and without using money. The *top-trading cycle algorithm* (TTCA) is *strategyproof* (meaning that truthful reports of preferences is a dominant strategy for agents) and finds an allocation in the *core*. An allocation in the core is stable, in the sense that no coalition of agents can *block* the outcome by reallocating their initial objects amongst themselves in a way that is weakly better for every agent and strictly better for at least one agent in the coalition. The TTCA is known to be essentially unique amongst mechanisms for the static house allocation problem with useful economic properties [8, 6].

In the dynamic model of the house allocation problem advanced here, each agent has an arrival period and a departure period and is only able to trade with other agents present simultaneously present in the market at the same time. For a motivating example, consider college housing, with students on different leases and willing to trade during the month before their lease expires. Another example is provided by the problem of reallocating offices in the workplace, or finding mutually-beneficial trades (e.g., swaps, three-cycles, etc.) of vacation property such as time-shares when inventory is in the market at different times.

Contributions. We establish general conditions under which no mechanism in which an agent can influence the set of agents with which it participates in a TTCA (e.g., the period in which it trades and thus the other agents that it trades with) can be strategyproof. Given this, we study *partition mechanisms*, in which each agent is

assigned online to a group of agents with which it will engage in a single TTCA.

We consider three variations of partition mechanisms:

- (i) trade on departure (*DO-TTCA*)
- (ii) trade on departure or when the population size is above some threshold (*T-TTCA*)
- (iii) a stochastic optimization approach to determine a good partitioning of agents (*SO-TTCA*).

We adopt *rank efficiency* as a measure of performance, which is defined as the preference rank of agents for allocated objects, averaged across all agents and across instances sampled from a distribution. This is a meaningful measure of performance for risk neutral agents, each of which has an equal difference in utility for successive houses in its preference list.

The experimental results show that SO-TTCA and T-TTCA outperform DO-TTCA by 15-20% in terms of rank efficiency when there are large number of agents and for uniform arrival time and patience distributions. Furthermore, when the agents arrive according to a Poisson process, then SO-TTCA outperforms DO-TTCA by 4-10% in terms of rank efficiency. For environments in which agents tend to be impatient, T-TTCA has performance competitive with that of SO-TTCA. But T-TTCA is less successful than SO-TTCA in identifying partitions in which each agent tends to have a large number of trading partners, which is useful when agents are patient.

Related Work. For an overview of mechanisms on matching and house allocation, the interested reader is referred to Sönmez and Ünver [11]. In previous models of multi-period house allocation problems, each agent expresses a preference on a sequence of allocations [1, 5]. Our model is different because an agent's preferences are expressed only over its eventual allocation upon departure and not on the sequence of allocations made while present in the market. Another difference is that our agents bring a house to the market upon arrival and depart with a house, while previous models maintain the same pool of objects throughout and agents arrive and leave empty-handed. Stochastic models of kidney exchanges, in which trades are identified amongst donor-patient pairs, are explored but without incentive considerations [2, 12]. For other dynamic models, there is existing work on matching (with strict preferences expressed in a bipartite graph) [4], assignment (which is different from house allocation because agents do not own an object upon arrival) [13], as well as a considerable literature on dynamic mechanisms with money (see Parkes [7] for a survey). For static problems, there is also computational work on the reallocation of objects through different types of local trades (e.g., swaps, 3-cycles, etc.), with a view to finding desirable outcomes such as efficient or fair allocations [9, 3].

2 The Model

Let $N = \{A_1, A_2, \dots, A_n\}$ denote the set of agents and $H = \{h_1, h_2, \dots, h_n\}$ denote the set of distinct houses. Agent A_i enters

¹ Indian Institute of Science. email: sujit@csa.iisc.ernet.in

² School of Engineering and Applied Sciences, Harvard University. email: jzou@fas.harvard.edu

³ School of Engineering and Applied Sciences, Harvard University. email: parkes@eecs.harvard.edu

the market with house h_i in period $a_i \in T$, where $T = \{1, 2, \dots\}$, and departs in $d_i \in T$. Let $Sched_N$ denote the set of all possible arrival and departure times of N agents. Agent A_i has strict preferences $\succ_i$ on the house allocated upon departure. Let $(h \succ_i h') \equiv (h' \not\succ_i h)$. The preference profile of all the agents is denoted by $\succ = (\succ_1, \succ_2, \dots, \succ_n) \in U$, where U is the set of all possible strict preference profiles. Let $\succ_{-i}$ denote the preference profile of agents except A_i . An agent knows its preferences $\succ_i$ upon arrival into the market. We allow for arbitrary misreports of preferences, but assume that arrival and departure times are truthfully reported.

Let $x : N \rightarrow H$ denote a house allocation, with agent A_i allocated to house $x(i)$. Let $X(\rho)$ denote the set of *feasible allocations* given arrival-departure schedule $\rho \in Sched_N$. An allocation is feasible for a given schedule ρ if there is a sequence of trades, one in each period, where the trade in any given period is only between those agents present and is feasible given the allocation determined through trade in the previous period, and where each agent allocated to exactly one house in every period.

An *online house allocation mechanism* $\phi(\succ, \rho) \in X(\rho)$, generates a feasible allocation given a reported type profile and arrival-departure schedule. The problem is online, in that an agent's type $(a_i, d_i, \succ_i)$ is not available until period $t = a_i$ and therefore we require, for all $A_j \in N$, that $\phi_j(\succ, \rho) = \phi_j(\succ', \rho')$ when $\succ, \succ'$ and ρ, ρ' differ only in periods after agent A_j 's departure. An assignment of houses, $x \in X$, *Pareto dominates* $y \in X$, if $x \succeq_i y$ for all A_i and $x \succ_j y$ for some A_j . An allocation $y \in X$ is *Pareto efficient* if there is no allocation $x \in X$ that Pareto dominates y .

Definition 2.1 (Pareto efficient). *A mechanism ϕ is Pareto efficient if allocation $x = \phi(\succ, \rho)$ is Pareto efficient for all preference profiles $\succ$ and all schedules ρ .*

Definition 2.2 (Strategyproof (SP)). *Let $x = \phi(\succ_i, \succ_{-i}, \rho)$ and $x' = \phi(\succ'_i, \succ_{-i}, \rho)$. Mechanism ϕ is strategyproof if $x(i) \succ_i x'(i)$, for all A_i , all $\rho \in Sched_N$ and all $\succ_{-i} \in U_{-i}$.*

Definition 2.3 (Individually Rational (IR)). *An online mechanism ϕ is individually rational if $x_i \succ_i h_i$, where $x = \phi(\succ, \rho)$, for all $\succ \in U$, all $\rho \in Sched_N$.*

An allocation x is *blocked* by a coalition of agents $S \subseteq N$, if there is a feasible allocation of the houses initially owned by agents in S amongst themselves that Pareto dominates, for agents in S , the allocation x .

Definition 2.4 (Core). *An online mechanism ϕ is core-selecting if allocation $x = \phi(\succ, \rho)$ is not blocked by any coalition of agents, for any preference profile $\succ$, and any schedule ρ .*

The core implies Pareto efficiency (by considering coalitions of size n) and IR (by considering coalitions of size one).

To obtain a quantitative measure of efficiency, we assume in our experimental analysis that agents are *risk neutral*, and with a uniform difference in utility between successive houses in their preference orders. Based on this, we can compare the expected utility of two mechanisms in terms of *rank-efficiency*. Let $rank_{\phi, i}(\succ, \rho)$ denote the rank that A_i assigns to the house allocated by mechanism ϕ given $\succ$ and ρ , and define

$$RANK_{\phi}(\succ, \rho) = \sum_i rank_{\phi, i}(\succ, \rho)$$

Definition 2.5. *The rank-efficiency of mechanism ϕ ,*

$$rank^{\phi} = \mathbb{E}_{\succ, \rho}[RANK_{\phi}(\succ, \rho)],$$

where the expectation is taken with respect to a distribution on agent preferences and arrival-departure schedules.

Claim 2.1. *No online house allocation mechanism with three or more agents can be Pareto efficient and individually rational (IR).*

Proof. Suppose there exists a online mechanism ϕ , which is IR as well as Pareto efficient. Consider the following two period dynamic house allocation with 3 agents. (Note, A_i owns house h_i .)

A_1	: $h_3 \succ_1 h_1 \succ_1 h_2, a_1 = 1 d_1 = 2$	[-----]
A_2	: $h_1 \succ_2 h_2 \succ_2 h_3, a_2 = 1 d_2 = 1$	[-----]
A_3	: $h_2 \succ_3 h_3 \succ_3 h_1, a_3 = 2 d_3 = 2$	[-----]

In this instance, $A_1 - h_3, A_2 - h_1, A_3 - h_2$ is the only feasible, Pareto efficient and IR allocation. As, A_2 departs in period 1, ϕ should assign h_1 in period 1 only. If it happens that $A_3 : h_3 \succ_3 h_1 \succ_3 h_2$, then the only allocation that is feasible, Pareto efficient and IR is, $A_1 - h_1, A_2 - h_2, A_3 - h_3$. For this allocation, ϕ should retain h_2 with A_2 at $t = 1$. But no online mechanism can correctly decide at $t = 1$ whether to assign h_1 or h_2 to A_2 because the preferences of A_3 are unknown. $\square$

In contrast, a *serial-dictatorship* mechanism, in which agents release ownership of their house upon arrival and receive their most preferred house of those available upon departure (with ties broken at random), is Pareto efficient but not IR. To see that this is Pareto efficient, note that the second agent to be allocated cannot receive a better house without the first agent receiving a worse house. This argument continues inductively. Failure of IR is easy to understand.

Because core is a stronger property than Pareto efficiency and IR we also know from Claim 2.1 that no online mechanism can be core-selecting.

3 Dynamic Top Trading Cycle Mechanisms

In this section, we define the static Top Trading Cycle Algorithm (TTCA) and introduce dynamic generalizations, leading to a result that constrains the use of TTCA for dynamic house allocation problems.

3.1 The Static TTCA

The Top Trading Cycle Algorithm (TTCA) [10] is strategyproof and selects a *core* allocation for the static house allocation problem.

Definition 3.1 (Top Trading Cycle Algorithm). *Every agent points to its most preferred house. There will be at least one cycle, and the agents on any such cycle (including self-loops) receive the house to which they point. These agents are removed from the system. Now each remaining agent points to its most preferred remaining house. The procedure continues till there are no houses left to allocate.*

Example 3.1 (TTCA). *Consider a problem with 5 agents, with agent A_i owning house h_i . Let the preferences of these agents over houses be:*

A_1	: $h_2 \succ_1 h_4 \succ_1 h_3 \succ_1 h_1 \succ_1 h_5$
A_2	: $h_3 \succ_2 h_4 \succ_2 h_5 \succ_2 h_1 \succ_2 h_2$
A_3	: $h_2 \succ_3 h_3 \succ_3 h_1 \succ_3 h_4 \succ_3 h_5$
A_4	: $h_5 \succ_4 h_2 \succ_4 h_3 \succ_4 h_4 \succ_4 h_1$
A_5	: $h_1 \succ_5 h_4 \succ_5 h_2 \succ_5 h_3 \succ_5 h_5$

The agents point to their most preferred house as: $A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_2, A_4 \rightarrow A_5 \rightarrow A_1$. Now, A_2 and A_3 form a cycle, trade and are removed. Now, the agents point to the houses as: $A_1 \rightarrow A_4 \rightarrow A_5 \rightarrow A_1$. This being a cycle, A_1 gets A_4 's house, A_4 gets A_5 's and A_5 gets A_1 's. Thus the final allocation by TTCA is $(A_1 - h_4, A_2 - h_3, A_3 - h_2, A_4 - h_5, A_5 - h_1)$.

We are interested to explore whether or not we can use TTCA as a building block for a family of strategyproof online mechanisms.

3.2 On-line TTCA

We first consider the simplest possible idea, which is to run TTCA in every period in which at least one agent departs and commit to the allocation determined for departing agents. Call this mechanism O-TTCA.

Claim 3.1. *O-TTCA is not strategy proof when there are three or more agents.*

Proof. Consider an example with 3 agents $N = \{A_1, A_2, A_3\}$, with agent A_i owning house h_i , and preferences:

$$\begin{array}{ll} A_1 & : \quad h_3 \succ_1 h_2 \succ_1 h_1 \quad (a_1 = 1, d_1 = 3) \\ & \quad [- \text{-----}] \\ A_2 & : \quad h_1 \succ_2 h_2 \succ_2 h_3 \quad (a_2 = 1, d_2 = 1) \\ & \quad [- \text{-----}] \\ A_3 & : \quad h_1 \succ_3 h_3 \succ_3 h_2 \quad (a_3 = 2, d_3 = 3) \\ & \quad [- \text{-----}] \end{array}$$

If A_1 reports truthfully, at $t = 1$, the $A_1 \leftrightarrow A_2$ trade will occur and there will be no trade in periods $t \in \{2, 3\}$. But A_1 can report $h_3 \succ'_1 h_1 \succ'_1 h_2$. Now no trade occurs in $t = 1$, and at $t = 3$, agents $A_1 \leftrightarrow A_3$ trade and A_1 will receive h_3 which is preferred to h_2 . $\square$

3.3 Precluding Multiple Trades

A *sample path* $\omega = (\succ, \rho)$ is an instance of the dynamic house allocation problem. At each time t , let $\omega(t)$ denote the restriction of $\succ$ and ρ to only those agents with $a_i < t$ and $\omega(t_1, t_2)$ denote the restriction to agents with $a_i \in \{t_1, \dots, t_2\}$. Sample path $\omega(t, t')$ is a *valid continuation* of $\omega(t)$ if $\omega(t, t')$ is an instance of agents and reported preferences arriving in $[t, t']$.

To avoid corner cases, in this section we consider a generalization of the model where there are N classes of equivalent houses; houses in each class are identical. Agents have strict preference over classes of houses, and multiple agents may own houses in the same class. Agents owning the same house are said to be *similar*, though they might have different preference reports.

For TTCA to remain SP, we need an arbitrary way to break ties among identical houses when looking at cycles. A natural way would be to break ties with the arrival order of agents, with house belonging to *later* arriving agents having higher priority (and otherwise at random). For example, A owns h_1 and arrives before B , who owns h_2 ; h_1 and h_2 are in the same class. If C most prefers that class, in TTCA he would point to h_2 first. If that's not available to him, then C would point to h_1 and so on.

Given a generalized TTCA with such a tie breaking scheme, we may simulate it with a classical TTCA, cTTCA, where there are only distinct houses. Consider agent A participating in TTCA with preference, $\succ_A = c_1 \succ_A c_2 \dots$, where c_i is a class of houses. We can construct an agent in cTTCA with preference among houses present in the system: $\succ'_A = c_{11} \succ'_A c_{12} \dots \succ'_A c_{21} \succ'_A c_{22} \dots$, where c_{i1}

is the house in class i with the highest tie-break priority, i.e. belonging to the latest arriving agent. Running classical cTTCA on $\{\succ'\}$ is identical to running TTCA on $\{\succ\}$ with the tie breaking scheme and remains SP.

We consider online mechanisms where agents trade through participation in TTCA cycles. In order to be feasible, if a TTCA cycle occurs at time t , then all participating agents must be present at t . An agent may participate in multiple TTCA cycles. However the example below shows how the ability to participate in multiple TTCA cycles can easily create incentives for agents to misreport their preferences. This is familiar from Kurino [5].

Motivating Example. Consider a scenario with three agents (A, h_A) , (B, h_B) , and (C, h_C) . A arrives and departs in time 1 and reports preference $h_C \succ_A h_A \succ_A h_B$. B arrives and departs in time 2 and reports preference $h_A \succ_B h_B \succ_B h_C$. C arrives and in 1, departs in 2 and the mechanism allows C to participate in TTCA at both times. Suppose C has true preference $h_B \succ_C h_C \succ_C h_A$. If C reports his true preference, then there would be no trade for any agent. But if C misreports $h_B \succ'_C h_A \succ'_C h_C$, then he can obtain h_A in the first round, which he can use to obtain h_B in the second round.

In the rest of this section, we restrict our consideration to mechanisms that allow each agent to participate in at most one TTCA. Given any sample path ω , we may unambiguously state the *participation time* of an agent A , denoted $t(\succ_A)$ for report $\succ_A$, and the time (if any) when it participates in TTCA. An agent present at time t and who has not participated in TTCA yet is said to be *available*. This is still a rich domain of mechanisms; which TTCA an agent participate in can depend on his arrival/departure and preference report.

A mechanism is *simple* if given a scenario $\omega(0, t)$ and agent $(A, h_A, \succ_A, a, d)$ present and available at t , there exists a set of agents $\Lambda = \{A_i, h_i, \succ_i, a_i > t, d_i\}$, called the *perfect match set* for A in scenario $\omega(0, t)$, such that (a) if a continuation $\omega(t+)$ contains Λ , then A receives his most preferred house according to $\succ_A$ under the scenario $\omega(0, t) + \omega(t+)$, and (b) if B is present in $\omega(0, t)$ and not similar to A then B does not trade with any agent in Λ .

Two observations about simple mechanisms:

- [O1] Given a scenario $\omega(0, t)$ and any agent A present and available in period t and with $d_A > t$, there exists a continuation ω_1 where the perfect match set for A arrives and A receives h_1 , his most preferred house.
- [O2] Given a scenario $\omega(0, t)$, and an agent A present and available in t with $d_A > t$, there exists a continuation where for each of the other present agents, B , not similar to A , a perfect match set for B arrives, and these are the only arriving agents. In this scenario, A would have no candidate for trade since all other originally present agents not similar to it would be matched up with perfect match sets and A can not trade with another agent's perfect match sets by definition. A can not trade with a similar agent since they have identical houses.

Later we will demonstrate that our mechanisms are *simple*. Now we state the requirements for simple mechanisms to be strategyproof.

Claim 3.2. *If an online house allocation mechanism is SP and simple and agent A participates in TTCA in period $t(\succ_A)$ for some report $\succ_A$, then fixing scenario $\omega(0, t)$ in regard to all agents except A , agent A continues to participate in period $t(\succ_A)$ for all reports $\succ'_A$.*

Proof. Consider any ω (fixing the instance for all agents except A .) Let t_1 be the earliest of $\{t(\succ'_A)\}$ for all reports $\succ'_A$ for agent A , and $\succ_A$ be the corresponding preference report. Let $\omega(0, t_1)$ be the

restriction of ω up to t_1 . We want to prove that if there exists $\succ_A$ with participation time $> t_1$, then we can construct a scenario where A has incentive to misreport. We consider various classes of $\succ_A$:

(1) If $\succ_A$ has $h_1 \succ_A h_0$, then A reporting $\succ_A$ must participate in TTCA at t_1 . Suppose $\succ_A$ does not participate at t_1 . By [O2], there exists a continuation $\omega_1(t_1+)$ from t_1 onward where A ends up keeping his original house h_0 . In this scenario, A would have benefited from misreporting $\succ_1 \neq \succ_A$ in order to obtain h_1 .

(2) Now consider if A has the true preference $\succ_A = h_2 \succ_A \dots \succ_A h_1 \succ_A \dots h_0 \succ_A \dots$, for any feasible $h_2 \notin \{h_1, h_0\}$. By above, $\succ_A$ participates at t_1 . There are two possibilities.

i. $(A, \succ_A)$ is allocated h_2 at t_1 . By (1) this implies in particular that *all* $\succ'_A$ that rank h_2 highest must trade at t_1 , and not just preferences of type $\succ_A$.

ii. $(A, \succ_A)$ is not allocated h_2 at t_1 . Suppose there exists $\succ'_A = h_2 \succ'_A \dots$ that does not trade at t_1 . Then there is a continuation $\omega_2(t_1+)$ from t_1 onward where A reporting $\succ'_A$ receives h_2 by [O1]. Then A would have benefited from reporting $\succ'_A$ instead of $\succ_A$ under this scenario. Since h_2 is arbitrary, this shows that all preference reports must participate at t_1 . $\square$

This shows that a simple SP mechanism cannot use agent's reported preference to decide at which time to let it participate in TTCA.⁴

4 Partition Mechanisms

Given the analysis in the previous section, we consider now a special class of mechanisms that ensure that each agent only participates in a single TTCA, and moreover determines the group of agents with which an agent can trade without considering the reported type profile of an agent.

Definition 4.1 (Partition mechanism). *Partition the agents into groups, $\{(P_1, t_1), (P_2, t_2), \dots, (P_k, t_k)\}$, such that all the agents in $P_j \subseteq N$ are present at $t = t_j$ and each agent is in exactly one group. The partition is constructed in a way that is independent of the agents' reported preferences. Run TTCA on trading set P_j in period t_j , for every $j \in \{1, \dots, k\}$.*

We have the following easy claim:

Claim 4.1. *A partition mechanism is strategyproof.*

This follows immediately from the strategyproofness of TTCA given that each agent is placed in a single TTCA event and this placement is independent of its report (and recall that arrival and departure times are not manipulable in our model.)

4.1 Simple Partition Mechanisms

One simple example of a partition mechanism is DO-TTCA, which runs TTCA only amongst the agents that depart in each period t .

⁴ It is possible for a simple SP mechanism to use reported preferences to decide in which partition of TTCA to let an agent participate in, so long as the partitions are concurrent and always occur in the same round. Consider a family of such mechanism as follows: at time t , take all the present agents and arbitrarily divide into two groups P_1 and P_2 (can be more than two). For a new agent A with reported pref $\succ$, simulate the TTCA of $\succ$ in P_1 and P_2 and see under which partition, A would have received better allocation under $\succ$ and then let A join that partition. Such mechanisms are strategyproof. Through simulation, we observe empirically that the performance of the mechanism improves when agents are allowed to participate in TTCA with a large number of other agents. So in practice, it's more effective to combine the partitions P_1 and P_2 into one large partition P . We therefore examine in detail mechanisms where the TTCA an agent participates in does not depend on its reported preference.

This continues until all agents have arrived, where-up TTCA is run with the remaining agents. One obvious flaw with DO-TTCA is that it forfeits the chance to execute trades amongst a large number of agents that are present but may depart at distinct times.

The *Threshold TTCA* (T-TTCA) is designed to address this problem. Let $D(t) = \{i \in N : d_i = t\}$. An active agent in period t is an agent that is present in the market and has not yet participated in a TTCA. Let $A(t)$ denote the active agents in period t .

Algorithm 1 T-TTCA

INPUT: $(N, \succ, \rho)$

OUTPUT: House allocation

```

1: while some agents still to arrive do
2:   At each time slot  $t$ 
3:   if  $D(t) \neq \emptyset$  then
4:     if  $|A(t)| > THRESHD$  then
5:       Execute TTCA with the agents  $A(t)$ 
6:       Mark all the agents in  $A(t)$  as inactive.
7:     else
8:       Execute TTCA with the agents  $D(t)$ 
9:     end if
10:  end if
11:  if all the agents have arrived then
12:    run TTCA with all the present agents
13:  end if
14: end while

```

The *THRESHD* parameter can be selected for a particular probabilistic model of agent preferences and arrival-departures to maximize system performance.

4.2 Stochastic Optimization

Mechanism SO-TTCA adopts a sample-based stochastic optimization method for partitioning the agents. Every agent in $A(t) \cap D(t)$, is included in the trading group in period t . For any other agent $A_i \in A(t) \setminus D(t)$, the decision about whether to include the agent is made based on the solution to K (sampled) offline scheduling problems. The offline problem determines a partition of agents given knowledge of the arrival-departure schedule but without considering agent preferences. A reasonable heuristic first identifies the period in which there are maximum number of agents present, say t_1 . Consider these agents as one group in the partition, say P_1 , and then recurse on the remaining agents. The intuition is that it tends to be beneficial to move an agent from a smaller trading group into a larger trading group because this leads to more trading options for each participant. An empirical analysis of the rank-efficiency for different methods to construct partitions supports this intuition.

Let N_i denote the number of times agent A_i is scheduled into period t over these K offline problems, where for each of these problems a possible arrival-departure schedule for the n_t agents still to arrive is sampled from a probabilistic model of the domain. Agent A_i is allocated to the trading group in the current period if this count N_i is greater than $K(1 - \frac{n_t}{n})$, where n_t is the number of agents that have arrived up to and including period t . The effect of comparing to $K(1 - \frac{n_t}{n})$ is that the mechanism is more likely to place an agent into the trading group as the number of agents still to arrive decreases (and thus the opportunity to trade decreases).

4.3 Partition mechanisms are simple

- *DO-TTCA*: Given agent $(A, h_0, \succ_A, a, d)$, the perfect match set is a set of n' identical agents $(A', h', \succ_{A'}, d, d)$ such that h' is A' 's most preferred house according to $\succ_A$ and $\succ_{A'}$ ranks h_0 the

Algorithm 2 SO-TTCAINPUT: $(N, \succ, \rho)$

OUTPUT: House allocation

```

1: while some agents still to arrive do
2:   At each time slot  $t$ 
3:   if  $D(t) \neq \emptyset$  then
4:      $S = D(t) \cap A(t)$ 
5:     Generate  $K$  samples of arrival-departure for the agents still
       to arrive.
6:     For each of these  $K$  scenarios, call schedule routine.
7:     For each agent  $i \in A(t) \setminus D(t)$ 
8:       if  $|\{k \in K \text{ s.t. scheduled period of } i = t\}| > (1 - \frac{n_t}{n}) \cdot K$ 
          then
9:          $S \leftarrow S \cup \{i\}$ 
10:      end if
11:      Execute TTCA with the agents from  $S$ 
12:    end if
13:    if all the agents have arrived then
14:      run TTCA with all the present agents
15:    end if
16: end while

```

highest and h' second. Here n' is the number of agents similar to A present in the system. In any continuation that contains such A 's, A 's and A will be in the same TTCA at time d and one A' will trade with A . Only agents similar to A may trade with an A' .

- **T-TTCA:** Given agent $(A, h_0, \succ_A, a, d)$ and any scenario $\omega(0, t)$ such that A is available at t , with $t < d$. Fix some threshold, $THRSHD$. A perfect match set for A is a set of m agents $\{A_i, h_i, \succ_i, t+1, t+1\}$ who arrive and depart in time $t+1$, where $m > THRSHD$. The first n' of agents A_i are identical to A' constructed for DO-TTCA. All other agents rank their own house h_i first. They are dummy agents who do not trade and only serve to trigger the threshold. In any continuation that contains $\{A_i\}$, at time $t+1$, there are more available agents than $THRSHD$, and hence all present agents participate in TTCA. Agents A and one A' would trade and A gets his top house h' .
- **SO-TTCA:** Given agent $(A, h_0, \succ_A, a, d)$ and any scenario $\omega(0, t)$ such that A is available at t , with $t < d$. Suppose there are n_t agents still yet to arrive. A perfect match for A is the same set of agents $\{A_i\}$ as constructed in the case of T-TTCA, with $m = n_t$. In any continuation containing $\{A_i\}$, there are no additional future arrivals. Hence in the offline scheduling problem, all agents present at $t+1$ would be put into the same partition. In this TTCA cycle, A would trade with A' and obtain h' .

5 Simulation Results

We perform experiments to evaluate the rank-efficiency of O-TTCA, DO-TTCA, T-TTCA, and SO-TTCA under various simulated environments. The preferences over houses are sampled uniformly at random for all the agents except in environment [E4]. For each environment and each mechanism, all results are averaged over 2000 random problem instances. By “waiting time” (or patience) we mean the number of periods an agent is present in the market.

[E1] A Poisson process with arrival rate λ is run until n agents arrive. Each agent’s waiting time is exponentially distributed with parameter μ . We adopt $\lambda = \frac{n}{8}$, and $\mu = 0.01\lambda$.

[E2] Same as [E1] except $\mu = 0.1\lambda$

[E3] Arrival time for every agent is uniformly distributed on

$\{1, \dots, T\}$. The departure time for agent A_i is uniformly distributed on $\{a_i, a_i + \frac{T}{8}\}$. We adopt $T = 30$ and if it happens that, $d_i > T$, we put $d_i = T$.

[E4] Non-uniform preferences: some houses are more demanded than the other. We first associate a *popularity index* with each house, with the popularity index for house A_j defined as the probability density $f(x)$ of a Normal distribution with mean 1, standard deviation 0.3, and evaluated at $x = \frac{j}{n}$. Given a popularity index assigned to each house, we generate a preference profile for agent A_i by sampling houses according to popularity, with houses sampled without replacement with probability proportional to the popularity of a house. The sample order defines an agent’s preferences, with the first house the most-preferred, the second house sampled the second most-preferred, and so on [13]. The arrival-departure are set as in [E1].

In T-TTCA, we varied the *THRLD* parameter around the expected number of the agents arriving in each period, and experimentally observed that the rank efficiency is optimized with a threshold that is set to the expected number of agents arriving in each period. For [E3], this is $THRLD = \frac{n}{T}$ and for [E1], [E2] and [E4] this is $THRLD = \lambda$.

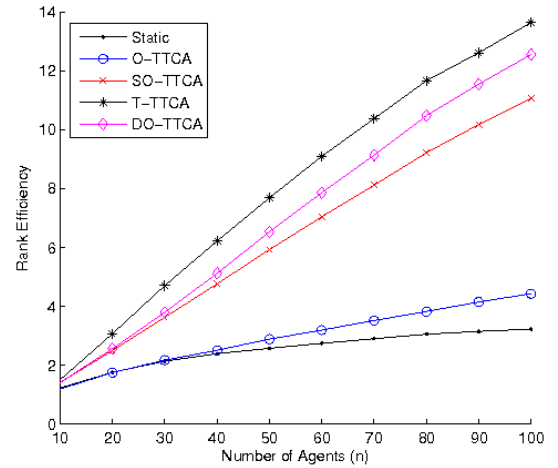


Figure 1. Rank-efficiency against the number of agents in environment [E1] for patient agents with $\lambda = n/8$ and $\mu = 0.01\lambda$.

We plot the rank-efficiency, for environments [E1] to [E4] in Figures 1–4 respectively. In each figure, we also include the rank-efficiency for the O-TTCA mechanism as a reference. Recall that the O-TTCA mechanism is not strategyproof, and significantly less constrained in that it allows for trading in every period. We observe that when agents arrive into the system by a Poisson process, SO-TTCA improves rank-efficiency over DO-TTCA by 4-10% depending upon the waiting times of the agents. Specifically, when the $\mu = 0.1\lambda$, SO-TTCA improves by 4% and for $\mu = 0.01\lambda$ it improves over DO-TTCA by 10%.

T-TTCA performs well only when the waiting times are lower (i.e., for higher μ , [E2]). When the waiting times are high (in experiment [E1]), T-TTCA fails to capture the future periods in which there may be large number of agents accumulated. Even if there are many active agents in the current period, they may be patient and willing to wait for some future period.

In [E3], the SO-TTCA and T-TTCA mechanisms have much better performance than DO-TTCA, improving rank-efficiency by 15-

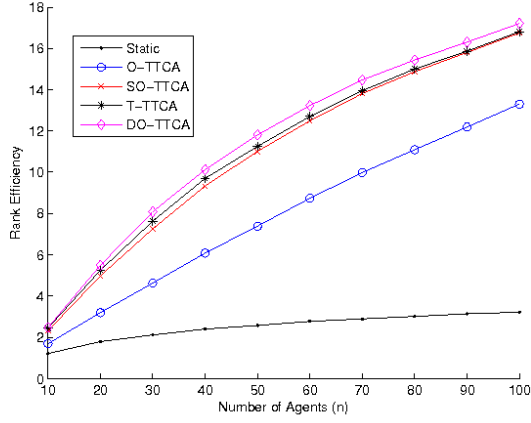


Figure 2. Rank-efficiency against the number of agents in environment [E2] for less patient agents with $\lambda = n/8$ and $\mu = 0.1\lambda$

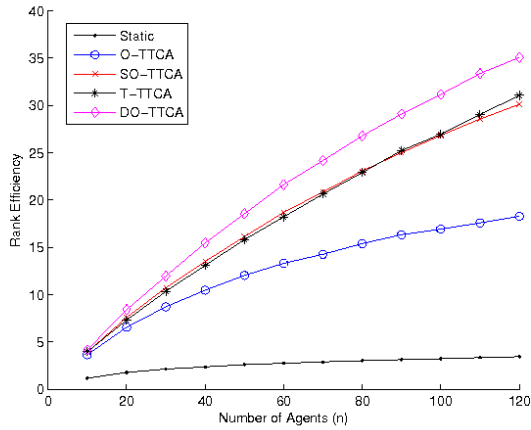


Figure 3. Rank-efficiency against the number of agents in environment [E3] with a uniform arrival-departure model.

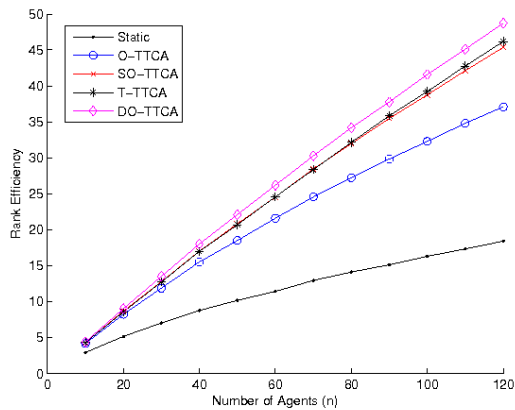


Figure 4. Rank-efficiency against the number of agents in environment [E4] with preferences that are correlated across agents and depend on the popularity index of a house.

20% for large numbers of agents. The performance of SO-TTCA and T-TTCA are almost identical under this model. With the waiting time being relatively small as compared to experiments in [E1] and [E2], T-TTCA is able to capture the periods in which more agents are present simultaneously. In general, SO-TTCA is more robust to different arrival-departure models than T-TTCA. In addition, SO-TTCA requires no tuning, whereas T-TTCA requires that the threshold is appropriately set.

6 Conclusions

In this paper we have considered a dynamic version of the house allocation problem. We identify a trade-off between efficiency and individual rationality in the online version of the house allocation problem, and identify a requirement that agents cannot be able to trade with different subsets of agents by changing their type report. We consider a family of partition mechanisms in which an agent's trading group is determined without regard to its type. A clear benefit is established for T-TTCA and SO-TTCA over a method that performs trading upon the departure of one or more agents, and the stochastic optimization approach (SO-TTCA) outperforming the threshold approach (T-TTCA) for Poisson arrival processes.

An interesting aspect of the use of stochastic optimization is that it determines *which* subset of agents should trade but not *what* trade should occur, and without consideration of specific agent types. This preserves strategyproofness. The most interesting immediate next steps are to consider a generalization in which agents can misreport arrival and departure, while continuing to explore the role for stochastic optimization in identifying useful trading groups.

Acknowledgments. The first author want to acknowledge Prof Y Narahari and Infosys Technologies Pvt Ltd for financial support.

REFERENCES

- [1] A. Abdulkadiroglu and T. Sönmez, 'House allocation with existing tenants', *J. of Economic Theory*, **88**(2), 233–260, (1999).
- [2] P. Awasthi and T. Sandholm, 'Online stochastic optimization in the large: Application to kidney exchange', in *Proc. IJCAI'09*, (2009).
- [3] Y. Chevaleyre, U. Endriss, and N. Maudet, 'Simple negotiation schemes for agents with simple preferences: Sufficiency, necessity and maximality', *J. Auton. Agents and Multiagent Systems*, **20**(2), 234–259, (2010).
- [4] S. Gujar and D. C. Parkes, 'Dynamic Matching with a Fall-back Option', in *Proc. ECAI*, (2010).
- [5] M. Kurino, 'House allocation with overlapping agents: A dynamic mechanism design approach', Technical report, Dept. of Economics, Univ. of Pittsburgh, (2009).
- [6] J. Ma, 'Strategy-proofness and the strict core in a market with indivisibilities', *Int. J. of Game Theory*, **23**(1), 75–83, (1994).
- [7] D. C. Parkes, 'Online mechanisms', in *Algorithmic Game Theory*, eds., N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, chapter 16, CUP, (2007).
- [8] A. E. Roth, 'Incentive compatibility in a market with indivisible goods', *Economics Letters*, **9**(2), 127–132, (1982).
- [9] T. Sandholm, 'An implementation of the contract net protocol based on marginal cost calculations', in *Proc. AAAI*, pp. 256–256, (1993).
- [10] L. Shapley and H. Scarf, 'On cores and indivisibility', *J. of Math. Econ.*, **1**(1), 23–37, (1974).
- [11] T. Sönmez and U. Ünver, 'Matching, allocation, and exchange of discrete resources', in *Handbook of Social Economics*, eds., J. Benhabib, A. Bisin, and M. Jackson, Elsevier, (2008).
- [12] U. Ünver, 'Dynamic kidney exchange', *Rev. of Econ. Stud.*, (2009).
- [13] J. Zou, S. Gujar, and D. C. Parkes, 'Tolerable Manipulability in Dynamic Assignment without Money', in *Proc. AAAI*, (2010).

Computation of the robust preference relation combining a Choquet integral and utility functions¹

Christophe Labreuche² and Pedro Miranda³ and Fabien Lehuédé⁴

Abstract. Preference modelling consists in constructing a preference relation from initial preferences given by a decision maker. We are interested in the preference relation obtained from the use of a Choquet integral and utility functions on each attribute. The necessary preference relation is constructed as the intersection of all preference relations corresponding to a Choquet integral which are compatible with the initial preferences of the decision maker. We are interested in determining whether an option is necessarily preferred to another one. We show that this is equivalent to solving a mixed-integer quadratic problem. Moreover, we give simple necessary and sufficient conditions for the satisfaction of the necessary preference relation.

Keywords: Choquet integral, Robust preference relation, utility functions.

1 Introduction

Multi-Criteria Decision Aid aims at representing the preferences of a Decision Maker (DM) over some options described by several points of view or attributes. We are interested in this paper in the case where the preference relation can be described by an overall utility in which the aggregation of the attributes is obtained through the Choquet integral. The preference model is then thoroughly constructed once the partial utility function over the attributes and the parameters of the Choquet integral - namely the capacity - are determined.

To this end, the facilitator asks the DM to provide some preferential information. Regarding the aggregation function, the most broadly used type of preferential information is when the DM says that an option is preferred to another one. The major difficulty the facilitator faces is that there usually does not exist one single value of the parameters of the model that fulfils the preferential information. Most of the elicitation methods based on the Choquet integral suffer from two limitations. First of all, the learning procedure usually ignores the construction of the utility functions (they are assumed to be given) and focuses only on the determination of the capacity. Secondly, the identification of the capacity consists in selecting a capacity (fulfilling the preferential information) that maximizes some functional, which may be an entropy for instance [8, 12, 13]. This

is not quite satisfactory for the DM since he does not usually understand what maximizing the functional really means. The use of a maximization problem introduces some additional information that does not come from the DM.

The facilitator shall rather stick strictly to what the DM says and add no further information. One then looks for a robust way to recommend some comparisons among the options from the preferential information [16]. The concept of *necessary preference relation* has been recently introduced for robust quantitative multi-criteria decision models [10, 11]. It has been applied to the Choquet integral in [2]. An option is *necessarily* preferred to another option according to the necessary preference relation if the first option is preferred to the second one according to all models that fulfil the preferential information provided by the DM. This necessary preference relation is usually incomplete, unless the model is completely specified from the preferential information of the DM. The necessary preference relation is cautious in the sense that it does not always provides a comparison between the options (it may be incomplete), but when it says that an option is necessarily preferred to another one, the preference is sure.

Let us mention that these concepts of necessary and possible preference representation are originated from earlier works on preference modelling. In the context of decision rules, the necessary and possible preference relations are the lower and upper approximations of a rough set [9]. Moreover, in qualitative decision models, these concepts correspond to the dominance and ordering queries for the Ceterus Paribus nets [5].

The approach from [2] using the Choquet integral does not include the utility functions in the construction of the robust preference relations. The idea of determining not only the admissible capacities but also the utility functions has been already considered in [1]. We first generalize this framework to incorporate the imprecision related to both the aggregation function and the utility functions. There are basically two ways to elicit the preference model: either by constructing all elements of the model at the same time from global learning examples, or by constructing each element of the model separately. We adopt the second way in this paper, which is the standard approach with the Choquet integral. In other words, the DM provides preferential information regarding only the utility functions, and preferential information about only the aggregation function. From this elicitation process, we show that determining whether an option is necessarily preferred to another one is equivalent to solving a mixed-integer quadratic program.

The paper is organized as follows. Section 2 recalls the background on the standard representation of the preference relation and the Choquet integral. The preferential information that is provided

¹ This research has been supported in part by the MOVIDA project funded by the ANR (French National Research Agency), and by the grant numbers MTM2007-61193, MTM2009-10072 and BSCH- UCM910707.

² Thales Research & Technology, 1 avenue Augustin Fresnel, F91767 Palaiseau Cedex, France, email: christophe.labreuche@thalesgroup.com

³ Facultad de Ciencias Matemáticas, Plaza de Ciencias, 3, 28040, Madrid, Spain, email: pmiranda@mat.ucm.es

⁴ Ecole des Mines de Nantes, La Chantrerie - BP20722, F44307 Nantes, France, email: fabien.lehuede@mines-nantes.fr

by the DM is described in Section 3. It separates the information regarding the utility functions and the aggregation function. The general necessary preference relation is also given. The computation of this relation is proposed in Section 4. We first show that this problem can equivalently be written as a mixed-integer quadratic problem. Finally, we give a necessary and a sufficient condition for the satisfaction of the necessary preference relation that can be easily solved by two linear programs.

2 Background

2.1 Organization of the preferences of the DM

The criteria of the DM agent are denoted by $N = \{1, \dots, n\}$. In order to quantify the satisfaction of the options on a criterion $i \in N$, the DM needs only to know the value of the options on an attribute characterized by a set X_i . The comparison of the options can be made only from a knowledge of the value of these options on the attributes. Hence the options can be considered as elements of $X = X_1 \times \dots \times X_n$.

We wish to construct the preferences of the DM regarding the elements of X through a binary relation $\succsim_X$ over X . One of the most classical representations of $\succsim_X$ is based on an overall utility $U : X \rightarrow [0, 1]$. Hence for all $x, y \in X$, we have

$$x \succsim_X y \iff U(x) \geq U(y). \quad (1)$$

A standard representation of U is the so-called transitive decomposable model $U(x) = F(u(x))$, where $u_i : X_i \rightarrow [0, 1]$ is the utility function on attribute $i \in N$,

$$u(x) = (u_1(x_1), \dots, u_n(x_n)),$$

and $F : [0, 1]^N \rightarrow [0, 1]$ is the aggregation function. The functions u and F need to be elicited.

2.2 Choquet integral

The aggregation function F is most of the time a simple weighted sum. This model fails to represent real life decision strategies such as veto, conditional relative importance between criteria, and complementarity and substitutability among the criteria. The Choquet integral has been proposed as a generalization of the weighted sum in MCDA [6]. It is based on the concept of a capacity (also called fuzzy measure) [18].

Definition 1 A capacity μ on N is a function $\mu : 2^N \rightarrow [0, 1]$, satisfying the following axioms.

- (i) $\mu(\emptyset) = 0, \mu(N) = 1$.
- (ii) $A \subseteq B \subseteq N$ implies $\mu(A) \leq \mu(B)$.

We introduce a useful linear transformation of capacities. The *Möbius transform* m of a capacity μ is the unique solution of the equation

$$\forall A \subseteq N \quad \mu(A) = \sum_{B \subseteq A} m(B),$$

and is given by:

$$m(A) := \sum_{B \subseteq A} (-1)^{|A|-|B|} \mu(B).$$

A capacity is defined by $2^n - 2$ coefficients, which is much more than a weighted sum. The concept of k -additive capacity has a complexity in-between a capacity and a weighted sum. More precisely,

a capacity μ is said to be k -additive [7] if $m(A) = 0$ whenever $|A| > k$ and there exists A with $|A| = k$ such that $m(A) \neq 0$. An interesting particular case is the 2-additive capacities [15].

A set function $m : \{A \subseteq N : |A| \leq k\} \rightarrow \mathbb{R}$ corresponds to k -additive Möbius coefficients iff the following monotonicity and boundary conditions are satisfied

$$\sum_{B \subseteq A : |B| < k} m(B \cup \{i\}) \geq 0, \quad \forall i \in N \quad \forall A \subseteq N \setminus \{i\} \quad (2)$$

$$\sum_{A \subseteq N : |A| \leq k} m(A) = 1 \quad (3)$$

We denote by $\mathcal{M}^k$ the set of functions $m : \{A \subseteq N : |A| \leq k\} \rightarrow \mathbb{R}$ satisfying (2) and (3).

In the remaining of the paper, we will restrict to k -additive capacities. Such model is a good compromise between the complexity of the model and its ability to represent subtle decision strategies. The value of $k \in \{2, \dots, n\}$ is fixed.

The expression of the Choquet integral with respect to $m \in \mathcal{M}^k$ has the following form [15] :

$$C_m(a) = \sum_{A \subseteq N : |A| \leq k} m(A) \min_{i \in A} a_i. \quad (4)$$

3 Description of the incomplete information and the overall preference relation

As we have already said, the overall problem of comparing options in X is complex due to the number of criteria and their interactions. The preferential information asked from the DM is not composed of comparison of elements of X . This problem is decomposed into several sub-problems. The DM is asked to give separately preferential information regarding the aggregation function, and regarding each utility function. This approach is adopted for the case of a weighted sum in [3, 17], and the Choquet integral in [14].

3.1 Incompleteness on the utility functions

Let $i \in N$. We assume that the attribute X_i is an interval of $\mathbb{R}$. Classically, an utility function u_i is represented as a piecewise affine function. It is thus characterized by two vectors $(t_i^1, t_i^2, \dots, t_i^{p_i})$ of elements of X_i and $(u_i^1, u_i^2, \dots, u_i^{p_i})$ of real numbers where $t_i^1 < t_i^2 < \dots < t_i^{p_i}$. The utility function u_i for any $x_i \in X_i$ is obtained by $u_i(x_i) = I_{(t_i^1, t_i^2, \dots, t_i^{p_i}), (u_i^1, u_i^2, \dots, u_i^{p_i})}(x_i)$, where

$$I_{(t_i^1, t_i^2, \dots, t_i^{p_i}), (u_i^1, u_i^2, \dots, u_i^{p_i})}(x_i) = \begin{cases} u_i^1 & \text{if } x_i \leq t_i^1 \\ u_i^k + \frac{x_i - t_i^k}{t_i^{k+1} - t_i^k} (u_i^{k+1} - u_i^k) & \text{if } x_i \in [t_i^k, t_i^{k+1}] \\ u_i^{p_i} & \text{if } x_i \geq t_i^{p_i} \end{cases}$$

The MACBETH method [3] enables to construct the values of $(u_i^1, u_i^2, \dots, u_i^{p_i})$ once p_i and the vector $(t_i^1, t_i^2, \dots, t_i^{p_i})$ are given. The idea of the MACBETH approach is to ask the DM to assess the difference of attractiveness between any pair (u_i^k, u_i^l) in an ordinal scale such as *very weak*, *weak*, *moderate*, *strong*, *very strong* and *extreme*. The values of $u_i^1, \dots, u_i^{p_i}$ are then obtained by linear programming.

In the elicitation process, the information provided by the DM is not sufficient to specify a unique vector $(u_i^1, \dots, u_i^{p_i})$. Rather, it specifies upper bounds $(\bar{u}_i^1, \dots, \bar{u}_i^{p_i})$ and lower bounds

$(\underline{u}_i^1, \dots, \underline{u}_i^{p_i})$, with $\bar{u}_i^k \geq \underline{u}_i^k$ for all k . Hence, instead of getting a unique utility function, one obtains a family of utility functions in-between two envelop utility functions $\underline{u}_i = I_{(t_i^1, t_i^2, \dots, t_i^{p_i}), (\underline{u}_i^1, \underline{u}_i^2, \dots, \underline{u}_i^{p_i})}$ and $\bar{u}_i = I_{(t_i^1, t_i^2, \dots, t_i^{p_i}), (\bar{u}_i^1, \bar{u}_i^2, \dots, \bar{u}_i^{p_i})}$. For consistency reasons, all admissible utility functions are supposed to be comonotone, and so are $\underline{u}_i$ and $\bar{u}_i$. Let us recall that two functions f and g on X_i are said to be comonotone iff for all $x_i, y_i \in X_i$

$$(f(x_i) - f(y_i))(g(x_i) - g(y_i)) \geq 0.$$

The set of admissible utility functions is (see Figure 1)

$$\mathcal{U}_i = \{u_i : X_i \rightarrow [0, 1] : \forall x_i \in X_i \underline{u}_i(x_i) \leq u_i(x_i) \leq \bar{u}_i(x_i) \text{ and } \underline{u}_i, u_i, \bar{u}_i \text{ comonotone}\}.$$

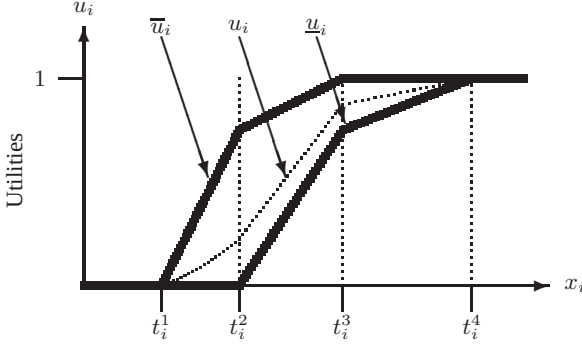


Figure 1. Functions $\underline{u}_i$, u_i and $\bar{u}_i$.

3.2 Incompleteness on the aggregation functions

Classically, the identification of the capacity is performed from learning data. This preferential information can classically be of three types [8]: a comparison of two profiles in $[0, 1]^N$, an overall score associated to a profile in $[0, 1]^N$, or less intuitive preferential information related to the importance or interaction of the attributes. The information of the second and the third types are not always easy to obtain from a DM, especially in multi-criteria decision aid. Moreover, as argued in [4], the main information obtained from empirical data of the second type is a requirement stating that the data set shall preserve the order deduced from the overall scores, which corresponds to the first type of information. We focus thus on the first type of information in this paper.

The preferential information regarding the aggregation function is thus composed on a partial order relation $\succeq$ over $[0, 1]^N$. For $a, b \in [0, 1]^N$, $a \succeq b$ means that a is judged at least as good as b . The set of Möbius coefficients that fulfil the preferential information $\succeq$ of the DM is:

$$\mathcal{M}^k(\succeq) = \{m \in \mathcal{M}^k : [a \succeq b \implies C_m(a) \geq C_m(b)]\}.$$

3.3 Overall necessary preference relation

We denote by Π the set of all preferential information of the DM regarding the aggregation function and the utility functions:

$$\Pi = \{\succeq, \mathcal{U}_1, \dots, \mathcal{U}_n\}.$$

The necessary preference relation restricted to the aggregation level is a partial preorder $\succeq_{[0,1]^N}^\Pi$ over $[0, 1]^N$ defined for all $a, b \in [0, 1]^N$ by [2]

$$a \succeq_{[0,1]^N}^\Pi b \iff \forall m \in \mathcal{M}^k(\succeq) \quad C_m(a) \geq C_m(b).$$

The preference relation can be easily extended to X taking into account the imprecision resulting from the preferential information Π .

Definition 2 The necessary preference relation over X resulting from Π is the partial preorder $\succeq_X^\Pi$ over X defined for all $x, y \in X$ by

$$x \succeq_X^\Pi y \iff \forall m \in \mathcal{M}^k(\succeq) \forall (u_1, \dots, u_n) \in \mathcal{U}_1 \times \dots \times \mathcal{U}_n \quad C_m(u_1(x_1), \dots, u_n(x_n)) \geq C_m(u_1(y_1), \dots, u_n(y_n)).$$

The partial preference relation $\succeq_X^\Pi$ represents better what the DM has provided as input data Π than the complete preference relation $\succeq_X$ (see (1)) in which the utility functions and the Möbius coefficients are uniquely fixed.

Once $\succeq_X^\Pi$ is determined, the stable sets are computed and displayed to the DM as recommended options [19]. A set $Y \subseteq X$ is a *stable set* w.r.t. the asymmetric part $\succ_X^\Pi$ of $\succeq_X^\Pi$ if $\nexists x, y \in Y$ such that $x \succ_X^\Pi y$ (internal stability), and $\forall x \in X \setminus Y, \exists y \in Y$ such that $y \succ_X^\Pi x$ (external stability).

4 Computation of the necessary preference relation

We fix $x, y \in X$ and we wish to determine whether $x \succeq_X^\Pi y$. One has $x \succeq_X^\Pi y$ iff

$$\min_{m \in \mathcal{M}^k(\succeq), (u_1, \dots, u_n) \in \mathcal{U}_1 \times \dots \times \mathcal{U}_n} (C_m(u_1(x_1), \dots, u_n(x_n)) - C_m(u_1(y_1), \dots, u_n(y_n))) \quad (5)$$

is non-negative

4.1 Uncertainties on the utility functions

For $x_i, y_i \in X_i$, let

$$\mathcal{U}_i(x_i, y_i) = \{(u_i(x_i), u_i(y_i)) : u_i \in \mathcal{U}_i\}.$$

We have seen that the functions $\underline{u}_i$ and $\bar{u}_i$ are affine in each interval $[t_i^j, t_i^{j+1}]$ for all $j \in \{1, \dots, p_i - 1\}$. For such index j , we have three cases:

- (i) $\underline{u}_i$ and $\bar{u}_i$ are non-decreasing in $[t_i^j, t_i^{j+1}]$ (i.e. $\underline{u}_i^j \leq \underline{u}_i^{j+1}$ and $\bar{u}_i^j \leq \bar{u}_i^{j+1}$), and $\underline{u}_i$ or $\bar{u}_i$ is strictly increasing in $[t_i^j, t_i^{j+1}]$ (i.e. $\underline{u}_i^j < \underline{u}_i^{j+1}$ or $\bar{u}_i^j < \bar{u}_i^{j+1}$). Then all $u_i \in \mathcal{U}_i$ are non-decreasing in $[t_i^j, t_i^{j+1}]$;
- (ii) $\underline{u}_i$ and $\bar{u}_i$ are non-increasing in $[t_i^j, t_i^{j+1}]$ (i.e. $\underline{u}_i^j \geq \underline{u}_i^{j+1}$ and $\bar{u}_i^j \geq \bar{u}_i^{j+1}$), and $\underline{u}_i$ or $\bar{u}_i$ is strictly decreasing in $[t_i^j, t_i^{j+1}]$ (i.e. $\underline{u}_i^j > \underline{u}_i^{j+1}$ or $\bar{u}_i^j > \bar{u}_i^{j+1}$). Then all $u_i \in \mathcal{U}_i$ are non-increasing in $[t_i^j, t_i^{j+1}]$;
- (iii) $\underline{u}_i$ and $\bar{u}_i$ are constant in $[t_i^j, t_i^{j+1}]$ (i.e. $\underline{u}_i^j = \underline{u}_i^{j+1}$ and $\bar{u}_i^j = \bar{u}_i^{j+1}$). Then there is no condition on the monotonicity of $u_i \in \mathcal{U}_i$ in $[t_i^j, t_i^{j+1}]$.

Let $j, l \in \{1, \dots, p_i - 1\}$ such that $x_i \in [t_i^j, t_i^{j+1}]$ and $y_i \in [t_i^l, t_i^{l+1}]$. Then there are the following cases to describe the set $\mathcal{U}_i(x_i, y_i)$:

- **Situation S1** (see Figure 2): x_i and y_i do not belong to the same segment (i.e. $j \neq l$), or x_i and y_i belong to the same segment (i.e. $j = l$) and it is of type (iii). There is no relation between the utilities of x_i and y_i . We have

$$\mathcal{U}_i(x_i, y_i) = [\underline{u}_i(x_i), \overline{u}_i(x_i)] \times [\underline{u}_i(y_i), \overline{u}_i(y_i)]. \quad (6)$$

- In the other situations, x_i and y_i belong to the same segment (i.e. $j = l$) and it is of type (i) or (ii). This means that u_i is either non-decreasing or non-increasing in the interval $[t_i^j, t_i^{j+1}]$. We have two cases:

- **Situation S2** (see Figure 3): either $[x_i \geq y_i$ and $[t_i^j, t_i^{j+1}]$ is of type (i)] or $[y_i \geq x_i$ and $[t_i^j, t_i^{j+1}]$ is of type (ii)]. Here we necessarily have

$$\forall u_i \in \mathcal{U}_i \quad u_i(x_i) \geq u_i(y_i).$$

Hence

$$\mathcal{U}_i(x_i, y_i) = \{(u_i^x, u_i^y) \in [\underline{u}_i(x_i), \overline{u}_i(x_i)] \times [\underline{u}_i(y_i), \overline{u}_i(y_i)] : u_i^x \geq u_i^y\}. \quad (7)$$

- **Situation S3** (see Figure 4): either $[y_i \geq x_i$ and $[t_i^j, t_i^{j+1}]$ is of type (i)] or $[x_i \geq y_i$ and $[t_i^j, t_i^{j+1}]$ is of type (ii)]. Here we necessarily have

$$\forall u_i \in \mathcal{U}_i \quad u_i(y_i) \geq u_i(x_i).$$

Hence

$$\mathcal{U}_i(x_i, y_i) = \{(u_i^x, u_i^y) \in [\underline{u}_i(x_i), \overline{u}_i(x_i)] \times [\underline{u}_i(y_i), \overline{u}_i(y_i)] : u_i^x \leq u_i^y\}. \quad (8)$$

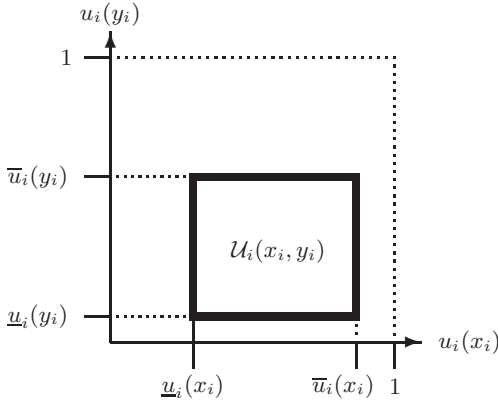


Figure 2. Set $\mathcal{U}_i(x_i, y_i)$ - situation S1.

Let us define $\hat{\mathcal{U}}_i(x_i, y_i)$ by

- **Situation S2** (see Figure 5):

$$\hat{\mathcal{U}}_i(x_i, y_i) = \{(v, v) : v \in [\underline{u}_i(x_i), \overline{u}_i(y_i)]\}. \quad (9)$$

We denote by M the set of criteria i in N that are in the situation S2.

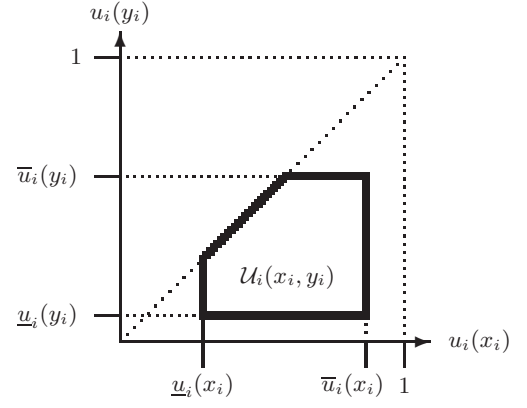


Figure 3. Set $\mathcal{U}_i(x_i, y_i)$ - situation S2.

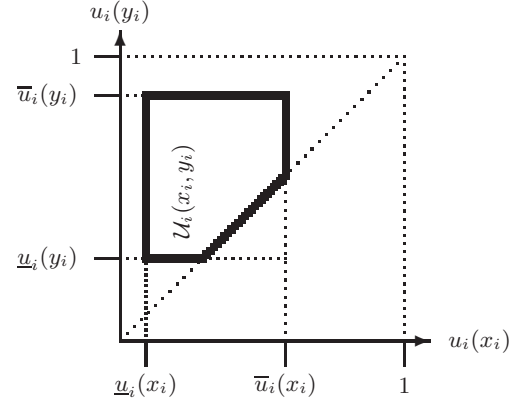


Figure 4. Set $\mathcal{U}_i(x_i, y_i)$ - situation S3.

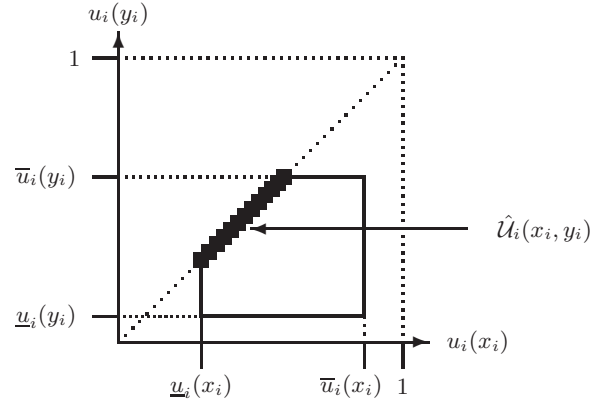


Figure 5. Set $\hat{\mathcal{U}}_i(x_i, y_i)$ - situation S2.

Due to the constraint $\sum_{i \in A} \epsilon_{A,i}^x = |A| - 1$ in (13), it is indeed sufficient when A is of cardinality 2 to use only one integer variable for each set A . The same holds for option y .

4.3 A necessary condition and a sufficient condition

We give a necessary and a sufficient conditions for the satisfaction of $x \succsim_X^\Pi y$ that can be easily solved by two linear programs.

We consider two options x and y . We set

$$\mathcal{M}^k(\Pi)[x, y] := \{m \in \mathcal{M}^k(\geq) : C_m(u(x)) \geq C_m(u(y))\} \\ \forall (u_1, \dots, u_n) \in \mathcal{U}_1 \times \dots \times \mathcal{U}_n.$$

The set $\mathcal{M}^k(\Pi)[x, y]$ is not a polytope. We have $x \succsim_X^\Pi y$ if and only if $\mathcal{M}^k(\Pi)[x, y] = \mathcal{M}^k(\geq)$.

Let

$$\overline{\mathcal{M}^k(\Pi)}[x, y] := \{m \in \mathcal{M}^k(\geq) : C_m(\overline{u}(x)) \geq C_m(\underline{u}(y))\} \\ \underline{\mathcal{M}^k(\Pi)}[x, y] := \{m \in \mathcal{M}^k(\geq) : C_m(\underline{u}(x)) \geq C_m(\overline{u}(y))\}$$

The sets $\overline{\mathcal{M}^k(\Pi)}[x, y]$ and $\underline{\mathcal{M}^k(\Pi)}[x, y]$ are polytopes.

Lemma 2 We have

$$\underline{\mathcal{M}^k(\Pi)}[x, y] \subseteq \mathcal{M}^k(\Pi)[x, y] \subseteq \overline{\mathcal{M}^k(\Pi)}[x, y].$$

Proof : Let $m \in \mathcal{M}^k(\Pi)[x, y]$ and $(u_1, \dots, u_n) \in \mathcal{U}_1 \times \dots \times \mathcal{U}_n$. Then by monotonicity of the Choquet integral,

$$C_m(u(x)) \geq C_m(\underline{u}(x)) \geq C_m(\overline{u}(y)) \geq C_m(u(y))$$

Hence $m \in \mathcal{M}^k(\Pi)[x, y]$.

Now, let $m \in \mathcal{M}^k(\Pi)[x, y]$. Then for all $(u_1, \dots, u_n) \in \mathcal{U}_1 \times \dots \times \mathcal{U}_n$

$$C_m(\overline{u}(x)) \geq C_m(u(x)) \geq C_m(u(y)) \geq C_m(\underline{u}(y)).$$

Hence $m \in \overline{\mathcal{M}^k(\Pi)}[x, y]$. ■

Corollary 1 We have

$$\min_{m \in \mathcal{M}^k(\geq)} C_m(\overline{u}(x)) - C_m(\underline{u}(y)) < 0 \implies x \not\succsim_X^\Pi y \\ \min_{m \in \mathcal{M}^k(\geq)} C_m(\underline{u}(x)) - C_m(\overline{u}(y)) \geq 0 \implies x \succsim_X^\Pi y$$

Proof : If $\min_{m \in \mathcal{M}^k(\geq)} C_m(\overline{u}(x)) - C_m(\underline{u}(y)) < 0$, it follows that $\overline{\mathcal{M}^k(\Pi)}[x, y] \neq \mathcal{M}^k(\geq)$. Applying Lemma 1, $\mathcal{M}^k(\Pi)[x, y] \neq \mathcal{M}^k(\geq)$, whence $x \not\succsim_X^\Pi y$.

Similarly, if $\min_{m \in \mathcal{M}^k(\geq)} C_m(\underline{u}(x)) - C_m(\overline{u}(y)) \geq 0$, it follows that $\mathcal{M}^k(\Pi)[x, y] = \mathcal{M}^k(\geq)$. Applying Lemma 1, $\mathcal{M}^k(\Pi)[x, y] = \mathcal{M}^k(\geq)$, whence $x \succsim_X^\Pi y$. ■

The two minimization problems

$$\min_{m \in \mathcal{M}^k(\geq)} C_m(\overline{u}(x)) - C_m(\underline{u}(y)) \\ \min_{m \in \mathcal{M}^k(\geq)} C_m(\underline{u}(x)) - C_m(\overline{u}(y))$$

are linear, and can thus be solved easily.

The usefulness of the two conditions in Corollary 1 depends on whether the envelopes $\underline{u}$ and $\overline{u}$ are close to one another.

5 Conclusion

We have described a problem where the preferential information provided by the DM is decomposed as information regarding each utility function and the aggregation function. We have described the necessary preference relation based on this preferential information. We have shown that the problem of determining whether an option is necessarily preferred to another one is equivalent to solving a mixed-integer quadratic problem. Finally, we gave necessary and sufficient conditions for the satisfaction of the necessary preference relation that can be easily solved by two linear programs.

REFERENCES

- [1] S. Angilella, S. Greco, F. Lamantia, and B. Matarazzo, 'Assessing non-additive utility for multicriteria decision aid', *European Journal of Operational Research*, **158**, 734–744, (2004).
- [2] S. Angilella, S. Greco, and B. Matarazzo, 'Non-additive robust ordinal regression: a multiple criteria decision model based on the Choquet integral', *European Journal of Operational Research*, **201**, 277–288, (2010).
- [3] C.A. Bana e Costa and J.C. Vansnick, 'A theoretical framework for Measuring Attractiveness by a Categorical Based Evaluation Technique (MACBETH)', in *Proc. XIth Int. Conf. on MultiCriteria Decision Making*, pp. 15–24, Coimbra, Portugal, (August 1994).
- [4] G. Beliakov and T. Calvo, 'Constructions of aggregation operators that preserve ordering of the data', in *EUSFLAT conference*, pp. 61–66, Ostrava, Tech Republik, (Sept 2007).
- [5] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole, 'CP-nets: a tool for representing and reasoning with conditional Ceteris Paribus preference statements.', *J. of Artificial Intelligence Research*, **21**, 135–191, (2004).
- [6] M. Grabisch, 'The application of fuzzy integrals in multicriteria decision making', *European J. of Operational Research*, **89**, 445–456, (1996).
- [7] M. Grabisch, 'Alternative representations of discrete fuzzy measures for decision making', *Int. J. of Uncertainty, Fuzziness, and Knowledge Based Systems*, **5**, 587–607, (1997).
- [8] M. Grabisch, I. Kojadinovic, and P. Meyer, 'A review of capacity identification methods for Choquet integral based multi-attribute utility theory — applications of the Kappalab R package', *Eur. J. of Operational Research*, **186**, 766–785, (2008).
- [9] S. Greco, B. Matarazzo, and R. Slowinski, 'Decision rule approach', in *Multiple Criteria Decision Analysis - State of the Art Surveys*, ed., M. Ehrgott J. Figueira, S. Greco, 507–561, Springer International Series, (2005).
- [10] S. Greco, B. Matarazzo, and R. Slowinski, 'Ordinal regression revisited: Multiple criteria ranking with a set of additive value functions', *European Journal of Operational Research*, **191**, 416–436, (2008).
- [11] S. Greco, R. Slowinski, J. Figueira, and V. Mousseau, 'Robust ordinal regression', in *New Trends in Multiple Criteria Decision Analysis*, ed., M. Ehrgott J. Figueira, S. Greco, Springer Verlag, (to appear).
- [12] I. Kojadinovic, 'Minimum variance capacity identification', *Quarterly J. of Operations Research (4OR)*, **12**, 23–36, (2006).
- [13] Ch. Labreuche, 'Identification of a fuzzy measure with an l^1 entropy', in *Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, Malaga, Spain, (June 22–27 2008).
- [14] Ch. Labreuche and M. Grabisch, 'The Choquet integral for the aggregation of interval scales in multicriteria decision making', *Fuzzy Sets & Systems*, **137**, 11–26, (2003).
- [15] T. Murofushi and S. Soneda, 'Techniques for reading fuzzy measures (III): interaction index', in *9th Fuzzy System Symposium*, pp. 693–696, Sapporo, Japan, (May 1993). In Japanese.
- [16] B. Roy, *Multicriteria methodology for decision aiding*, Kluwer Academic Publishers, Dordrecht, 1996.
- [17] T. L. Saaty, *The Analytic Hierarchy Process*, McGraw Hill, New York, 1980.
- [18] M. Sugeno, *Theory of fuzzy integrals and its applications*, Ph.D. dissertation, Tokyo Institute of Technology, 1974.
- [19] J. von Neumann and O. Morgenstern, *Game Theory and Economic Behavior*, Princeton University Press, Princeton, NJ, 1953.

On Finding Compromise Solutions in Multiobjective Markov Decision Processes

Patrice Perny and Paul Weng¹

Abstract. A Markov Decision Process (MDP) is a general model for solving planning problems under uncertainty. It has been extended to multiobjective MDP to address multicriteria or multiagent problems in which the value of a decision must be evaluated according to several viewpoints, sometimes conflicting. Although most of the studies concentrate on the determination of the set of Pareto-optimal policies, we focus here on a more specialized problem that concerns the direct determination of policies achieving well-balanced tradeoffs. We first explain why this problem cannot simply be solved by optimizing a linear combination of criteria. This leads us to use an alternative optimality concept which formalizes the notion of best compromise solution, i.e. a policy yielding an expected-utility vector as close as possible (w.r.t. Tchebycheff norm) to a reference point. We show that this notion of optimality depends on the initial state. Moreover, it appears that the best compromise policy cannot be found by a direct adaptation of value iteration. In addition, we observe that in some (if not most) situations, the optimal solution can only be obtained with a randomized policy. To overcome all these problems, we propose a solution method based on linear programming and give some experimental results.

1 INTRODUCTION

In many practical applications of Artificial Intelligence such as path planning, game search, preference-based configuration, the comparison of solutions involves several aspects or points of view that must be analyzed to make a decision. This is typically the case of some state-space search problems where any action that transitions from a state to another is evaluated with respect to different criteria, agents or scenarios, thus leading to a multi-dimensional evaluation system, non-necessarily reducible to a scalar reward or cost. For this reason, several vector-valued generalizations of search algorithms have been investigated such as MOA* [16, 12], a multiobjective extension of A* [10] finding all Pareto-optimal solution paths to reach a goal node from an initial node in an implicit state space graph. In the field of planning under uncertainty, the same type of concern is present. This explains the current interest for multicriteria or multiagent extensions of Markov Decision Processes [13, 2, 1, 8].

When several objectives must be optimized simultaneously, most of the studies on Markov Decision Processes concentrate on the determination of the entire set of Pareto-optimal solutions, i.e. policies having a cost vector that cannot be improved on a criterion without being downgraded on another criterion. However, the size of the Pareto set is often very large due to the combinatorial nature of the set of deterministic policies, its determination induces prohibitive response times and requires very important memory space as the num-

ber of states and/or criteria increases. Fortunately, there is generally no need to determine the entire set of Pareto-optimal policies, but only specific compromise policies achieving a good tradeoff between the possibly conflicting objectives (minimize time, energy consumption, risk...).

This notion of compromise is natural in multiobjective decision support and has counterparts in other optimisation contexts involving several dimensions. For example, in the context of multi-agent optimisation problems, the notion of compromise solution refers to equity or fairness [7]. In the context of optimisation under multiple scenarios, compromise solutions refer to the idea of robustness [11]. The quality of the compromise achieved can be measured using a scalarizing function discriminating between Pareto-optimal solutions [15, 4].

Motivated by such examples, we study in this paper the determination of well-balanced policies in multiobjective Markov Decision Processes. The paper is organized as follows: In Section 2, we recall the basic notions related to Markov decision processes and their multiobjective extension. In Section 3, we discuss the choice of scalarizing functions to generate compromise solutions. This leads us to adopt the augmented Tchebycheff norm as a proper scalarizing function. Section 4 is devoted to the search of Tchebycheff-optimal policies. Finally, Section 5 presents some experimental results showing the effectiveness of our approach in reaching well-balanced policies.

2 BACKGROUND

2.1 Markov Decision Processes

A Markov Decision Process (MDP) [14] is described as a tuple (S, A, T, R) where:

- S is a finite set of states,
- A is a finite set of actions,
- $T: S \times A \rightarrow \mathbf{Pr}(S)$ is a transition function, giving for each state and each action, a probability distribution over states (in the sequel, we write $T(s, a, s')$ for $T(s, a)(s')$),
- $R: S \times A \rightarrow \mathbb{R}$ is a reward function giving the immediate reward for executing a given action in a given state.

In this context, a *decision rule* δ is a procedure that determines which action to choose in each state. A decision rule can be *deterministic*, i.e. defined as a function: $\delta: S \rightarrow A$, or more generally, *randomized*, i.e. defined as a function $\delta: S \rightarrow \mathbf{Pr}(A)$ where $\mathbf{Pr}(A)$ is the set of probability distributions over A .

A *policy* π is a sequence of decision rules $(\delta_0, \delta_1, \dots, \delta_t, \dots)$ that indicates which decision rule to apply at each step. It is said to be

¹ LIP6, UPMC, France, email: firstname.surname@lip6.fr

deterministic if all the decision rules are deterministic and *randomized* otherwise. If the same decision rule δ is applied at each step, the policy is said to be *stationary* and is denoted δ^∞ .

In standard MDPs, the value of a policy π is defined by a function $v^\pi : S \rightarrow \mathbb{R}$, called *value function*, which gives the expected discounted total reward yielded by applying π from each initial state. For $\pi = (\delta_0, \delta_1, \dots, \delta_t, \dots)$, they are given by: $\forall s \in S, \forall h > 0, \forall t = 1, \dots, h$,

$$\begin{aligned} v_0^\pi(s) &= 0 \\ v_t^\pi(s) &= R(s, \delta_{h-t}(s)) + \gamma \sum_{s' \in S} T(s, \delta_{h-t}(s), s') v_{t-1}^\pi(s') \end{aligned}$$

where $\gamma \in [0, 1]$ is the discount factor. This sequence converges to the value function of π .

In this standard framework, there exists an optimal stationary policy that yields the best expected discounted total reward in each state. Solving an MDP amounts to finding one of those policies and its associated value function. The optimal value function $v^* : S \rightarrow \mathbb{R}$ can be determined by solving the *Bellman equations*:

$$\forall s \in S, v^*(s) = \max_{a \in A} R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') v^*(s') \quad (1)$$

There are three main approaches for solving MDPs. Two are based on dynamic programming: value iteration and policy iteration. The last approach is based on linear programming. We now recall value iteration and the linear programming approach as they are needed for the exposition of our results.

Value iteration consists in computing the solution of the Bellman equations using the following recursive sequence: $\forall s \in S, \forall t > 0$,

$$v_0^*(s) = 0 \quad (2)$$

$$v_t^*(s) = \max_{a \in A} R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') v_{t-1}^*(s') \quad (3)$$

This sequence converges to the optimal value function and the optimal stationary policy can be recovered by a greedy optimization.

VALUE ITERATION

- 1: $\forall s \in S, v(s) \leftarrow 0$
- 2: **repeat**
- 3: **for all** $s \in S$ **do**
- 4: **for all** $a \in A$ **do**
- 5: $q(s, a) \leftarrow R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') v(s')$
- 6: **end for**
- 7: $v(s) \leftarrow \max_{a \in A} q(s, a)$
- 8: **end for**
- 9: **until** convergence of v

An MDP can also be solved by linear programming. The Bellman equations state that value functions satisfying the following inequalities are upper bounds of the optimal value function:

$$\forall s \in S, \forall a \in A, v(s) \geq R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') v(s')$$

The linear program ($\mathcal{P}$) can then be written as follows:

$$\begin{aligned} \min \quad & \sum_{s \in S} \mu(s) v(s) \\ \text{s.t.} \quad & v(s) - \gamma \sum_{s' \in S} T(s, a, s') v(s') \geq R(s, a) \\ & \forall s \in S, \forall a \in A \end{aligned}$$

where weights μ could be interpreted as the probability of starting in a given state. Any positive μ can in fact be chosen to determine the optimal value function.

Writing the dual of this program ($\mathcal{D}$) is interesting as it uncovers the dynamic of the system:

$$\begin{aligned} \max \quad & \sum_{s \in S} \sum_{a \in A} R(s, a) x(s, a) \\ \text{s.t.} \quad & \sum_{a \in A} x(s, a) - \gamma \sum_{s' \in S} \sum_{a \in A} x(s', a) T(s', a, s) = \mu(s) \\ & \forall s \in S \\ & x(s, a) \geq 0 \quad \forall s \in S, \forall a \in A \end{aligned}$$

To give an interpretation to variables $x(s, a)$, we recall the two following propositions that relate feasible solutions of the dual linear program to stationary randomized policies in the MDP [14].

Proposition 1. *For a policy π , if x^π is defined as:*

$$x^\pi(s, a) = \sum_{t=0}^{\infty} \gamma^t p_t^\pi(s, a) \quad \forall s \in S, \forall a \in A \quad (4)$$

where $p_t^\pi(s, a)$ is the probability of reaching state s and choosing action a at step t , then x^π is a feasible solution of the dual linear program.

Proposition 2. *If $x(s, a)$ is a solution of the dual problem, then the stationary randomized policy δ^∞ , defined by:*

$$\delta(s, a) = \frac{x(s, a)}{\sum_{a \in A} x(s, a)} \quad \forall s \in S, \forall a \in A \quad (5)$$

defines values $x^{\delta^\infty}(s, a)$ as in Equation 4, that are equal to $x(s, a)$.

Thus, the basic solutions of the dual program $\mathcal{D}$ correspond to deterministic policies. Moreover, the basic solutions of the primal program $\mathcal{P}$ correspond to the value functions of deterministic policies. Those of randomized policies are in the convex hull of those basic solutions. Note that in an MDP, any feasible value function can be obtained with a randomized policy [14].

2.2 Multiobjective MDP

MDP has been extended to take into account multiple dimensions or criteria. A multiobjective MDP (MMDP) is defined as an MDP where the reward function has been replaced by:

- $R : S \times A \rightarrow \mathbb{R}^n$ where n is the number of criteria, $R(s, a) = (R_1(s, a), \dots, R_n(s, a))$ and $R_i(s, a)$ is the immediate reward for criterion i .

Now, a policy π is valued by a value function $V^\pi : S \rightarrow \mathbb{R}^n$, which gives the expected discounted total reward vector in each state. To compare the value of policies in a given state s , the basic model adopted in most previous studies [5, 17, 18] is *Pareto dominance* defined as follows: For any two policies π, π' , for any state s , $V^\pi(s)$ Pareto-dominates $V^{\pi'}(s)$, denoted $V^\pi(s) \succ_P V^{\pi'}(s)$ if and only if:

$$V^\pi(s) \neq V^{\pi'}(s) \text{ and } \forall i = 1 \dots n, V_i^\pi(s) \geq V_i^{\pi'}(s) \quad (6)$$

Due to the incompleteness of Pareto dominance, there may exist several optimal vectors in a given state. For a set $X \subset \mathbb{R}^n$, the set of

Pareto optimal vectors of X is defined by $M(X, \succ_P) = \{x \in X : \forall y \in X, \text{ not } y \succ_P x\}$.

Standard methods for MDPs can be extended to solve MMDPs although some problems remain open for methods based on dynamic programming, as noted by [18]. As it is needed later for the exposition of our results, we recall the linear program that has been proposed by [17] for finding non-dominated solutions in a MMDP.

The dual linear program formulated for MDPs can be extended to a multiobjective linear program as the dynamic of a MDP and that of a MMDP are identical. We only need to change the objective function of $\mathcal{D}$ by its vector counterpart. Thus we obtain the following multiobjective linear program $v\mathcal{D}$:

$$\begin{aligned} \text{v-max} \quad & \sum_{s \in S} \sum_{a \in A} R(s, a) x(s, a) \\ \text{s.t.} \quad & \sum_{a \in A} x(s, a) - \gamma \sum_{s' \in S} \sum_{a \in A} x(s', a) T(s', a, s) = \mu(s) \\ & \forall s \in S \\ & x(s, a) \geq 0 \quad \forall s \in S, \forall a \in A \end{aligned}$$

where v-max is a vector maximization with respect to Pareto dominance. Looking for all non-dominated solutions can be difficult and time-consuming as there could be many non-dominated solutions. In fact, there exists instances of problems where the number of solutions is exponential in the number of states. We illustrate this point by adapting an example proposed by [9].

Example 1. Let $N > 0$. Consider the following MMDP defined by (S, A, T, R) where $S = \{0, 1, \dots, N\}$, $A = \{a, b\}$, $\forall i = 1, \dots, N-1$, $T(i, a, i+1) = 1$, $T(i, b, i+1) = 1$, and the only non null reward vectors are defined by $R(i, a, i+1) = (2^i, 0)$, $R(i, b, i+1) = (0, 2^i)$. Here, we can take $\gamma = 1$ as state N is absorbing.

In this example, there are 2^{N+1} stationary deterministic policies. Stationary deterministic policies that only differ from one another on the choice of the action in the last state N have the same value functions as the reward and the transition in those states for both actions are identical. The remaining policies induce 2^N different valuation vectors, of the form $(x, 2^N - 1 - x)$ for $x = 0, 1, \dots, 2^N - 1$. Those different vectors are in fact all Pareto-optimal as they are on the line $x + y = 2^N - 1$. It is then infeasible in such a case to determine all non-dominated solutions.

Besides, in practice, one is generally only interested in one particular solution among all the non-dominated solutions that gives interesting tradeoffs between all the criteria. A more interesting approach for MMDP would be to directly search for that particular solution instead of finding first all the non-dominated solutions.

3 SEARCH FOR COMPROMISE SOLUTIONS

We introduce the notion of *scalarizing function* that will be used to discriminate between Pareto-optimal vectors in a given state. Formally, a scalarizing function is a function $\psi : \mathbb{R}^n \rightarrow \mathbb{R}$ that defines an overall value function $v : S \rightarrow \mathbb{R}$ from a vector value function $V : S \rightarrow \mathbb{R}^n$:

$$v(s) = \psi(V_1(s), \dots, V_n(s)) \quad (7)$$

3.1 Weighted Sum

The most straightforward choice for ψ seems to be the weighted sum. In this case, $v(s) = \sum_{i=1}^n \lambda_i V_i(s)$ where $\lambda_i > 0, \forall i = 1 \dots n$

so as to preserve the monotonicity with respect to Pareto dominance. By linearity of the mathematical expectation and the weighted sum, optimizing v is equivalent to solving the standard MDP obtained from the MMDP where the reward function is defined as: $r(s, a) = \sum_{i=1}^n \lambda_i R_i(s, a), \forall s, a$. In that case, an optimal stationary deterministic policy exists and standard solution methods can then be applied. However, using a weighted sum is not a good procedure for reaching balanced solutions as weighted sum is fully compensatory operator that does not encode the idea of balanced solutions. This is well illustrated by the two following examples:

Example 2. Consider the following MMDP with two criteria using the same valuation scale and characterized by (S, A, T, R) where $S = \{1\}$, $A = \{a, b\}$, $T(1, a, 1) = T(1, b, 1) = 1$, $R(1, a) = (1, 9)$ and $R(1, b) = (5, 5)$. In this MDP, there only exists two deterministic stationary policies depending on the choice of the action in state 1. They are thus denoted a and b respectively. Their value functions are given by: $V^a(1) = (1/(1-\gamma), 9/(1-\gamma))$ and $V^b(1) = (5/(1-\gamma), 5/(1-\gamma))$. If the rewards of the agents are simply summed (i.e. we give equal weights to each criterion), then both policies are optimal and have the same value functions. However, the policy choosing action b yields a much better balanced vector and should be preferred.

From the previous example, one could think that by choosing appropriate weights, one could reach a balanced solution. This is not the case. There exists instances where well-balanced solutions cannot be obtained by optimizing a weighted sum.

Example 3. Consider the MDP defined by (S, A, T, R) where $S = \{1\}$, $A = \{a, b, c\}$, $T(1, a, 1) = T(1, b, 1) = T(1, c, 1) = 1$, $R(1, a) = (1, 9)$, $R(1, b) = (4, 4)$, $R(1, c) = (9, 1)$. Here, there are three stationary deterministic policies. They are denoted a , b and c respectively. Their value functions are given by: $V^a(1) = (1/(1-\gamma), 9/(1-\gamma))$, $V^b(1) = (4/(1-\gamma), 4/(1-\gamma))$ and $V^c(1) = (9/(1-\gamma), 1/(1-\gamma))$. By giving equal weights to both criteria, the policy that chooses action a and the one choosing action c are equivalent. As soon as the weights are different, only one of those two policies is optimal and it therefore yields an unbalanced valuation vector. However, the policy choosing action b is not dominated by the other policies. As it yields a much better balanced valuation vector, one could arguably prefer that solution.

In fact, it is possible to reach balanced solutions that are even better than those corresponding to deterministic policies. Consider Figure 1 which represents value functions of deterministic policies in an initial state for a given MMDP ($n = 2$). If a weighted sum is used to combine the two criteria, the only solutions that can be found are those represented by points a , b and d because they are on the boundary of the convex hull of feasible vectors. However, one could argue that point c brings a better trade-off between the two criteria. Yet, if we consider randomized policies, we can do much better as shown in Figure 2.

The valuation vectors of randomized policies are in the convex hull of the valuation vectors of deterministic policies, represented by the greyed zone. The dark greyed zone represents all feasible valuation vectors that are preferred to point c . The dotted lines linking points a , b and d represent all Pareto-optimal valuation vectors. It is then easy to see that point c is dominated by all the randomized policies that are both in the dark greyed zone and on the dotted line. The next subsection explains how to reach such points.

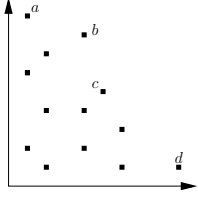


Figure 1. Valuation vectors

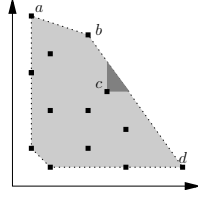


Figure 2. Better solutions

3.2 Tchebycheff Scalarizing Function

In multiobjective optimization, the question of finding balanced solutions among the non-dominated ones is a crucial issue. The standard way of generating compromise solutions in the Pareto-optimal set is resorting to the so-called reference point approach that consists in finding a feasible vector that minimizes a distance to a prescribed reference point [15, 19, 4]. The standard approach for that is to minimize the *Tchebycheff scalarizing function* defined for all $x \in \mathbb{R}^n$ by:

$$\psi(x, r, \lambda) = \max_{i=1 \dots n} \lambda_i |r_i - x_i| + \varepsilon \sum_{i=1 \dots n} \lambda_i |r_i - x_i| \quad (8)$$

where $r \in \mathbb{R}^n$ is the reference point, $\lambda \in \mathbb{R}^n$ is a positive weighting vector and ε is a positive real chosen arbitrarily small. The best compromise solution $V^{T*} : S \rightarrow \mathbb{R}^n$, called *Tchebycheff-optimal*, can then be computed with:

$$V^{T*} = \operatorname{argmin}_V \psi\left(\sum_{s \in S} \mu(s) V(s), r, \lambda\right) \quad (9)$$

where μ is a distribution probability over initial states. To apply this equation, we need to set properly those parameters. Generally, reference point r is taken as the *ideal point*. It can be computed with n different one-dimensional optimizations, that is we solve the MMDP as a standard MDP successively with reward function R_i for $i = 1 \dots n$. We denote V^{i*} the vectorial value function optimal for the i -th criterion. In a state s , the ideal point then can be formally defined as follows: $v_i^I = \sum_{s \in S} \mu(s) V_i^{i*}(s)$ for all $i = 1 \dots n$. With the V^{i*} 's, a second point can also be defined: $v_i^A = \sum_{s \in S} \mu(s) \min_{j=1 \dots n} V_i^{j*}(s)$ which is in fact an approximation of the Nadir point, i.e. a lower bound of the Pareto-optimal solutions. We use an approximation of the Nadir point as it is generally difficult to determine exactly [3]. Then the weights are defined as follows:

$$\lambda_i = \frac{w_i}{|v_i^I - v_i^A|} \quad (10)$$

where $w \in \mathbb{R}^n$ represents the weights of criteria. The construction of a compromise solution using Equations 8-10 is illustrated in Figure 3. The greyed zone represents the feasible solutions. The point I (resp. A) is the ideal point (resp. approximate Nadir point). The best compromise solution is the non-dominated point that is on the line linking I and A when using equal weights w_i . Actually, using Equation 10 with equal weights amounts to projecting point I on the Pareto-optimal frontier in the direction of A . The direction of projection can be modified with weights w_i to get other tradeoffs as illustrated on Figure 4.

As shown in [19], this way of constructing compromise solutions guarantees some nice properties. Contrary to the weighted

sum, here, any Pareto-optimal solution can be reached by minimizing the Tchebycheff scalarizing function with a proper choice of w . Moreover, any Tchebycheff-optimal solution is Pareto-optimal. These properties justified the use of Tchebycheff optimality in multiobjective path planning problems [6].

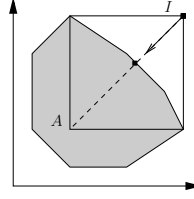


Figure 3. Compromise solution

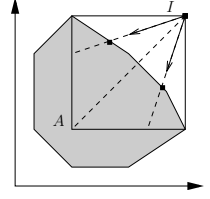


Figure 4. Different weights

Coming back to Example 3, the ideal point in state 1 is $(9/(1 - \gamma), 9/(1 - \gamma))$, the approximated Nadir point is $(1/(1 - \gamma), 1/(1 - \gamma))$ and λ can be taken $((1 - \gamma)/8, (1 - \gamma)/8)$. Then the Tchebycheff-optimal value, which is close to $1/2$ when ε is close to 0, is reached by the policy choosing action b . This simple example explains why the Tchebycheff scalarizing function is preferred to any weighted sum in multiobjective optimization.

As mentioned previously, the quality of compromise solutions found is even better if we consider randomized policies. In the next section, we present the problems that we need to overcome for finding Tchebycheff-optimal (possibly randomized) policies and we propose a solution method for compromise search in MMDPs.

4 SOLUTION METHOD

4.1 Tchebycheff Optimality is State-Dependent

In a standard MDP, an optimal policy is optimal in every initial state. However, here, the optimality notion based on the Tchebycheff function depends on the initial state, i.e. a best compromise policy in a given initial state may not be a best compromise solution in another state. We illustrate this point with a simple example:

Example 4. Consider an MMDP defined by (S, A, T, R) where $S = \{1, 2\}$, $A = \{a, b\}$, T and R are defined as follows: $T(1, a, 1) = 1$, $T(1, b, 2) = 1$, $T(2, a, 2) = 1$, $T(2, b, 2) = 1$ and $R(1, a) = (0, 6)$, $R(1, b) = (5, 0)$, $R(2, a) = (0, 5)$, $R(2, b) = (2, 2)$. Take $\gamma = 0.5$.

Point I in state 1 is $(7, 12)$, that in state 2 is $(4, 10)$. Point A is equal to $(0, 2)$ in state 1 and $(0, 4)$ in state 2. Thus, the weights are $(1/7, 1/10)$ in state 1 and $(1/4, 1/6)$ in state 2.

Now, the compromise solution in state 2 is given by $(2, 7)$, obtained by choosing actions a or b with probability 0.5. If we applied this strategy in state 2, then the best compromise in state 1 is obtained by choosing action a with probability $99/169$ and action b with probability $70/169$, which yields approximately $(3.515, 7.021)$. The distance to the ideal is equal to $99/199 \approx 0.498$.

However, if we had chosen to take action a in state 2, we could have obtained a better compromise solution in state 1. Indeed by choosing action a with probability $29/64$ and action b with probability $35/64$, we could have obtained a valuation vector roughly equals to $(3.536, 7.051)$ whose distance to the ideal point equals to $49/99 \approx 0.495$.

From this observation, we can conclude that the best compromise policy viewed from one state is not necessarily a best compromise solution viewed from another state.

Therefore, to use the Tchebycheff optimality, one needs to know the initial state. This is not, in our opinion, a very demanding requirement as for most problems, this information is available. Moreover, when the initial state is unknown, we can instead consider the average of a value function over all states.

4.2 Dynamic Inconsistency

Due to the non-linearity of the Tchebycheff scalarizing function, we cannot transform the MMDP into a standard MDP. More specifically, solving an MDP obtained by aggregating the vector rewards of an MMDP with the Tchebycheff scalarizing function is not equivalent to optimizing Equation 9 in the original MMDP.

As another consequence of the non-linearity of the scalarizing function, we can no longer rely on algorithms based on dynamic programming, such as value iteration. Indeed, the natural counterpart of value iteration for Tchebycheff optimization should be:

```

1:  $\forall s \in S, V(s) \leftarrow (0, \dots, 0)$ 
2: repeat
3:   for all  $s \in S$  do
4:     for all  $a \in A$  do
5:        $Q(s, a) \leftarrow R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V(s')$ 
6:     end for
7:      $V(s) \leftarrow \operatorname{argmin}_{a \in A} \psi(Q(s, a), v^I(s), \lambda(s))$ 
8:   end for
9: until convergence of  $V$ 

```

where ψ is defined by Equation 8. Let us apply this algorithm on an example.

Example 5. Consider the MMDP defined by (S, A, T, R) where $S = \{1, 2\}$, $A = \{a, b, c\}$, T and R are defined as follows: $T(1, a, 1) = 1$, $T(1, b, 1) = 1$, $T(1, c, 2) = 1$, $T(2, a, 2) = 1$, $T(2, b, 2) = 1$, $T(2, c, 2) = 1$ and $R(1, a) = (0, 5)$, $R(1, b) = (5, 0)$, $R(1, c) = (1, 1)$, $R(2, a) = R(2, b) = R(2, c) = (0, 0)$. Set $\gamma = 0.5$.

So, state 2 is an absorbing state and the only valuation vector that can be obtained in that state is $(0, 0)$. In state 1, the ideal point is defined by $(10, 10)$, point A by $(0, 0)$ and we set the weights to $(1/10, 1/10)$.

Initialize $v_0(1) = (0, 0)$. The value of applying action a in state 1 is given by $Q(1, a) = (0, 5)$. For action b , we get $Q(1, b) = (5, 0)$. And finally, for action c , we get $Q(1, c) = (1, 1)$. $Q(1, a)$ and $Q(1, b)$ are at a distance of about 1 from the ideal point while $Q(1, c)$ is at a distance of about 0.9 from the ideal point. If we choose action c , $v_1(1) = (1, 1)$. We can easily see that the next iteration would give the same computations. Therefore, the algorithm would converge in two steps and it gives as optimal valuation vector $(1, 1)$ for state 1. However, it is easy to see that choosing actions a or b with probability 0.5 would have yield much better rewards with respect to Tchebycheff optimality.

In this example, the ideal point has been computed at the infinite horizon. We could have also used instead the ideal point for a given horizon. It is easy to check the same problem occurs.

This last observation motivates us to propose a solving method based on linear programming.

4.3 Solving Method

Although the Tchebycheff norm is not linear, the dynamic of an MMDP remains identical to a standard MDP and thus is linear.

For finding Tchebycheff-optimal solutions, it is therefore possible to adapt the linear program proposed for finding Pareto-optimal solutions in a MMDP. Compromise solutions can be found with the following non linear program:

$$\begin{aligned}
\min \quad & \max_{i=1 \dots n} \lambda_i (\mu \cdot v_i^I - R_i \cdot x) + \varepsilon \sum_{i=1 \dots n} \lambda_i (\mu \cdot v_i^I - R_i \cdot x) \\
\text{s.t.} \quad & \sum_{a \in A} x(s, a) - \gamma \sum_{s' \in S} \sum_{a \in A} x(s', a) T(s', a, s) = \mu(s) \\
& \forall s \in S \\
& x(s, a) \geq 0 \quad \forall s \in S, \forall a \in A
\end{aligned}$$

where $\mu \cdot v_i^I = \sum_{s \in S} \mu(s) v_i^I(s)$ and $R_i \cdot x = \sum_{s \in S} \sum_{a \in A} R_i(s, a) x(s, a)$.

This program can easily be linearized as follows:

$$\begin{aligned}
\min \quad & z + \varepsilon \sum_{i=1 \dots n} \lambda_i (\mu \cdot v_i^I - R_i \cdot x) \\
\text{s.t.} \quad & z \geq \lambda_i (\mu \cdot v_i^I - R_i \cdot x) \quad \forall i = 1 \dots n \\
& \sum_{a \in A} x(s, a) - \gamma \sum_{s' \in S} \sum_{a \in A} x(s', a) T(s', a, s) = \mu(s) \\
& \forall s \in S \\
& x(s, a) \geq 0 \quad \forall s \in S, \forall a \in A
\end{aligned}$$

Our previous observation concerning the state-dependency of the optimality based on best compromise tells us that the Tchebycheff-optimal solution might change with μ , which differs from the classical case. When we do not know the initial state, distribution μ can be chosen as the uniform distribution over the possible initial states. When the initial state s_0 is known, $\mu(s)$ should be set to 1 when $s = s_0$ and to 0 otherwise. The solution found by the linear program does not specify which action to choose for the states that receive a null weight and that are not reachable from the initial state as they do not impact the value of the Tchebycheff-optimal policy.

5 EXPERIMENTAL RESULTS

We tested our solving method on the navigation problem over a grid $N \times N$. In this problem, the robot can choose among four actions: (L)eft, (U)p, (R)ight, (D)own. Figure 5 gives the transitions for action (R)ight. The whole transition function can then be obtained by symmetry.

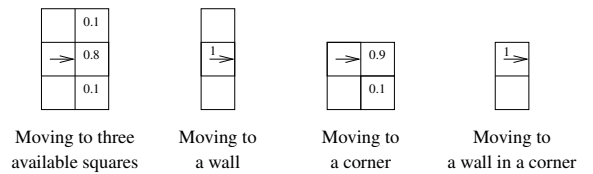


Figure 5. Transitions

Rewards are two-dimensional vectors whose components are randomly drawn within interval $[0, 1]$. The discount factor is set to 0.9 and the initial state is set arbitrarily to the upper left corner of the grid. We ran two series of experiments. As in real problems, criteria are generally conflicting. For the first set of experiments, to generate realist random instances, we simulate conflicting criteria with the following procedure: we pick one criterion randomly for each state and action and its value is drawn uniformly in $[0, 0.5]$ and the value of the other is drawn in $[0.5, 1]$. The results over 100 experiments are represented on Figure 6. One point on that figure (a dot when the weighted

sum is used and a circle for the Tchebycheff norm) represents the optimal value function in the initial state for one instance. Naturally, for some instances, the weighted sum can find a balanced solution. But, in most cases, the weighted sum gives a bad compromise solution. Figure 6 actually shows that we do not have any control on trade-offs obtained with a weighted sum. On the contrary, when using a Tchebycheff norm, the profile of the solutions are always balanced.

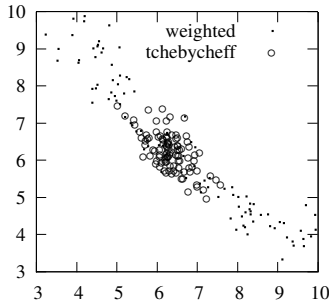


Figure 6. First experiments

To show the effectiveness of our approach, we ran a second set of experiments on pathological instances of the navigation problem. All the rewards are drawn randomly as for the first set of experiments. Then in the initial state, for each action, we choose randomly one of the criteria and add a constant (here, arbitrarily set to 5). Then by construction, the value functions of all deterministic policies in the initial state are unbalanced. Those value functions are represented on Figure 7 with dots as the weighted sum can only reach deterministic policies. Reassuringly, best compromise solutions found by minimizing the Tchebycheff norm are always well-balanced.

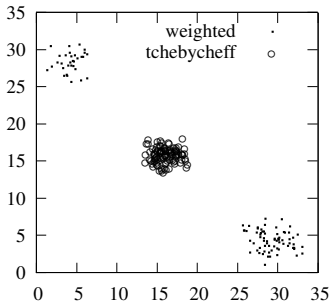


Figure 7. Second experiments

Finally, we give the average execution time in function of the problem size in Table 5. All the experiments were run using CPLEX 12.1 on a PC (Intel Core 2 Quad CPU 2.83Ghz) with 4Go of RAM. The first column n gives the number of objectives. The second column gives the number of states of the problem. Finally, column TW gives the execution time for the weighted sum approach while column TT gives the execution time for the Tchebycheff criterion. All the times are averages over 20 experiments and are given in seconds.

6 CONCLUSION

In this paper, we have presented the search of best compromise solutions in MMDPs with the use of a Tchebycheff distance. Although

n	Size	TW	TT
2	400	0.3	0.5
2	2500	6.1	13.8
2	10000	111.3	319.4
4	400	1.1	1.1
4	2500	6.1	24.6
4	10000	128.7	614.2
8	400	0.3	1.4
8	2500	6.2	42.8
8	10000	131.3	1115.4

Table 1. Average execution time in seconds

this is a non-linear criterion, we have provided a LP-solvable formulation of the problem. Experiments have shown the practical feasibility of the approach on difficult instances specially designed to exhibit conflicting criteria. In all experiments, the Tchebycheff criterion significantly outperforms the weighted sum concerning the quality of compromises found. Interestingly enough, this way of incorporating non-linear functions in MMDPs could be extended to other non-linear criteria. For instance, our approach can be applied to multiagent problems with a non-linear social-welfare function to determine policies that fairly share rewards among agents.

REFERENCES

- [1] C. Boutilier, 'Sequential optimality and coordination in multiagent systems', in *Proc. IJCAI*, (1999).
- [2] K. Chatterjee, R. Majumdar, and T.A. Henzinger, 'Markov decision processes with multiple objectives', in *STACS*, (2006).
- [3] M. Ehrgott and D. Tenfelde-Podehl, 'Computation of ideal and Nadir values and implications for their use in MCDM methods', *EJOR*, **151**, 119–139, (2003).
- [4] Matthias Ehrgott, *Multicriteria optimization*, Springer, 2005.
- [5] N. Furukawa, 'Vector-valued markovian decision processes with countable state space', *Ann. Math. Stat.*, **36**, (1965).
- [6] L. Galand and P. Perny, 'Search for compromise solutions in multiobjective state space graphs', in *ECAI*, pp. 93–97, (2006).
- [7] B. Golden and P. Perny, 'Infinite order lorenz dominance for fair multiagent optimization', in *AAMAS*, (2010).
- [8] C. Guestrin, D. Koller, and R. Parr, 'Multiagent planning with factored mdps', in *NIPS*, (2001).
- [9] P. Hansen, *Bicriterion Path Problems*, chapter Bicriterion Path Problems, 109–127, Springer, 1980.
- [10] P. E. Hart, N. J. Nilsson, and B. Raphael, 'A formal basis for the heuristic determination of minimum cost paths', *IEEE Trans. Syst. and Cyb.*, **4**(2), 100–107, (1968).
- [11] P. Kouvelis and G. Yu, *Robust discrete optimization and its applications*, Kluwer Academic Publisher, 1997.
- [12] L. Mandow and J.L. Pérez de la Cruz, 'A new approach to multiobjective A* search', in *IJCAI*, pp. 218–223, (2005).
- [13] A.I. Mouaddib, 'Multi-objective decision-theoretic path planning', in *IEEE Int. Conf. Robotics and Automation*, volume 3, pp. 2814–2819, (2004).
- [14] M.L. Puterman, *Markov Decision Processes - Discrete Stochastic Dynamic Programming*, John Wiley and Sons, 1994.
- [15] R.E. Steuer, *Multiple criteria optimization*, John Wiley, 1986.
- [16] B.S. Stewart and C.C. White III, 'Multiobjective A*', *J. ACM*, **38**(4), 775–814, (1991).
- [17] B. Viswanathan, V.V. Aggarwal, and K.P.K. Nair, 'Multiple criteria Markov decision processes', *TIMS Studies in the management sciences*, **6**, 263–272, (1977).
- [18] D.J. White, 'Multi-objective infinite-horizon discounted Markov decision processes', *Journal of mathematical analysis and applications*, **89**, 639–647, (1982).
- [19] A.P. Wierzbicki, 'On the completeness and constructiveness of parametric characterizations to vector optimization problems', *OR Spectrum*, **8**, 73–87, (1986).

Extracting Features from Ratings: The Role of Factor Models

Joachim Selke and Wolf-Tilo Balke¹

Abstract. Performing effective preference-based data retrieval requires detailed and preferentially meaningful structured information about the current user as well as the items under consideration. A common problem is that representations of items often only consist of mere technical attributes, which do not resemble human perception. This is particularly true for integral items such as movies or songs. It is often claimed that meaningful item features could be extracted from collaborative rating data, which is becoming available through social networking services. However, there is only anecdotal evidence supporting this claim; but if it is true, the extracted information could be very valuable for preference-based data retrieval. In this paper, we propose a methodology to systematically check this common claim. We performed a preliminary investigation on a large collection of movie ratings and present initial evidence.

1 INTRODUCTION

Recommender systems [1, 17] are one of the most prominent applications of preference handling technology [6] and a highly active area of research. In particular, fueled by the Netflix competition and its one million dollar prize money [2], research on collaborative recommendation techniques [21] has recently made significant advances, most notably through the introduction of *factor models* [16, 22].

In collaborative recommender systems, *users* repeatedly express their preferences for *items*, which usually is done by giving explicit *ratings* on some predefined numerical scale. This data can be modeled using a *rating matrix*, whose rows correspond to items, columns to users, and entries to ratings. Typically, ratings matrices are very sparse, that is, only a small fraction of all possible ratings have actually been observed. Personalized recommendations are generated by predicting unobserved ratings from the available data and, for each user, selecting those items considered to be most appealing.

Most state-of-the-art collaborative recommendation methods—including the winner of the Netflix Prize—are based on factor models, which are known to yield much more accurate predictions than traditional neighborhood-based methods [14, 15, 22, 23, 24]. In factor models, each user and each item is represented by a vector in some shared real coordinate space. The vectors are chosen such that each observed rating is closely approximated by the dot product of the corresponding item and user vectors. The selection of coordinates usually is formalized as an optimization problem. Predictions for unobserved ratings are generated by computing the respective scalar products. Equivalently, this approach can be seen as a factorization of the rating matrix into the product of an item matrix (whose rows are the item vectors) and a user matrix (whose columns are the user vectors).

The success of factor models is usually attributed to the intuition that the coordinate space used to represent items and users actually is a *latent feature space*. That is, its dimensions capture the items' perceptual properties as well as the users' preference judgments regarding these properties. For example, when items are movies, the individual dimensions are generally thought to measure (more or less) "obvious" features such as horror vs. romance, the level of sophistication, or orientation towards adults. For users, each coordinate is thought to describe the relative degree of importance attached to the respective dimension. This understanding of factor models can be found throughout the literature, for example, in [2, 15, 16, 18, 23].

Although it is intuitively appealing, to our knowledge, the correspondence to features has never been systematically proven, but is only reported anecdotally. For example, Koren et al. [16] performed a factorization on the Netflix movie data set and manually interpreted the first two coordinates for selected movies as follows:

Someone familiar with the movies shown can see clear meaning in the latent factors. The first factor has on one side lowbrow comedies and horror movies, aimed at a male or adolescent audience, while the other side contains drama or comedy with serious undertones and strong female leads. The second factorization axis has independent, critically acclaimed, quirky films on the top, and on the bottom, mainstream formulaic films.

Further evidence has been provided by Takács et al. [23]. After performing a factorization of the Netflix data set, they manually assigned labels to individual dimensions of their coordinate space, such as *Legendary*, *Typical for men*, *Romantic*, and *NOT Monty Python*.

In this paper, we propose a *systematic* method for studying the coordinate spaces derived from factor models and apply it to the MovieLens 10M data set, a large real-world collection of movie ratings. The main contribution of our work consists in laying important groundwork, on which further research in recommender systems and preference handling can be built. In particular, we see two concrete directions for future work:

- First, knowing what kind of semantic information is extracted by factor models—and how it is represented in coordinate spaces—will enable a deeper understanding of these methods. Ultimately, these findings may lead to a more systematic development and refinement of recommender systems. In particular, a systematic assessment of semantic structures provides an additional way of evaluating the effectiveness of factor-based recommenders. This would perfectly complement traditional evaluation methods [11], which focus on predictive accuracy.
- Second, we believe that factor models might be a powerful tool for automatically extracting meaningful descriptions of otherwise

¹ Institut für Informationssysteme, Technische Universität Braunschweig, Germany

hard-to-describe items such as movies or songs—particularly, essential features of movies cannot be characterized at all by purely technical features such as runtime, language, or release date.² But given a coordinate representation of movies that matches human perception, the full machinery developed in preference handling research can be applied [6, 9]. For example, clustering techniques can give user an initial high-level impression of the available items, item rankings can be learnt from ordinal preference statements [10] or utilities [5], and the best items can be retrieved by means of Top-k algorithms [12].

Since our primary research interest lies in applying preference-based retrieval techniques to item collections, in this paper we will concentrate on evaluating the semantic structures contained in the item matrix A . Performing a similar analysis of the user matrix B may require entirely different methods.

The paper is structured as follows: After introducing notation and reviewing the most important factor models, we develop general guidelines on how to evaluate coordinate spaces for semantic information. Then, we illustrate how to apply these guidelines to the evaluation of factor spaces generated from movie rating data and perform experiments on the MovieLens 10M data set.

2 PRELIMINARIES

In the following, we use the variables i and j to identify items, whereas u and v denote users. We are dealing with ratings given to I items by U users. Let $R = (r_{i,u}) \in \{\mathbb{R} \cup \emptyset\}^{I \times U}$ be the corresponding rating matrix, where $r_{i,u} = \emptyset$, if item i has not been rated by user u ; otherwise, $r_{i,u}$ expresses the strength of user u 's preference for item i . Ratings are usually limited to a fixed integer scale (for example, one to ten stars). Moreover, $\mathcal{R} = \{(i, u) \mid r_{i,u} \neq \emptyset\}$ is the set of all item–user pairs for which ratings are known. Let n be the total number of ratings observed (the cardinality of $\mathcal{R}$). Typically, n is very small compared to the number of possible ratings $I \cdot U$ (for example, in the Netflix data set it is $\frac{n}{I \cdot U} \approx 1.4\%$).

Given some target dimensionality d , the basic idea underlying factor models is to find matrices $A = (a_{i,r}) \in \mathbb{R}^{I \times d}$ and $B = (b_{r,u}) \in \mathbb{R}^{d \times U}$ such that their product $\hat{R} = A \cdot B$ closely resembles R on all known entries. To quantify this notion of “close resemblance,” the sum of squared errors (SSE) is popularly chosen. The SSE difference between the rating matrix R and its estimation $\hat{R} = (\hat{r}_{i,u})$ is defined as

$$\text{SSE}(R, \hat{R}) = \sum_{(i,u) \in \mathcal{R}} (r_{i,u} - \hat{r}_{i,u})^2.$$

Factor models are typically formulated as optimization problems over A and B , in which the SSE (or some other measure) is to be minimized.

Probably the most popular factor model is Brandyn Webb's regularized SVD model [16, 18], in which A and B are defined as the solution of the least squares problem

$$\min_{A,B} \text{SSE}(R, A \cdot B) + \lambda \sum_{(i,u) \in \mathcal{R}} \sum_{r=1}^d (a_{i,r}^2 + b_{r,u}^2).$$

Here, $\lambda \geq 0$ is a regularization constant used to avoid overfitting.

More advanced versions of the SVD model exclude systematic rating deviations from the factorization and model them explicitly

using new variables. Bell and Koren [3] propose to estimate rating $r_{i,u}$ by

$$\hat{r}_{i,u} = \mu + \delta_i + \delta_u + \sum_{r=1}^d a_{i,r} b_{r,u},$$

where the constant μ denotes the mean of all observed ratings; δ_i and δ_u are $I + U$ new model parameters expressing systematic item and user deviations from μ . Again, the parameters are chosen according to a regularized least squares problem:

$$\min_{A,B,\delta_*} \text{SSE}(R, \hat{R}) + \lambda \sum_{(i,u) \in \mathcal{R}} \left(\sum_{r=1}^d (a_{i,r}^2 + b_{r,u}^2) + \delta_i^2 + \delta_u^2 \right).$$

The rationale underlying this approach—which we refer to as δ -SVD in the following—is that the removal of item- and user-specific general trends from the factorization allows to focus on more sophisticated rating patterns.

The third basic factor model being relevant to our work performs a non-negative factorization of the rating matrix [23]. It is identical to the regularized SVD model up to the additional constraint that all entries of A and B must be non-negative. Extending this model by explicit item and users deviations is not reasonable since this would require negative entries in A and B to approximate R close enough. The non-negative matrix factorization model aims at creating a coordinate space in which effects of different dimensions on the estimated ratings cannot cancel out each other. Henceforth, we refer to this model as NNMF.

3 EVALUATING COORDINATE SPACES

Given an item–feature matrix $A \in \mathbb{R}^{I \times d}$ generated by some factor model, how can we determine whether the items' coordinates in this d -dimensional space resemble a “semantically meaningful” pattern? The most straightforward approach consists in extending and systematizing the casual investigations described in the introduction. This could easily be done by presenting the item coordinate space to a number of different people and asking them to label its dimensions. The correspondence between the generated item coordinates and human perception could, for example, be done by measuring the degree of consensus among people or the average time needed to come up with adequate labels.

Although this kind of investigation seems very reasonable, it contains some severe flaws, which cannot be fixed by careful study design:

1. The dimensionality chosen in most applications of factor models typically ranges between $d = 10$ and $d = 100$. A comprehensive analysis of the resulting data sets would require the users to comprehend high-dimensional spaces, which is impossible even when using advanced visualization techniques.
2. Due to hindsight bias, given enough time, users will be able to assign a fitting label to almost any dimension of the coordinate space. Chances are good that this effect accounts for rather questionable labels such as *NOT Monty Python*.
3. By using free association to name dimensions, the collection of resulting labels tend to show a high variability and reflect individual differences between users. To produce statistically significant results, either the sample size must be extended (which requires more study participants and results in higher costs), or the variability must be reduced, for example, by training participants to use an established domain-specific vocabulary to articulate the semantic properties they recognize in the data (which also increases time and effort).

² A complementary approach to closing this *semantic gap* is content-based image and video retrieval [8].

4. Typically, there are many near-optimal solutions to the above mentioned optimization problems, which can be transformed into one another by rotation of the coordinate axes. This is because, for any invertible matrix $M \in \mathbb{R}^d$, the solution pairs (A, B) and $(AM, M^{-1}B)$ produce the same SSE. Although regularization usually enforces the theoretical existence of a unique optimal solution pair, in practice the enormous problem size often allows only finding one of the many near-optimal solutions. Consequently, the direction of the coordinate axes is completely arbitrary, which makes the task of assigning labels a hopeless undertaking.

3.1 Some Guidelines

In this section, we devise a set of guidelines on which to base more appropriate approaches to the analysis of coordinate spaces.

- In the view of problems (1) and (4), we recommend to avoid any direct human interaction with *item coordinates*. Instead, human input should concentrate on describing *item properties*, which in turn are related to coordinates as well as compared by algorithmic means.
- The only effective way to eliminate hindsight bias (2) is collecting feedback on items before generating and presenting any information extracted by the factor models under consideration.
- To resolve problem (3), we primarily recommend to adapt a domain-specific vocabulary to allow a structured description of items. For example, to characterize music, the rich vocabulary developed by allmusic³ seems appropriate; amongst others, it includes very detailed information about genres, styles, moods, and connections between artists. Since this kind of semantic information can be (or already have been) provided by a small number of experts and usually is little prone to debate, it is easy to assemble and work with. In later stages of analysis, unrestricted user feedback may be included to reveal the position and extent of more fine-grained and rather subjective concepts in the coordinate space.

We also propose to apply a standardization procedure to the generated coordinate space. This is for the following reasons: First, recall that, for any invertible matrix $M \in \mathbb{R}^d$, the solution pairs (A, B) and $(AM, M^{-1}B)$ are equivalent; to enable comparisons between different factor models and even different runs of the same optimization algorithms, we need to define one solution pair as the standard representation. Second, to enable a better separation of different effects in the data, the axes of the item (and user) coordinate space should be chosen to be orthogonal. Moreover, axes should be ordered according to their relative importance (measured by the variance of data along each axis); that is, the first dimension should be assigned to the most important axis.

The perfect tool for matching these requirements is the singular value decomposition, a well-known matrix factorization technique from linear algebra, which inspired the SVD factor model. It is based on the fact that, for any rank- d matrix $X \in \mathbb{I} \times \mathbb{U}$, there is a column-orthonormal matrix $U \in \mathbb{R}^{\mathbb{I} \times d}$, a diagonal matrix $S \in \mathbb{R}^{d \times d}$, and a row-orthonormal matrix $V \in \mathbb{U} \times \mathbb{U}$ such that $X = USV$. By reordering rows and columns, S can be chosen such that its diagonal elements are ordered by increasing magnitude. Moreover, the diagonal matrix S can be eliminated from this factorization by setting $X = U'V'$, where $U' = US^{\frac{1}{2}}$ and $V' = VS^{\frac{1}{2}}$. The matrices U' and V' are unique if all diagonal elements of S have been mutually different.

³ <http://www.allmusic.com>

In our setting, we will apply the singular value decomposition to transform the product $X = A \cdot B$ into a new product $A' \cdot B'$ as just described. Since rating data tends to be very “noisy,” we can safely assume that (A', B') is a unique representation of (A, B) ; we did not encounter any counterexamples during our experiments on large real-world rating data. Moreover, any equivalent pair $(AM, M^{-1}B)$ also gets transformed into (A', B') , which we define as the corresponding standard representation. It can be computed efficiently using the product decomposition algorithm proposed in [7, Sec. 3].

3.2 Use Case: Movie Ratings

Based on these guidelines, we now present a concrete method for performing a basic evaluation of coordinate spaces generated from movie ratings. Our focus rests on immediate applicability, so we relate the item coordinates to reference data that is already available.

The reference source for all kinds of movie-related information is IMDb, the Internet Movie Database⁴, which currently covers about 1.6 million titles. Most of IMDb’s data has been created with the help of its users. Therefore, a large proportion of the available content can freely be downloaded and used for non-commercial purposes⁵. Based on this comprehensive data, one should be able to cross-reference any collection of movie ratings with IMDb.

For the semantic evaluations we are going to perform, the following attributes of titles may prove helpful: genres, certifications (e.g., USA:PG for *parental guidance suggested*), year of release, and plot keywords. To illustrate the general procedure, we will only exploit genre information in this paper. Extending our method to other types of semantic information is straightforward. Checking the correspondence between genres and item coordinates also makes up a good first test of whether at least some basic semantic properties of movies are represented in coordinate spaces, which is exactly the purpose of the current work.

IMDb recognizes 28 different genres, from *Action* to *Western*, where each movie may belong to multiple genres. The assignment of genres is done by IMDb’s expert staff in cooperation with IMDb users. To enforce consistency, this process is based upon a collection of publicly available guidelines⁶. Therefore, this data source matches the requirements developed in the previous section.

To analyze whether the distribution of genres in coordinate space displays any significant pattern, we turn to established classification algorithms, which explicitly have been designed to exploit any relevant patterns in the data if there are any. In particular, we propose to measure the degree of adherence to a pattern by the classification accuracy shown by these algorithms when predicting the genre of movies based on their coordinates. In essence, we transform our analysis into a sequence of binary classification problems (one for each genre), which enables us to build on solid grounds. Following the common methodology, we use cross-validation; that is, accuracy is measured on a data set, which is independent of the one used to train the classifier. By applying proven techniques to counter overfitting, our approach also overcomes any possible problems related to hindsight bias.

For a start, we selected two popular classification algorithms, which are able to detect different kinds of patterns in the data: support vector machines and kNN-classifiers.

Support vector machines will be used in two different flavors: first, using a linear kernel (referred to as SVM-lin), and second, using a

⁴ <http://www.imdb.com>

⁵ <http://www.imdb.com/interfaces#plain>

⁶ <http://www.imdb.com/updates/guide/genres>

Gaussian radial basis function kernel (SVM-RBF). Linear support vector machines will show a high classification accuracy if most movies of the respective genre are grouped at one side of the data set, which can be separated from all remaining movies by a hyperplane. For example, this can be used to disprove the hypothesis that there exists a direction in the coordinate space along which, say, the amount of action, increases monotonically. In contrast, the SVM-RBF classifier detects whether groups of movies with the same genre tend to be located in close vicinity.

kNN-classifiers perform well if the distance between movies having the same genre typically is smaller than the distance to movies not having this genre. Therefore, they can be used to check whether genres form spatially separated patterns in coordinate space. Since factor models are not based on a notion of proximity, it is not clear what measure of distance suits factor models best. We will try out the following four measures: Euclidean distance, standardized Euclidean distance (where, to ensure equally weighted dimensions, coordinate values are divided by the standard deviation of the data with respect each dimension), negative scalar product (which essentially adapts the method of rating prediction to measure distance), and cosine similarity (which is monotonically related to the angle between two vectors).

To evaluate the true benefit of coordinate spaces generated from factor models, we propose the following baseline, which is derived from traditional neighborhood-based recommendation methods [20] and constructed as follows: First, for any items i and j , we compute their Pearson correlation coefficient

$$\varrho_{i,j} = \frac{\sum_{u \in \mathcal{R}_{i,j}} (r_{i,u} - \mu_{i,j})(r_{j,u} - \mu_{j,i})}{\sqrt{\sum_{u \in \mathcal{R}_{i,j}} (r_{i,u} - \mu_{i,j})^2} \sqrt{\sum_{u \in \mathcal{R}_{i,j}} (r_{j,u} - \mu_{j,i})^2}},$$

where $\mathcal{R}_{i,j}$ is the set of all users who rated both i and j , and $\mu_{i,j}$ is the mean rating given to item i by users who rated both i and j . If $\mathcal{R}_{i,j}$ is empty, then $\varrho_{i,j}$ is undefined. The Pearson correlation coefficient $\varrho_{i,j}$ measures the tendency of users to rate items i and j similarly. To avoid biased estimates in cases where $n_{i,j} = |\mathcal{R}_{i,j}|$ is very small, we derive a new measure of similarity

$$s_{i,j} = \frac{n_{i,j}}{n_{i,j} + \lambda} \cdot \varrho_{i,j}$$

from $\varrho_{i,j}$ by shrinking towards zero [15]. Here, $\lambda \geq 0$ is a regularization parameter. Finally, we carry over these similarity into distances by applying a logarithmic transformation:

$$d_{i,j} = -\ln \left(\frac{1 + s_{i,j}}{2} \right).$$

To derive a d -dimensional coordinate space in which items i and j approximately have distance $d_{i,j}$, we use metric multidimensional scaling [4]. Since neighborhood-based recommendation methods are usually outperformed by factor models, we expect our baseline coordinate space to be far inferior to those constructed using factor models. We refer to our baseline model as MDS.

4 EXPERIMENTS ON MOVIELENS 10M

We applied our approach to the MovieLens 10M data set⁷, which consists of about 10 million ratings collected by the online movie recommender service MovieLens⁸. After postprocessing the original data (removing one non-existing movie, merging several duplicate

movie entries, and removing movies that received less than 20 ratings), our new data set consists of 9,984,419 ratings of 8938 movies provided by 69878 users. The ratings use a 10-point scale from 0.5 (worst) to 5 (best). Each user contributed at least 14 ratings.

Our analysis requires the genre information maintained by IMDb, so we had to map each movie in the data set to its corresponding IMDb entry. This task has been simplified a lot by the fact that all items in the MovieLens 10M data set are relatively well-known movies developed for cinema.⁹ We mapped about 8000 movies automatically by comparing titles and release years; the remaining movies have been assigned manually or semi-automatically.

To avoid the problem of learning from very small samples for now, we did not use all 28 genres distinguished by IMDb. Instead, we take only those genres into consideration that have been assigned to at least 5% of all movies in our data set. Table 1 lists all remaining 13 genres and their relative frequencies. On average, 2.3 genres have been assigned to each movie.

Genre	%	Genre	%
Action	16.0	Horror	10.1
Adventure	12.7	Mystery	9.1
Comedy	38.2	Romance	25.2
Crime	16.6	Sci-Fi	8.6
Drama	54.6	Thriller	24.2
Family	8.4	War	5.2
Fantasy	8.3		

Table 1. Relative frequencies of genres.

4.1 Generating Coordinate Spaces

We implemented each of the four coordinate extraction methods in MATLAB and executed them on our rating data.

For SVD, δ -SVD, and NNMF, we followed the literature and used an optimization procedure based on gradient descent; to reduce computation time, we applied the Hessian speedup proposed in [19]. Adapting the common methodology, we chose the regularization parameter λ by cross-validation such that the SSE is minimized on randomly chosen test sets. We ended up with a value of $\lambda = 0.04$ for each of the three algorithms.

Since optimization by gradient descent is known to get stuck in local extrema of the function to be minimized, we ran the three procedures at least three times, each with different initial coordinates, which have been chosen randomly. For each result, we computed the standardized solution pair as described in the previous section. We found that the solutions generated by each extractor do not differ significantly after standardization. This indicates that our coordinate spaces match the unique solution of each optimization problem.

For our MDS procedure, we used the regularization constant $\lambda = 20$, which we determined by adapting the recommendation Koren gave for the Netflix data set [15]. The coordinates have been generated by MATLAB's `mdscale` function using the metric stress criterion. Since in our data set about 14 percent of all movie–movie pairs had no raters in common, we treated the respective entries of the distance matrix as missing data.

To measure the effect of dimensionality, we generated three different coordinate spaces with each extractor by varying the parameter d . We chose $d = 10$, $d = 50$, and $d = 100$.

⁷ <http://www.grouplens.org/node/73>

⁸ <http://www.movielens.org>

⁹ This is the reason why we did not consider the Netflix data set. It consists of all kinds of DVD titles, which often lack a clear correspondence in IMDb.

4.2 Applying the Classifiers

In total, we used 14 different classifiers to evaluate each of the 12 coordinate spaces with respect to each of the 13 genres.

We implemented the two support vector machine classifiers by soft-margin SVMs with parameters $C = 4$ and (for SVM-RBF) $\gamma = 0.1$, which have been determined by cross-validation to maximize classification accuracy.

Each of the four different kNN-classifiers will be applied to the data sets with three different choices of k . To measure whether movies of the same genre tend to occur in larger groups, we chose $k = 1$, $k = 3$, and $k = 9$. In the following, we will refer to these 12 classifiers as k NN-Eucl, k NN-sEucl, k NN-scal, and k NN-cos.

To enable comparisons among classifiers and data sets, we generated 20 pairs of training and test sets, each by randomly choosing 40% of all movies for training and 10% (of the remaining movies) for testing. For each of the resulting 2184 combinations of coordinate spaces, classifiers and genres, we use the same 20 pairs of item sets for training and testing. In each case, we measured the classification accuracy. All results reported below are averages over the 20 runs.

4.3 Results

Probably the most popular way of assessing a classifier’s performance is measuring its accuracy, that is, the fraction of test items which have been classified correctly. However, in our setting, this measure is not very helpful. To see this, recall that the relative frequency of genres is very different in our data set. For example, over half of all movies belong to the genre *Drama*, but there are only about 5% *War* movies. While attaining an accuracy of 95% would be significant for the genre *Drama*, it can easily be achieved for the genre *War* just by classifying any movie as *non-War*. To enable comparisons across genres, we propose to use a modified version of Cohen’s kappa measure.

Any result of a binary classification task can be described by four numbers, which sum up to 1: the fraction of true positives (α_{tp}), the fraction of false positives (α_{fp}), the fraction of false negatives (α_{fn}), and the fraction of true negatives (α_{tn}). Accuracy is defined as $acc = \alpha_{tp} + \alpha_{tn}$. Moreover, the accuracy of a static majority-based classifier (which always returns the label of the more frequent class) is $acc_{maj} = \max\{\alpha_{tp} + \alpha_{fn}, \alpha_{fp} + \alpha_{tn}\}$. We propose to use this kind of naive classifier for normalizing the accuracy and define $\kappa = (acc - acc_{maj}) / (1 - acc_{maj})$. This measure expresses a classifier’s relative performance with respect to the majority-based classifier. If $acc = 1$ then $\kappa = 1$, if $acc > acc_{maj}$, then $\kappa > 0$, if $acc = acc_{maj}$, then $\kappa = 0$, and if $acc < acc_{maj}$, then $\kappa < 0$.

By measuring accuracy in terms of κ , we can average classification performance over different genres. Tables 2–5 report the mean κ s over all 260 classification results obtained for each combination of coordinate space and classifier type. All entries larger than 0.10 have been marked in boldface. We can observe the following:

- The coordinate space derived by NNMF does not contain much helpful information about genres that can be exploited by our classifiers. The performance in all other spaces is significantly better.
- Except for NN-sEucl, classification performance generally improves with increasing dimensionality. However, the difference in performance between $d = 10$ and $d = 50$ is much larger than the one between $d = 50$ and $d = 100$. This indicates that our ordering of dimensions during standardization indeed captures some notion of relative importance. This is probably also the reason for NN-sEucl’s decreasing performance with growing d ; treating all

	SVD-10	SVD-50	SVD-100
SVM-lin	0.08	0.18	0.20
SVM-RBF	0.15	0.23	0.25
1NN-Eucl	−0.24	−0.21	−0.19
3NN-Eucl	0.01	0.05	0.04
9NN-Eucl	0.12	0.16	0.14
1NN-sEucl	−0.25	−0.27	−0.31
3NN-sEucl	0.01	0.00	−0.06
9NN-sEucl	0.12	0.12	0.04
1NN-scal	−0.42	−0.30	−0.30
3NN-scal	−0.16	−0.03	−0.03
9NN-scal	0.01	0.11	0.12
1NN-cos	−0.25	−0.18	−0.16
3NN-cos	0.00	0.06	0.06
9NN-cos	0.12	0.17	0.16

Table 2. Kappas for coordinates generated by SVD.

	δ -SVD-10	δ -SVD-50	δ -SVD-100
SVM-lin	0.07	0.16	0.18
SVM-RBF	0.13	0.20	0.23
1NN-Eucl	−0.26	−0.26	−0.26
3NN-Eucl	−0.01	0.01	−0.02
9NN-Eucl	0.11	0.12	0.08
1NN-sEucl	−0.26	−0.29	−0.36
3NN-sEucl	0.00	−0.03	−0.11
9NN-sEucl	0.11	0.09	−0.01
1NN-scal	−0.41	−0.28	−0.22
3NN-scal	−0.06	0.02	0.06
9NN-scal	0.05	0.13	0.16
1NN-cos	−0.26	−0.19	−0.16
3NN-cos	0.00	0.07	0.09
9NN-cos	0.12	0.18	0.19

Table 3. Kappas for coordinates generated by δ -SVD.

	NNMF-10	NNMF-50	NNMF-100
SVM-lin	0.02	0.05	0.11
SVM-RBF	0.02	0.09	0.14
1NN-Eucl	−0.56	−0.47	−0.41
3NN-Eucl	−0.20	−0.16	−0.13
9NN-Eucl	−0.02	0.01	0.02
1NN-sEucl	−0.56	−0.47	−0.45
3NN-sEucl	−0.20	−0.16	−0.16
9NN-sEucl	−0.02	0.01	0.00
1NN-scal	−0.37	−0.34	−0.34
3NN-scal	−0.11	−0.10	−0.09
9NN-scal	−0.02	0.00	0.02
1NN-cos	−0.56	−0.45	−0.41
3NN-cos	−0.20	−0.15	−0.13
9NN-cos	−0.03	0.02	0.03

Table 4. Kappas for coordinates generated by NNMF.

	MDS-10	MDS-50	MDS-100
SVM-lin	−0.16	0.15	0.19
SVM-RBF	0.03	0.16	0.17
1NN-Eucl	−0.29	−0.19	−0.18
3NN-Eucl	−0.01	0.06	0.06
9NN-Eucl	0.13	0.18	0.18
1NN-sEucl	−0.29	−0.23	−0.29
3NN-sEucl	−0.01	0.05	−0.01
9NN-sEucl	0.13	0.17	0.12
1NN-scal	−0.29	−0.19	−0.18
3NN-scal	−0.01	0.07	0.08
9NN-scal	0.12	0.18	0.18
1NN-cos	−0.28	−0.18	−0.16
3NN-cos	0.00	0.07	0.08
9NN-cos	0.13	0.19	0.19

Table 5. Kappas for coordinates generated by MDS.

dimensions equally seems to overweight information from dimensions at the end of the list.

- The SVM-RBF classifier slightly outperforms SVM-lin, but is comparable in performance to 9NN-Eucl, 9NN-scal, and 9NN-cos. This indicates that genres indeed tend to cluster in coordinate spaces, even with respect to different measures of distance.
- The NN-classifiers display bad performance for $k = 1$ and $k = 3$, which indicates that, although movies of the same genre roughly occur in clusters, each cluster usually also contains movies that do not have assigned the respective genre.
- In contrast to our expectations, the performance in coordinate spaces generated by factor models is comparable to the performance shown on our baseline coordinate space MDS.

Moreover, the results suggest that the performance of k NN-classifiers might even further increase for larger values of k . To check this, we performed some preliminary tests with $k \approx 20$, but have not been able to confirm this conjective.

We also investigated the influence of individual genres on classification performance; as an example, the results for SVM-RBF are reported in Table 6. Entries larger than 0.20 have been indicated. We can see that some genres, such as *Horror* and *Drama*, can clearly be identified by the classifier, while others cannot. We have expected much better performance on clear-cut genres such as *War*.

	SVD-100	δ -SVD-100	NNMF-100	MDS-100
Action	0.34	0.31	0.22	0.22
Adventure	0.13	0.12	0.08	0.00
Comedy	0.45	0.42	0.25	0.42
Crime	0.08	0.06	-0.01	0.00
Drama	0.47	0.43	0.37	0.44
Family	0.43	0.46	0.31	0.34
Fantasy	0.03	0.05	0.01	0.00
Horror	0.56	0.54	0.31	0.61
Mystery	0.06	0.04	-0.00	0.00
Romance	0.11	0.10	-0.00	0.00
Sci-Fi	0.23	0.20	0.09	0.00
Thriller	0.31	0.27	0.14	0.15
War	0.05	0.06	-0.00	0.00

Table 6. Kappas for SVM-RBF by genre.

In summary, these preliminary experiments suggest that the coordinate spaces derived by SVD, δ -SVD, and MDS indeed contain some significant semantic information about the represented movies. However, the situation is by far not as clear as claimed by the literature.

5 CONCLUSION AND OUTLOOK

In the current paper, we presented a general methodology for systematically analyzing whether coordinate spaces generated from factor models contain semantic information, as it is commonly claimed. We applied our approach to the MovieLens 10M data set and found initial evidence for this claim.

Our results encourage us to follow this line of research in several ways. First, we would like to investigate whether our results also carry over to more advanced and complex factor models, which have been proposed very recently [13, 15]. It would also be interesting to see what more traditional methods such as multidimensional scaling can contribute to the problem of feature extraction from rating data, since our results indicate that these methods can successfully be modified for use in our new setting.

References

- [1] Gediminas Adomavicius and Alexander Tuzhilin, ‘Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions’, *IEEE Transactions on Knowledge and Data Engineering*, **17**(6), 734–749, (2005).
- [2] Robert M. Bell, Jim Bennett, Yehuda Koren, and Chris Volinsky, ‘The million dollar programming prize’, *IEEE Spectrum*, **46**(5), 28–33, (2009).
- [3] Robert M. Bell and Yehuda Koren, ‘Scalable collaborative filtering with jointly derived neighborhood interpolation weights’, in *Proceedings of ICDM 2007*, pp. 43–52. IEEE Computer Society, (2007).
- [4] Ingwer Borg and Patrick J. F. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, Springer, second edn., 2005.
- [5] Craig Boutilier, Kevin Regan, and Paolo Viappiani, ‘Preference elicitation with subjective features’, in *Proceedings of RecSys 2009*, pp. 341–344. ACM, (2009).
- [6] Ronen I. Brafman and Carmel Domshlak, ‘Preference handling: An introductory tutorial’, *AI Magazine*, **30**(1), 58–86, (2009).
- [7] Zlatko Drmač, ‘Accurate computation of the product-induced singular value decomposition with applications’, *SIAM Journal on Numerical Analysis*, **35**(5), 1969–1994, (1998).
- [8] Peter Enser and Christine Sandom, ‘Towards a comprehensive survey of the semantic gap in visual image retrieval’, in *Proceedings of CIVR 2003*, volume 2728 of *LNCS*, pp. 291–299. Springer, (2003).
- [9] Johannes Fürnkranz and Eyke Hüllermeier, ‘Preference learning’, *Künstliche Intelligenz*, **2005**(1), 60–61, (2005).
- [10] Ralf Herbrich, Thore Graepel, and Klaus Obermayer, ‘Large margin rank boundaries for ordinal regression’, in *Advances in Large Margin Classifiers*, 115–132. MIT Press, (2000).
- [11] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl, ‘Evaluating collaborative filtering recommender systems’, *ACM Transactions on Information Systems*, **22**(1), 5–53, (2004).
- [12] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman, ‘A survey of top-k query processing techniques in relational database systems’, *ACM Computing Surveys*, **40**(4), (2008).
- [13] Yehuda Koren, ‘Factorization meets the neighborhood: A multifaceted collaborative filtering model’, in *Proceedings of KDD 2008*, pp. 426–434. ACM Press, (2008).
- [14] Yehuda Koren, ‘Collaborative filtering with temporal dynamics’, *Communications of the ACM*, **53**(4), 89–97, (2010).
- [15] Yehuda Koren, ‘Factor in the neighbors: Scalable and accurate collaborative filtering’, *ACM Transactions on Knowledge Discovery from Data*, **4**(1), (2010).
- [16] Yehuda Koren, Robert Bell, and Chris Volinsky, ‘Matrix factorization techniques for recommender systems’, *IEEE Computer*, **42**(8), 30–37, (2009).
- [17] Don Monroe, ‘Just for you’, *Communications of the ACM*, **52**(8), 15–17, (2009).
- [18] Gregory Piatetsky-Shapiro, ‘Interview with Simon Funk’, *ACM SIGKDD Explorations Newsletter*, **9**(1), 38–40, (2007).
- [19] Tapani Raiko, Alexander Ilin, and Juha Karhunen, ‘Principal component analysis for large scale problems with lots of missing values’, in *Proceedings of ECML 2007*, volume 4701 of *LNAI*, pp. 691–698. Springer, (2007).
- [20] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, ‘Item-based collaborative filtering recommendation algorithms’, in *Proceedings of WWW 2001*, pp. 285–295. ACM Press, (2001).
- [21] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen, ‘Collaborative filtering recommender systems’, in *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *LNCS*, 291–324. Springer, (2007).
- [22] Ajit P. Singh and Geoffrey J. Gordon, ‘A unified view of matrix factorization models’, in *Proceedings of ECML PKDD 2008: Part II*, volume 5212 of *LNCS*, pp. 358–373. Springer, (2008).
- [23] Gábor Takács, István Pilászy, Botyán Németh, and Domonkos Tikk, ‘Scalable collaborative filtering approaches for large recommender systems’, *Journal of Machine Learning Research*, **10**, 623–656, (2009).
- [24] Markus Weimer, Alexandros Karatzoglou, and Alex Smola, ‘Improving maximum margin matrix factorization’, *Machine Learning*, **72**(3), 263–276, (2008).

Preference Dominance Approaches for Conversational Recommender Systems

Walid Trabelsi and Nic Wilson and Derek Bridge¹ and Francesco Ricci²

Abstract. Conversational recommender systems typically involve iteratively showing the user a small set of options for them to choose between. In order to choose an appropriate set to display at each stage, it is useful to have information regarding the user’s relative preference between options: specifically whether an option is dominated by others with respect to the user’s preference relation. This paper describes a simple semantic framework for preference dominance, and two instances of the framework are developed for a recent kind of conversational recommender system. The first is based on a basic quantitative preferences formalism, where products are compared using sums of weights of features, and the input language involves comparisons between options. The second is a qualitative preference formalism, where models are a kind of generalised lexicographic order, and the inputs are comparative preference statements (in a language generalising CP-nets). A key feature of both methods is that deductions of preference dominance can be made efficiently, since this procedure needs to be applied for many pairs of products. The two approaches have been implemented, with encouraging experimental results.

1 Introduction

In an era of overwhelming choice, *recommender systems* are a new source of assistance, helping their users to decide which goods, services or information to purchase or consume [1]. These systems infer user preferences from data gathered either explicitly, e.g. in the form of product ratings, or implicitly by observing user behaviour. The prediction algorithms used within recommender systems include collaborative filtering, content-based filtering and case-based reasoning [1, 6].

No matter how good recommender systems become, they are unlikely ever to be sufficiently prescient that their first set of recommendations always satisfies the user. Conversational recommenders allow for this, and recognise that their users may be willing and able to reveal more of their constraints and preferences, over a short dialogue. This is also an opportunity for the recommender system to guide the user by asking questions, giving advice, displaying candidate products, and giving explanations [11, 6, 13, 12]. In 2007, Bridge & Ricci introduced a new kind of *conversational recommendation*, which they call *Information Recommendation* [7]. We describe Information Recommendation in Section 5. We use it as the setting in which we explore different models of user preferences.

Conversational recommender systems typically involve iteratively showing the user a small set of options (e.g., products) for them to choose between. In order to choose an appropriate set to display at each stage, from a much larger collection of options, it is useful to have information regarding which options are likely to be preferred to others by the user, based on previous responses they have given in the dialogue; if one assumes that the user has some kind of preference relation over products, this amounts to determining if certain products are dominated according to this preference relation.

This paper describes, in Section 2, a simple semantic framework for this kind of reasoning, where a set of models of the user is assumed, with an associated preference ordering, along with a satisfaction relation between models and preference statements in an appropriate language. Based on such a framework, given a set of input statements, we infer that one choice is preferred to another if this preference holds for all models satisfying the input statements. Two instances of the framework are developed. The first, described in Section 3, is based on a simple quantitative preferences formalism, involving a sum of weights, with an associated language of linear inequalities. The second (see Section 4) is a qualitative preference formalism, where models are a kind of generalised lexicographic order, and the inputs are comparative preference statements (in a language generalising CP-nets [3]). A crucial feature of both methods is that deductions can be made efficiently, since this procedure of testing dominance needs to be applied for many pairs of products.

Both the quantitative and the qualitative approaches are developed in Section 5 for Information Recommendation [7]. Implementations of the methods have been experimentally tested, as described in Section 6.

2 A Framework for Dominance of Preferences

We assume a set Ω of outcomes, representing possible choices, for example, products that a user is interested in buying. We would like to generate some kind of (partially ordered) preference relation on Ω , based on previous information we have received regarding the user’s preferences. This section describes a simple semantic framework for generating such a preference relation. We will define in Section 3 and Section 4 two instances of the framework.

In order to make non-trivial inferences regarding the user’s relative preferences over outcomes, we will have to make some assumptions. We assume information of the following form:

- A set of models $\mathcal{M}$: each of these is intended to represent a possible user (or way the user could be). Associated with each $M \in \mathcal{M}$ is a total pre-order $\succsim_M$ on outcomes, i.e., a reflexive, transitive and complete relation (so for all outcomes α and β , we have either $\alpha \succsim_M \beta$ or $\beta \succsim_M \alpha$ (or both)).

¹ Department of Computer Science, University College Cork, Ireland, email: {w.trabelsi,n.wilson}@4c.ucc.ie; d.bridge@cs.ucc.ie

² Faculty of Computer Science, Free University of Bozen-Bolzano, email: fricci@unibz.it

- A formal input preference language $\mathcal{L}$, used to represent statements induced from the user's responses.
- A relation $\models$ between $\mathcal{M}$ and $\mathcal{L}$. For $M \in \mathcal{M}$ and $\varphi \in \mathcal{L}$, we interpret $M \models \varphi$ to mean that φ holds for the preferences of M .

For any particular situation the assumption is that we will be given a particular set $\Phi \subseteq \mathcal{L}$ of input statements. We can then consider the preference orderings on outcomes which hold for every model satisfying input statements Φ . Formally, we define relation $\succ_\Phi$ on outcomes as follows: $\alpha \succ_\Phi \beta$ if and only if $\alpha \succ_M \beta$ for all M satisfying (every member of) Φ . It follows that $\succ_\Phi$ is a pre-order (a reflexive and transitive relation) on outcomes. Now, $\alpha \succ_\Phi \beta$ means that every user who agrees with Φ considers that outcome α is at least as desirable as outcome β (assuming this particular model of users). It is possible that we also have $\beta \succ_\Phi \alpha$. Define relation $\succ_\Phi$ to be the strict part of $\succ_\Phi$, so that $\alpha \succ_\Phi \beta$ if and only if $\alpha \succ_\Phi \beta$ and $\beta \not\succ_\Phi \alpha$. Relation $\succ_\Phi$ is irreflexive and transitive. We say that *Given Φ , α strictly dominates β* , if $\alpha \succ_\Phi \beta$, i.e., if all users M agreeing with Φ regard α as at least as preferable as β (i.e., $\alpha \succ_M \beta$), and at least one user M regards α as strictly preferable to β (i.e., $\beta \not\succ_M \alpha$).

Outcomes generated from features. In this paper we consider a set of outcomes with a particular structure. We assume a collection of features $V = \{F_1, \dots, F_n\}$. Define a configuration α to be a mapping from $\{1, \dots, n\}$ to $\{1, 0\}$. Configuration α can also be thought of a set of features, i.e., all features F_i such that $\alpha(i) = 1$. The features are intended to relate to a set of products that the user is interested in choosing between; for example, in choosing a hotel room, one feature might be whether the hotel has a swimming pool. We assume here that having a feature is always at least as good as failing to have it. The set of outcomes is then a subset of the set of configurations, corresponding to choices (e.g., products) that are available to the user. For convenience we will also write a configuration such as $(1, 0, 1)$ as $f_1 \bar{f}_2 f_3$.

3 Sum of Weights Model of the User

In this section we consider our first kind of model of users, where it is assumed that a user assigns a weight to each feature, and outcomes are compared on the sum of weights of the associated set of features. This is a very commonly used model for preference representation, specifically, in Multi-Attribute Utility Theory (MAUT) [8].

The set of models is the set of all vectors of weights $w = (w_1, \dots, w_n)$, where w_i is a non-negative real number. w_i can be considered as a weighting assigned to feature F_i . Given a weights vector w , the overall value $w(\alpha)$ of a configuration α is the sum of weights of the features included, i.e., $w(\alpha) = \sum_{i:\alpha(i)=1} w_i$. This is used to define the ordering on configurations. We define the preference relation $\succ_w$ for model w by $\alpha \succ_w \beta$ if and only if $w(\alpha) > w(\beta)$, i.e., iff $\sum_i w_i(\alpha(i) - \beta(i)) > 0$. Thus $\succ_w$ is a total pre-order on configurations.

Preference language: this consists of statements of the form $\alpha \geq \beta$, where α and β are outcomes. Weight vector w is defined to satisfy $\alpha \geq \beta$ if $\alpha \succ_w \beta$. Let Φ be a set of input preference statements. The definitions in Section 2 lead to the following definition of $\succ_\Phi$, the induced preference relation given preference statements Φ :

For outcomes α and β , $\alpha \succ_\Phi \beta$ if and only if $\alpha \succ_w \beta$ for all weight vectors w satisfying Φ . Dominance relation $\succ_\Phi$ is then the strict part of $\succ_\Phi$. When comparing with the second semantics, we also use the notation $\succ_\Phi^{sw}$ for $\succ_\Phi$.

Example 1. Let Φ be the pair of preference statements: $f_1 \bar{f}_2 f_3 \geq \bar{f}_1 f_2 f_3$, and $f_1 f_2 \bar{f}_3 \geq \bar{f}_1 f_2 f_3$. Let α be the outcome $f_1 \bar{f}_2 \bar{f}_3$, and let β be the outcome $\bar{f}_1 f_2 f_3$. Weights vector w satisfies $f_1 \bar{f}_2 f_3 \geq \bar{f}_1 f_2 f_3$ if and only if $w(f_1 \bar{f}_2 f_3) \geq w(\bar{f}_1 f_2 f_3)$, i.e., $w_1 + w_3 \geq w_2 + w_3$, which holds if and only if $w_1 \geq w_2$. By similar reasoning, w satisfies Φ if and only if $w_1 \geq w_2$ and $w_1 \geq w_3$. Also, w satisfies $\alpha \geq \beta$ if and only if $w_1 \geq w_2 + w_3$. Thus Φ does not entail $\alpha \geq \beta$, so we do not have $\alpha \succ_\Phi^{sw} \beta$, since, for example, weights vector w with $w_1 = 4$, $w_2 = 2$ and $w_3 = 3$ satisfies Φ but does not satisfy $\alpha \geq \beta$.

Example 2. Suppose now that there are four features, and let Ψ be the pair of preference statements $f_1 \bar{f}_2 f_3 f_4 \geq \bar{f}_1 f_2 f_3 f_4$ and $f_1 f_2 \bar{f}_3 \bar{f}_4 \geq \bar{f}_1 f_2 f_3 \bar{f}_4$. With the sum of weights semantics this implies $f_1 \bar{f}_2 f_3 f_4 \geq \bar{f}_1 f_2 f_3 f_4$, since the first statement implies $w_1 \geq w_2$, and the second statement implies $w_2 \geq w_3$, implying $w_1 \geq w_3$, which implies that the third statement is satisfied. We therefore have $f_1 \bar{f}_2 f_3 f_4 \succ_\Psi^{sw} \bar{f}_1 f_2 f_3 f_4$.

Computational aspects: We wish to determine if $\alpha \succ_\Phi \beta$, for given configurations α and β . Let Pos be the set of constraints $w_i \geq 0$, for $i = 1, \dots, n$, representing the non-negativity of the weights (and corresponding to the assumption that including a feature is always at least as good as not including it). The definition implies that $\alpha \succ_\Phi \beta$ if and only if the linear constraints $\Phi \cup Pos$ (over real-valued variables w_i) entail the constraint $\sum_i (\alpha(i) - \beta(i)) w_i \geq 0$.

For implementation of this using a Linear Programming solver, it can be convenient to express it as a linear optimisation problem. Define A_{min} to be the minimum value of $\sum_i (\alpha(i) - \beta(i)) w_i$ subject to constraints $\Phi \cup Pos$. It can be easily shown that $\alpha \succ_\Phi \beta$ if and only if $\Phi \cup Pos$ entails $\sum_i (\alpha(i) - \beta(i)) w_i \geq 0$ if and only if $A_{min} \geq 0$.

4 Comparative Preferences Model of the User

In this scenario, models are a kind of generalised lexicographic order, called *cp-trees* [16], which are similar to search trees used for solving constraint satisfaction problems. Figure 1 gives an example of a cp-tree. There is a most important variable, F_2 , and an ordering of its values. In order to be consistent with the requirement that including a feature is never worse than not including it, this local ordering is $f_2 \geq \bar{f}_2$, where f_2 means F_2 is included ($F_2 = 1$) and $\bar{f}_2$ means that F_2 is not included ($F_2 = 0$).

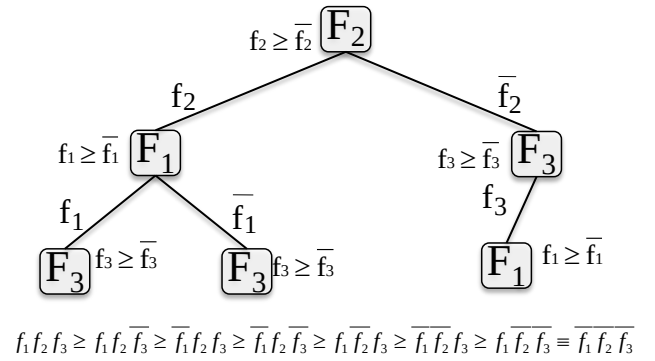


Figure 1. A cp-tree σ , along with its associated ordering on outcomes $\succ_\sigma$.

Two configurations α and β are compared first on this most important variable. If they do not agree on this variable then the comparison is settled: If α contains feature F_2 and β does not, then α is better than β . This happens, for example, if α is the outcome $f_1 f_2 \bar{f}_3$ and β

is $f_1\bar{f}_2f_3$. Otherwise, α and β agree on the most important variable. The user may then have a next most important variable; which this is can depend on the value assigned to the most important variable (thus this model allows *conditional preferences*). If there is no such next important variable, then α and β are considered equally preferable according to this cp-tree. Thus the cp-tree σ generates a total pre-order $\succsim_\sigma$ on outcomes.

Note that a node in the cp-tree in Figure 1 is associated with a single variable. In fact, we can allow a more general representation, where at most γ variables (where e.g., $\gamma = 1, 2$ or 3) are associated with a node, along with a total pre-order over the assignments to that set of at most γ variables. For example, if a node is associated with the pair of variables $Y = \{F_2, F_3\}$ then the local ordering is over assignments to Y , and might be e.g., $f_2f_3 \geq \bar{f}_2\bar{f}_3 \equiv \bar{f}_2f_3 \geq \bar{f}_2\bar{f}_3$. Let $\mathcal{M}(\gamma)$ be the set of cp-trees over Ω , where the set of variables associated to a node involves at most γ variables. A 1-cp-tree over Ω is defined to be an element of $\mathcal{M}(1)$, i.e., a cp-tree with a single variable being associated with each node. The local ordering associated with a node associated with feature F_i , must then be $f_i \geq \bar{f}_i$, since including a feature is always at least as good as not including it. Full definitions of cp-trees are given in [16].

The Language: The preference language will include preference statements that compactly express comparative (and sometimes conditional) preferences. There is a substantial and fast growing literature on this topic in the AI and Philosophy of Science communities, see e.g., [14, 9, 3, 15, 5, 4, 16].

Each statement ϕ in the language has an interpretation ϕ^* , which is a relation between outcomes, giving the direct implications of ϕ regarding preferences between outcomes. We say that total pre-order $\succsim$ satisfies ϕ iff $\succsim$ extends ϕ^* , i.e., if $(\alpha, \beta) \in \phi^*$ implies $\alpha \geq \beta$.

The language will include only comparative preference statements ϕ of the form $p \geq q \parallel T$, where P, Q and T are sets of features, and p is an assignment to P (i.e., a function from P to $\{0, 1\}$), and q is an assignment to Q . Informally, the statement $p \geq q \parallel T$ represents the following: p is preferred to q if T is held constant. Formally, the semantics of this statement is given by the relation ϕ^* which is defined to be the set of pairs (α, β) of outcomes such that α extends p (i.e., α restricted to P equals p), and β extends q , and α and β agree on T : $\alpha(T) = \beta(T)$. A very important kind of input statement is one expressing a preference for one outcome, α , over another, β . This can be written as $\alpha \geq \beta \parallel \emptyset$; as above we also write such a preference statement as $\alpha \geq \beta$.

Since including a feature is at least as good as not including it, we always include, in the set Φ of input statements, the preference statement $f_i \geq \bar{f}_i \parallel V \setminus \{F_i\}$ for each feature F_i . Hence our language is strongly related to *Conditional Importance Networks* [4].

Comparative preference dominance relation. We define the set of models of users to be the $\mathcal{M}(\gamma)$ for some $\gamma = 1, 2$, or 3 . Each cp-tree σ generates a total pre-order $\succsim_\sigma$ on outcomes. Let σ be a cp-tree and let ϕ be an input comparative preference statement. σ satisfies ϕ if and only if $\succsim_\sigma$ satisfies ϕ , which is if and only if $\succsim_\sigma$ extends ϕ^* . In the same way as in Section 2 and Section 3, we define, for a given set of input comparative preference statements Φ , $\succsim_\Phi$ (or more precisely, $\succsim_\Phi^{cp}$), in the following way: $\alpha \succsim_\Phi \beta$ holds if and only if $\alpha \succsim_\sigma \beta$ holds for all cp-trees σ in $\mathcal{M}(\gamma)$ satisfying Φ .

Example 1 continued. With the cp-tree semantics, when $\gamma = 1$, the pair of statements Φ implies the preference statement $f_1\bar{f}_2\bar{f}_3 \geq \bar{f}_1f_2f_3$, so we have $f_1\bar{f}_2\bar{f}_3 \succsim_\Phi^{cp} \bar{f}_1f_2f_3$. The reason is that, for any 1-cp-tree σ satisfying $f_1\bar{f}_2\bar{f}_3 \geq \bar{f}_1f_2f_3$, the most important feature

must be either F_1 or F_3 . (If F_2 were the most important feature, then we would not have $f_1\bar{f}_2\bar{f}_3 \succsim_\sigma \bar{f}_1f_2f_3$, because the local ordering is $f_2 \geq \bar{f}_2$, since the presence of a feature is never worse than its absence.) Similarly, if 1-cp-tree σ satisfies $f_1\bar{f}_2\bar{f}_3 \geq \bar{f}_1f_2f_3$, then the most important feature must be either F_1 or F_2 . Hence for any 1-cp-tree satisfying Φ , F_1 is the most important variable. The top node then decides pair of outcomes $f_1\bar{f}_2\bar{f}_3$ and $\bar{f}_1f_2f_3$. Since the local ordering of this node must be $f_1 \geq \bar{f}_1$, we have $f_1\bar{f}_2\bar{f}_3 \succsim_\sigma \bar{f}_1f_2f_3$. Hence we have $f_1\bar{f}_2\bar{f}_3 \succsim_\Phi^{cp} \bar{f}_1f_2f_3$. The qualitative and lexicographic nature of the cp-trees semantics ensures this inference, in contrast with the numerical sum of weights method.

Example 2 continued. In contrast with the sum of weights semantics, Ψ does not imply $f_1f_2\bar{f}_3f_4 \geq \bar{f}_1f_2f_3f_4$. To show this we can construct a 1-cp-tree σ with F_4 as the most important (top) variable, and where, given f_4 , F_3 is more important than F_1 which is more important than F_2 , and given $\bar{f}_4$, F_2 is more important than F_3 which is more important than F_1 . σ then satisfies Ψ , but not $f_1f_2\bar{f}_3f_4 \geq \bar{f}_1f_2f_3f_4$.

A key issue here is that cp-trees can represent *conditional preferences*: the preferences can be different given f_4 from those given $\bar{f}_4$. In contrast, the sum of weights semantics assumes preferential independence, so preferences are not conditional at all, which is why the inference holds for the sum of weights semantics.

The pair of examples shows that the two preference dominance techniques are incomparable: $\succsim_\Phi^{cp}$ can sometimes include preferences not included in $\succsim_\Phi^{sw}$, and vice versa.

Computation of Preference: Given set of preference statements Φ , and outcomes α and β we can determine in polynomial time whether or not $\alpha \succsim_\Phi \beta$ holds, using the algorithm given in [16], as shown by Theorem 1 in [16].

5 Development of Approaches for a Conversational Recommender System

We compare the Sum of Weights Model and the Comparative Preferences Model in the conversational product recommender system application first described in [7]. In this setting, a user repeatedly edits and resubmits a query until she finds a product that she wants. The recommender system: observes the user's actions; infers user preferences; uses the preference statements to deduce which queries a user is likely to try next; and advises the user to avoid those that are unsatisfiable. In [7], users are represented by the Sum Of Weights Model.

Below we describe this recommender system in more detail; we relate it to the framework for dominance of preferences presented above; and we show how users can alternatively be represented by the Comparative Preferences Model.

In this setting, outcomes Ω , over which preferences are expressed, are queries over a set of features $F_1, \dots, F_n$ (rather than e.g., products). If a user issues query q and if $f_i \in q$, this means that the user is interested in products that have the i th feature. In accordance with most Web-based product search systems, $f_i \notin q$ means only that the user has not (yet) declared any interest in feature F_i ; it does not mean that the user wants products that lack the i th feature.

A query is *satisfiable* if and only if there exists a product which has all the features in the query; otherwise, it is *unsatisfiable*. Users cannot be expected to know in advance which queries are satisfiable although they may have incomplete knowledge of this.

In the kind of system envisaged in [7], the user submits an initial query, typically one that is quite under-specified: ‘to test the water’. In our experiments we use an empty initial query.

The recommender system does not know the user’s preferences and does not ask about them. It may only infer them from the sequence of queries that the user submits. As the dialogue proceeds, the recommender system will infer preference statements expressed in preference language $\mathcal{L}$. We will denote the current set of statements by Φ . Initially Φ contains a set of ‘background’ assumptions. In particular, we wish to express that including a feature in a query is at least as good as not including it. For the Sum of Weights Model, initially $\Phi = Pos$, i.e. the set of constraints $w_i \geq 0$, for $i = 1, \dots, n$. For the Comparative Preferences Model, initially Φ contains the preference statement $f_i \geq \bar{f}_i \parallel V \setminus \{F_i\}$ for each feature F_i .

The interaction between the user and the recommender system proceeds as follows:

1. The recommender system analyzes q , with particular regard to differences between current query q and the previous query the user submitted. (In the case where q is the very first query, the previous query is the empty set.) The system induces some additional preference statements to add to inputs Φ .
2. The recommender system generates a set of candidate next possible queries and prunes this set to those that are satisfiable and undominated (see below). It advises the user to confine her next query to this set.
3. The user chooses and submits her next query. This becomes the new current query q . In the experiments reported in the next section, we arrange that the user always chooses one of the queries that the system advises (although this might not be so in practice.)

Steps 1–3 are repeated until the user is satisfied with q or the set of undominated, satisfiable candidates is empty, in which case as far as the recommender system is concerned q cannot be bettered. At this point, the user can request to see the products that satisfy q .

The goal of the recommender system is to give the advice that has the greatest value. We consider this to be that which minimizes the total quantity of advice given and the dialogue length, while guiding the user to the best product.

During step 2 above, the recommender system computes the following three sets of queries:

- *Candidates*: Candidate queries are ones which are close, in a particular sense, to the current query. Each is a low-cost edit to the current query. For example, for each $f_i \notin q$, the set of candidates will include the query that results from adding just feature f_i to q .
- *Satisfiables*: The recommender system never includes unsatisfiable queries in its advice: they make interaction length longer without leading the user to the best product. Hence, the system eliminates from *Candidates* those queries which are unsatisfiable; the remaining queries are called the *Satisfiables*.
- *Undominated*: The system could advise the user to confine her query to *Satisfiables*. However, this set can be large. Hence, the system eliminates from *Satisfiables* each query which is dominated by (i.e., worse than) some other member of *Satisfiables*; the remaining set of queries is called *Undominated*. The dominance relation is based on what is induced in step 1 above. The rationale is to exclude from the system’s advice queries that, on the basis of what the system has induced about the user’s preferences, it thinks the user would regard as inferior.

In the following, we will explain how to obtain the three sets of queries.

Generating the candidates: Real user behaviour in query editing tends to proceed with modifications of limited ‘reach’. Hence, following [7], we define *Candidates* as the set of queries which we obtain by applying three editing operations, *Add*, *Switch* and *Trade*, to the current query. These operations are defined as follows. Given current query q and $f_i \notin q$, then $Add(q, f_i)$ adds just feature f_i to q , giving the new query $q \cup \{f_i\}$, which we sometimes write as q^i . $Switch(q, f_i, f_j)$ where $f_i \in q, f_j \notin q, i \neq j$ discards feature f_i in favour of feature f_j , giving the new query $q \setminus \{f_i\} \cup \{f_j\}$. Finally, $Trade(q, f_i, f_j, f_k)$ where $f_i \in q, f_j \notin q, f_k \notin q, i \neq j, i \neq k, j \neq k$ discards feature f_i and introduces features f_j and f_k .

Checking satisfiability: As defined earlier, a query is satisfiable if and only if there exists a product which has all the features present in the query. (It may also have other features, not present in the query.) To produce *Satisfiables* requires eliminating unsatisfiable queries from *Candidates*. If products are stored explicitly in a database, satisfiability of a candidate query can be checked by a scan of the database. For configurable products, where the set of products is represented as a set of solutions to a Constraint Satisfaction Problem, satisfiability of a candidate query can be checked by determining if the CSP has solutions containing all the features in the query (which can be checked by checking satisfiability of an augmented CSP).

Checking for dominance: The final pruning of the satisfiable candidate queries can be performed using one of the two strict dominance approaches described earlier. In either case, $q \in Satisfiables$ is pruned if it is strictly dominated, i.e., dominated according to relation $\succ_{\Phi}$, by $q' \in Satisfiables$. This dominance relation is based on the set Φ of formal preference statements which are induced in step 1 of the dialogue.

It remains for us to explain what the system induces in step 1 above, when it observes the user’s queries. We explain this below for each of the two preference models. Due to space limits, we confine our treatment only to what can be inferred when the system observes the user adding a feature f_i to a query q , $Add(q, f_i)$. However, our system induces preference statements for *Switch* and *Trade* as well as for *Add*.

Generating induced preference statements for sum of weights model: If the user has added feature f_i to query q , then statements $q^i \geq q^j$ are induced for all $f_j \notin q, i \neq j$ unless $Add(q, f_j) = q^j$ is unsatisfiable. This assumes that the value of the new query must be at least as much as the value of other satisfiable queries that could have been generated by adding other features. This implies that the weight vector satisfies the linear inequality $w_i \geq w_j$. However, we do not infer $q^i \geq q^j$ in all cases. In particular, we do not infer it if $Add(q, f_j)$ is unsatisfiable. Users may have (incomplete) knowledge of which queries are unsatisfiable: if she knows a query is unsatisfiable, then she will not submit it. We ‘play it safe’: when q^j is unsatisfiable, in case the user knows this, we do not assume that the query that she does submit has higher weight than this unsatisfiable query.

Generating induced preference statements for comparative preferences model: Again consider the situation where the user has chosen to add feature f_i instead of feature f_j . There is more than one way one might induce a comparative preference statement from this decision by the user. We consider two, each being a kind of coun-

terpart for the constraint $w_i \geq w_j$ induced for the sum of weights approach.

Basic: Let q be the current query, let q^i be the current query q with the feature f_i added, and let q^j be q with the feature f_j added. A basic, somewhat conservative, approach is to just model the preference of feature i over feature j by the preference statement: $q^i \geq q^j || \emptyset$, i.e., $q^i \geq q^j$, which just expresses a preference for q^i over q^j .

Importance: Alternatively, and less conservatively, we can induce $f_i \geq f_j || V \setminus \{F_i, F_j\}$, which says that the presence or not of the feature F_i is more important than the choice of F_j . Thus, whatever the state of the feature F_j in the query the user will prefer F_i to be present in the query so that this feature is included in the best product.

Note too that, in either case we ensure that the recommender system ‘plays it safe’ when inducing preference statements, in the same way that we explained for the Sum of Weights Model.

6 Experiments

In this section, we report experiments with simulated users that demonstrate the feasibility of using both the Sum of Weights Model and the Comparative Preferences Model within the conversational recommender system that we described above. We use two separate product databases, that we scraped from the Web, each describing hotels by their amenities expressed as Boolean features such as *airport shuttle*, *pets permitted*, *restaurant on-site*, etc. The Marriott-NY database records 9 features about 81 hotels; many offer the same amenities, and so there are 36 distinct products in the database. The Trentino-10 database records 10 features about 4056 hotels, of which 133 are distinct.

The simulated users behave in the following somewhat idealized way: within a dialogue, they do not try queries that they have tried earlier in the dialogue; they are aware of their own preferences and never choose a next query that would be inferior to the current one; and they take heed of all advice given, i.e. if the recommender system tells them to confine their next query to a certain set, then they do so; indeed they choose the best possible query from this set. (In the terminology of [7], these are *optimizing users*.) The user’s true preferences are represented either in the Sum of Weights Model by randomly generating weight vectors over product features or in the Comparative Preferences Model by randomly generating cp-trees over product features. While this vector or cp-tree is known to the simulated user and used for query selection, it is not known to the recommender system, which knows only what it induces and adds to Φ when observing user query behaviour.

In the experiments, we pair the users with each of several recommender systems. One recommender system uses the Sum of Weights Model. Six use the Comparative Preferences Model, differing first on which of the two alternative preference statements they infer (Basic or Importance), and on their value for γ (1, 2 or 3). For each pairing of a user with a recommender system, we ran 500 simulated dialogues.

The experiments allow us to compare the pruning rate achieved by using the Sum of Weights Model with those achieved by the six recommender systems that use the Comparative Preferences Model. The pruning rate is defined as follows:

$$\text{pruning rate} = \frac{|\text{Satisfiables} \setminus \text{Undominated}|}{|\text{Satisfiables}|} \times 100$$

It shows the extent to which an approach eliminates what it takes to be inferior satisfiable candidate queries from its advice. Other things being equal, the shorter the advice the better, as this reduces the choice the user has to make. The results in the case where the users’ true preferences are represented in the Sum of Weights Model are shown below.

	$\gamma=1$	$\gamma=2$	$\gamma=3$
Marriott-NY			
Comp. Prefs. Basic	87.50	14.48	12.65
Comp. Prefs. Importance	87.50	87.49	87.42
Sum of Weights	87.38		
Trentino-10			
Comp. Prefs. Basic	87.49	16.51	13.98
Comp. Prefs. Importance	87.42	87.57	86.72
Sum of Weights	85.72		

The table shows that, in nearly all settings, the Comparative Preferences approach is pruning non-optimal queries a little more than the Sum of Weights approach. For example, in the *Marriott-NY* part of the table, the Comparative Preferences Model using *Basic* preference statements and with $\gamma = 1$ eliminates 87.5% of satisfiable candidates in dialogues about the Marriott-NY database, where the Sum of Weights approach prunes 87.38%. The tables also show that, on the whole, with the Comparative Preference Model, the amount of pruning increases as the preference statements induced become less conservative (from Basic to Importance). For example, in the *Trentino-10* part of the table with $\gamma = 2$, pruning goes from 16.51% Basic to 87.57% Importance. (The very slight exception to this for the Trentino- $\gamma = 1$ case is probably due to random variation in tie breaking.)

Furthermore, we see that the parameter γ , (the maximum number of variables that are associated with a node in a cp-tree), affects the degree of pruning. Specifically, as γ increases, the number of queries pruned tends to decrease. For example, in the *Marriott-NY* part of the table with preference statements Importance, pruning goes from 87.50% ($\gamma = 1$) to 87.49% ($\gamma = 2$) to 87.42% ($\gamma = 3$). This is a reflection of the monotonicity with respect to γ observed above (however, pruning is to do with strict dominance, which is not necessarily monotonic with respect to γ , but very often will be, because of the monotonicity of dominance.) The effect is especially marked in the Basic model where the pruning rate falls from nearly 90% to around 16.5% or less. When γ is increased from 1 to 2, many queries of the *Trade* form become undominated in the Basic model, because of the more expressive preference relations which can be represented by cp-trees with $\gamma = 2$ (allowing more than one feature to be assigned at a node). With the stronger Importance preference form, these *Trade* queries are still dominated.

What is also of concern from a practical point of view is the average length of the advice that the system gives, i.e., the number of options the user has to choose from; this is inversely related to the pruning rate. Except in the cases where pruning is very low (Basic with $\gamma = 2$ or 3), advice from the Sum of Weights recommenders is very slightly longer than it is in the case of the Comparative Preferences recommenders, being around 10 for both datasets. Dialogue lengths are very similar in the case of all recommenders: around 6 steps on average for Marriott-NY, and around 6.8 steps for Trentino-10.

Of course, it is not enough to know that one approach prunes more than another, or gives shorter advice. If it were doing so to the detriment of other factors, in particular the ability of the user to reach the best product, then the extra pruning would be of little value. We have

measured the extent to which the final queries that the user reaches in a dialogue (and hence the final product that she might choose) agree across the different recommenders. Space limits preclude the inclusion of detailed results but we find that the Comparative Preferences approaches agree with the Sum of Weights approach between 91 and 99% of the time, and the more an approach prunes, the less this agreement is. For example, for Trentino-10, Basic $\gamma = 1$ agrees with Sum of Weights 92.6% of the time; this rises to 96.2% for $\gamma = 2$; and it falls to 92% for Importance $\gamma = 1$.

Where the user's true preferences are represented in the Sum of Weights Model, we have also measured the amount by which the utility of the product that the user ultimately chooses falls short of the utility of the best product that she could have reached, normalized by the difference between the products of highest and lowest utility. Unsurprisingly, these follow a similar pattern to the percentage agreements reported in the previous paragraph. The values are very close to zero, ranging from 0 to 0.008.

When $\gamma = 1$, the time taken by the different implementations of the pruning is roughly similar—for example, around 0.2 seconds for a dialogue with the basic comparative preferences pruning for the Trentino dataset—with the sum of weights linear programming algorithm taking a little longer than the two others. The computation time increases exponentially with γ ; for example, the Basic-Trentino- $\gamma = 3$ dialogue takes on average around 7.5 seconds.

Overall, for this experimental setup it seems that it is better to use the more restrictive set of models corresponding to $\gamma = 1$, at least for the Basic preferences input form, because, it generates much greater pruning, leading to manageable sets of options for the user, and is computationally cheaper. However, in other situations the more cautious reasoning corresponding to $\gamma = 2$ or 3 might pay off in terms of the final quality of solutions.

We have shown results in the case where the user's true preferences are represented in the Sum of Weights Model. Space limits prevent us from showing what happens when their preferences are modeled using cp-trees. Suffice to say that the numbers are very similar and follow similar patterns to those reported above.

7 Discussion

There has been a lot of excellent theoretical work produced on comparative preference formalisms in recent years, for example, the award winning papers [3, 10]; however, development towards applications has been lagging somewhat. A major contribution of this paper is to show how a comparative preferences approach can be adapted for a conversational recommender system. For this kind of application it is important that the preference language allows the expression of direct comparisons between outcomes (that one outcome is preferred to another); and also that the inference technique is both efficient and rather adventurous in its inferences (or else the pruning of possibilities is too weak). The language, inference method and algorithm described in [16] fit these requirements. We have shown that it is possible to implement this inference method so that it is efficient in a practical (rather than theoretical) sense, as part of a form of conversational recommender system. We have tested our method on datasets involving real hotel data, and shown that it leads to strong pruning of possibilities, but without eliminating the best options, even when 'best' is defined based on a different semantics (based on a sum of weights of included features).

An attractive feature of the comparative preferences approach is that it would also allow general conditional preference statements to be expressed by the user, such as *If the hotel is not in the city centre,*

then I'd like there to be an on-site restaurant. Such statements can further strengthen the pruning capability, and could potentially be re-used for different searches.

In future work, we will extend these approaches for other kinds of recommender systems, including for non-boolean features, and for configurable products, where the set of possibilities is expressed implicitly as the solutions of a Constraint Satisfaction Problem. We will also consider other inference procedures. For instance, the weighted sum model can be easily extended to a sum of functions of more than one variable (a GAI representation [2]).

Acknowledgements

This material is partly supported by the Science Foundation Ireland under Grant No. 08/PI/I1912.

REFERENCES

- [1] Gediminas Adomavicius and Alexander Tuzhilin, 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE Transactions on Knowledge and Data Engineering*, **17**(6), 734–749, (2005).
- [2] F. Bacchus and A. Grove, 'Graphical models for preference and utility', in *Proc. UAI-95*, pp. 3–10, (1995).
- [3] C. Boutilier, R. I. Brafman, C. Domshlak, H. Hoos, and D. Poole, 'CP-nets: A tool for reasoning with conditional *ceteris paribus* preference statements', *Journal of Artificial Intelligence Research*, **21**, 135–191, (2004).
- [4] Sylvain Bouveret, Ulle Endriss, and Jérôme Lang, 'Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods', in *Proc. of IJCAI*, pp. 67–72, (2009).
- [5] R. Brafman, C. Domshlak, and E. Shimony, 'On graphical modeling of preference and importance', *Journal of Artificial Intelligence Research*, **25**, 389–424, (2006).
- [6] Derek Bridge, Mehmet Göker, Lorraine McGinty, and Barry Smyth, 'Case-based recommender systems', *The Knowledge Engineering review*, **20**(3), 315–320, (2006).
- [7] Derek Bridge and Francesco Ricci, 'Supporting product selection with query editing recommendations', in *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pp. 65–72, New York, NY, USA, (2007). ACM.
- [8] J. Figueira, S. Greco, and M. Ehrgott, *Multiple Criteria Decision Analysis - State of the Art Surveys*, Springer International Series in Operations Research and Management Science Volume 76, 2005.
- [9] S. O. Hansson, 'Preference logic', in *Handbook of Philosophical Logic*, eds., D. Gabbay and F. Guenther, 319–393, Kluwer, (2001).
- [10] Frédéric Koriche and Bruno Zanuttini, 'Learning conditional preference networks with queries', in *Proc. IJCAI 2009*, pp. 1930–1935, (2009).
- [11] David McSherry, 'Retrieval failure and recovery in recommender systems', *Artificial Intelligence Review*, **24**(3–4), 319–338, (2005).
- [12] Pearl Pu, Paolo Viappiani, and Boi Faltings, 'Increasing user decision accuracy using suggestions', in *Procs. of the SIGCHI conference on Human Factors in computing systems*, pp. 121–130. ACM Press, (2006).
- [13] Francesco Ricci, Dario Cavada, Nader Mirzadeh, and Adriano Venturini, 'Case-based travel recommendations', in *Destination Recommendation Systems: Behavioural Foundations and Applications*, eds., D. R. Fesenmaier et al., 67–93, CABI, (2006).
- [14] G. H. von Wright, *The logic of preference*, Edinburgh University Press, 1963.
- [15] N. Wilson, 'Extending CP-nets with stronger conditional preference statements', in *Proceedings of AAAI-04*, pp. 735–741, (2004).
- [16] N. Wilson, 'An efficient deduction mechanism for expressive comparative preference languages', in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 961–966, (2009).

Alice and Bob Will Fight: The Problem of Electing a Committee in the Presence of Candidate Interdependence¹

Joel Uckelman²

Abstract. The problem of electing a committee which satisfies voters is one for which good solutions are scarce. Extending single-winner voting rules to the multi-winner case works well only when voters have no preferential dependencies among candidates for the committee. (Our motivating example is a voter who believes that Alice and Bob are the best candidates, but also that the worst possible committee is one with *both* Alice and Bob.) In order to tackle the interdependence problem, we propose a voting rule called the Goalbase Summation Rule (GSR), which uses goalbases (sets of weighted propositional formulas) as ballots. Using goalbases as ballots lets voters express complex preferences in a compact fashion, while the computational complexity of finding winning committees remains reasonable when the number of seats is fixed. Additionally, the GSR is able to simulate and extend many existing voting rules.

1 Introduction

The problem of electing a committee which satisfies voters is one for which good solutions are scarce. There have been numerous attempts to devise methods for committee election, some which have their origins in single-winner voting methods [4, 5, 6], others of which are intended to produce outcomes which are proportional in some way [7, 18]. Single-winner voting systems frequently fail to respect voter preferences when extended to a multi-winner setting, due mainly to the fact that they deny voters the ability to express interdependence among candidates. Moreover, the way in which such systems measure the “representativeness” of committees may not be at all similar to the way in which voters measure it.

Overcoming preferential dependence is an important issue in decision making, whether the voters are people or software agents. In this latter guise, the problem arises in Multiagent Systems and Computational Social Choice—where there is an extensive literature on preference representation and associated problems (e.g., [8, 9, 10, 13, 14, 15, 16, 17, 20, 21])—and hence solutions for the problem are applicable to AI. To tackle the interdependence problem, we propose a voting method which uses goalbases as ballots, in the spirit of combinatorial vote [15].

In Section 2, we present some known methods for committee selection and find fault with them due to their lack of expressive power. In Section 3 we introduce the Goalbase Summation Rule, which uses sets of weighted propositional formulas as ballots, is able to handle dependencies between candidates in a compact fashion, and can simulate and extend many other voting rules. In Section 4 we consider the computational complexity of finding winning committees using this

method. Finally, Section 5 discusses issues arising from the use of the GSR and touches on several avenues for further investigation.

2 Existing Methods for Committee Election

Voters cast ballots for candidates; voting theory concerns itself with the properties of *voting rules*, which map collections of ballots to winners (a subset of the candidates). In what follows, we make frequent reference to four voting rules: *Plurality* is the familiar method where each voter casts a single vote. *Approval* permits a voter to cast at most one vote per candidate. *Borda* has voters ranking the candidates, with each candidate receiving a number of points equal to the number of other candidates who rank below him. *Cumulative* vote gives each voter a supply of points which he may distribute over the candidates as he wishes. In all four cases, the winner is the candidate who receives the most points. There are many other voting rules, but these four suffice for our present purposes.

Various methods for committee elections have been proposed. (For a general discussion of the difficulties of committee elections, see [8] and [16].) The naïve—and perhaps for that reason, most widely used—approach is to extend single-winner plurality to choose more winners.

Definition 1 (*m*-vote, *ℓ*-support, top-*k*). *Call a voting rule m-vote if each voter may cast at most m votes; ℓ-support if each voter may cast at most ℓ votes per candidate; and top-k if the k candidates receiving the most votes are the winners.*

Each tuple $\langle k, \ell, m \rangle$ where $k, \ell, m \in \mathbb{N}$ and $\ell \leq m$ defines a voting rule. Standard, single-winner plurality is the 1-vote 1-support top-1 rule, while approval voting is the n -vote 1-support top-1 rule. Cumulative vote is m -vote m -support top-1, while Borda is the $\frac{n(n-1)}{2}$ -vote $n-1$ -support top-1 with the additional constraint that support must be given exactly in the amounts $n-1, n-2, \dots, 1, 0$.

All m -vote ℓ -support top- k rules have the following flaw when $k > 1$: Voters for whom the candidates are not independent may have no principled way to vote. For example, suppose that we are electing a three-seat committee from the five candidates Alice, Bob, Charlie, Dave, and Elaine (a, b, c, d, e , respectively). Suppose further that a voter Valerie believes that

1. Alice and Bob are the best candidates, so any committee with one of them is better than any committee with neither, and
2. Alice and Bob will fight if they are on the committee together, so any committee with both is worse than any committee with neither.

These two constraints generate the preference order

$$acd, ace, ade, bcd, bce, bde > cde > abc, abd, abe,$$

¹ A 2-page version of this paper will appear in the proceedings of ECAI-2010.

² Institute for Logic, Language and Computation, University of Amsterdam.
Email: j.d.uckelman@uva.nl

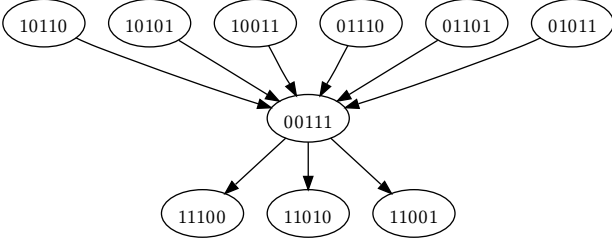


Figure 1. A realistic order on ballots.

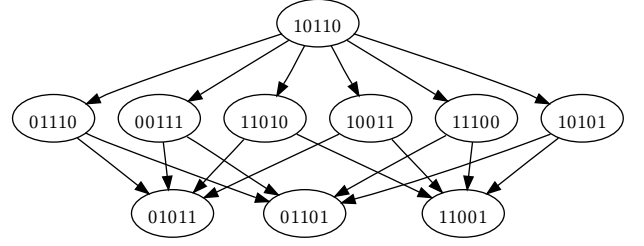


Figure 2. Order on ballots induced by Hamming distance from 10110.

as seen in Figure 1, where the committees are represented as binary vectors and nodes at the same level are equally preferred.

How should Valerie vote if faced with an m -vote ℓ -support top-3 rule? If Valerie votes for both Alice and Bob in hopes that only one will win, she risks electing one of her least-favored committees. If Valerie votes for neither Alice nor Bob in hopes that other voters will prefer one over the other, she risks electing her second choice committee cde . If Valerie votes for Alice but not Bob, or Bob but not Alice, she has no principled way to choose between these options, as she prefers Alice-committees and Bob-committees equally. In fact, Valerie's problem arises for nearly all such rules:

Proposition 2. *Let $a, b, x, y \in C$ (the set of candidates) be distinct, $\vec{c} \subseteq C \setminus \{a, b, x, y\}$, and $|C| \geq 4$. Then no m -vote ℓ -support top- k rule has a ballot compatible with (i.e., induces an order which extends) the partial order $ab\vec{c} < xy\vec{c} < ax\vec{c}, by\vec{c}$, for $k \geq 2$.*

Proof. Fix a ballot $c_1, \dots, c_n$, where c_i denotes the number of votes given that candidate. Suppose that the ballot induces the order $ab\vec{c} < xy\vec{c} < ax\vec{c}, by\vec{c}$. Wlog, let $\vec{c} = \emptyset$. So $a + b < x + y < a + x, b + y$. But then $a + b < a + x \implies b < x$, and $x + y < b + y \implies x < b$, contradiction. $\square$

As summing votes seems to drive this impossibility result, we could try a distance-based vote aggregator such as minisum [4]. Consider ballots and committees as binary vectors; then the winning committee minimizes the *sum* of the Hamming distances to the ballots cast:

$$c \text{ is a winner} \iff \forall c' \in C^k: \sum_{b \in B} H(c, b) \leq \sum_{b \in B} H(c', b),$$

where C^k is the set of k -seat committees and B is the multiset of ballots cast, expressed as binary vectors. (The Hamming distance H between a ballot and a committee is the least number of bits which must be flipped to transform one vector into the other. For example, $H(01010, 01101) = 3$.) Intuitively, any winning committee is one which is as similar in membership as it can be to as many ballots as it can be. This intuition is *wrong*: Any m -vote 1-support k -minisum winner will consist of the top k candidates [6, Appendix, Prop. 4].

Consider instead minimax [6], where

$$c \text{ is a winner} \iff \forall c' \in C^k: \max_{b \in B} H(c, b) \leq \max_{b \in B} H(c', b),$$

with the variables as before. The m -vote 1-support k -minimax rule is not the same as any summation rule for nontrivial numbers of voters and candidates, but has the quirk that the number of each type of ballot cast is irrelevant and the farthest outlier has the most influence on the result. If one incorrigible voter casts the ballot 00101 while all others choose 11101, then it makes no difference if there are two, ten, or a

million ballots cast in total—the winning three-member committees are 10101 and 01101. Usually this is not a desirable feature (though it arguably is appropriate in some circumstances, e.g., multilateral treaty negotiation [4]).

While the *intent* of minimax is to minimize the dissatisfaction of the farthest outlier, the *rule* is to minimize the dissimilarity between the farthest outlier's ballot and the winning committee. But why should we suppose that the farthest outlier (or any voter, for that matter) actually cares about his ballot's similarity to the winning committee? It is quite reasonable to think that some voters' committee preferences will not track the Hamming distance at all. For example, if c is voter v 's preferred committee, then any substitution of n members in c is strictly better according to Hamming distance than every substitution of m members, for $n < m$, and all committees c' which are Hamming-equidistant from c are equally preferred by v —both of which are dubious when applied to voters electing real committees.

Returning to our previous example, observe that Valerie's preference ordering is sensitive to small changes in committee composition. Each of the best committees is only one substitution away from some worst committee. Put another way, the Hamming distance between some pairs of best and worst committees is 2, which is always the minimum Hamming distance between two committees of the same size. From Valerie's point of view, $acd \sim bde > abc$; but $H(10110, 01011) = 4$ while $H(10110, 11100) = 2$, so the ordering induced by the Hamming distance from acd is $acd > abc > bde$, thus putting an optimal committee last and one of her least favored committees in second place (cf. Figures 1 and 2). If we use a minimax procedure and have many voters with preferences like Valerie's, we risk an outcome that is similar in composition to voters' first choices, yet is widely disliked. Furthermore, m -vote 1-support k -minimax rules suffer from the same limited expressivity as top- k rules were shown to in Proposition 2.

The problem we have identified is that the question of committee membership for one candidate is not necessarily independent of the question of committee membership for some other candidate.³ To borrow from the language of utility functions, some voters have nonmodular preferences. It could be argued that the problem is caused not by the voting methods we examined, but rather because of the way they are applied: The candidates are individuals rather than committees. If committees were raised to the status of first-class citizens—that is, if voters were to vote for whole committees rather than for individuals—perhaps we would not have this problem. However, this approach is unhelpful implemented one way, and scales poorly implemented another. For example, the 2003 Game Theory Society election filled

³ There is a similar problem with the election of representative assemblies—not only might a voter have complex preferences over the composition of the assembly, but preferences over candidates for his district might depend on candidates for other districts where he isn't even able to cast a vote [2].

12 seats from a slate of 24 candidates, giving 2704156 possible committees for voters to consider [6]. This is still manageable if the voter is queried for his most preferred committee only—but then we are left with no information about candidate interdependence, and so we are no better off than before. It is also likely that no two voters will share the same committee as their first choice, so the result will be a many-way tie. If we ask voters for rankings, we begin to sink into the combinatorial morass: Voters would balk at ranking their top 0.001% of the possible committees let alone all 2.7 million of them, and even if the voters were able to rank all of the possible committees the vote tabulator would be overwhelmed by the data.

The unsuitability of single-candidate voting systems for electing committees may also be seen by observing how many voter profiles single-candidate voting methods can accommodate. The most expressive voting method considered above is approval voting, where every subset of candidates is a valid ballot. (All other methods mentioned restrict the set of valid ballots to a proper subset of the powerset of candidates.) In comparison, there are

$$\sum_{i=1}^{\binom{n}{k}} \sum_{j=1}^i (-1)^{i-j} \binom{i}{j} j^{\binom{n}{k}}$$

distinct voter profiles over k -seat committees chosen from n candidates, which, for any useful value of n , dwarfs the 2^n distinct approval ballots.⁴ Taking the Game Theory Society election as our example, the largest term in the sum is $\binom{24}{12} \binom{24}{12} = 2704156^{2704156}$, which is rather large.⁵ Clearly, we need a different approach.

3 Extending Voting Rules By Goalbase Ballots

Rather than electing a committee by voting for individual candidates, we might consider voting for *properties* of the winning committee. What Valerie wants is to indicate her preference for “Alice-XOR-Bob-ness” in the winning committee. Now we present a balloting framework in which this is expressible.

First, we require some definitions, following [21]:

Definition 3 (Goalbases and Utility Functions). A goalbase G is a set of weighted propositional formulas (φ, w) , where φ is formed from Boolean connectives and propositional symbols from the set $\mathcal{PS}$, and $w \in \mathbb{R}$. A utility function $u: 2^{\mathcal{PS}} \rightarrow \mathbb{R}$ maps models $M \subseteq \mathcal{PS}$ to their values. Each goalbase G generates a utility function u_G such that $u_G(M) = \sum_{(\varphi, w) \in G, M \models \varphi} w$. $\text{Var}(G)$ is the set of propositional symbols occurring in G .

Imagine a voting rule where voters submit goalbases with binary weights as ballots. Valerie might then wish to cast $G = \{(\neg a \vee \neg b, 1), (a \vee b, 1)\}$ as her ballot. The utility function u_G which this ballot generates,

$$u_G(M) = \begin{cases} 2 & \text{if } a \in M, b \notin M \text{ or vice versa,} \\ 1 & \text{if } a, b \notin M, \\ 0 & \text{if } a, b \in M, \end{cases}$$

⁴ A set of size n may be partitioned into k nonempty subsets $\binom{n}{k}$ ways (where $\binom{n}{k} = \frac{1}{k!} \sum_{j=1}^k (-1)^{k-j} \binom{k}{j} j^n$ denotes a Stirling number of the second kind [11, Section 6.1]), and in each case these k subsets may themselves be strictly ordered in $k!$ ways. Thus the number of (not necessarily strict) linear orders on n items is $\sum_{k=1}^n k! \binom{n}{k} = \sum_{k=1}^n \sum_{j=1}^k (-1)^{k-j} \binom{k}{j} j^n$.

⁵ For comparison, an estimated 106 billion people had ever lived as of the year 2002 [12]. The profile space is more than adequate to permit every person who has ever lived a unique opinion on the 2003 Game Theory Society committee election.

induces an ordering on the committees which coincides exactly with Valerie’s preference ordering. The models acd and bde score 2, ahead of cde which scores 1, ahead of abc which scores 0.

Here we formulate a general voting rule using goalbases as ballots, in which winners correspond to the optimal states over the sum of all voters ballots:

Definition 4 (Goalbase Summation). If G, G' are goalbases, then

$$G \oplus G' = \left\{ \left(\varphi, \sum_{(\varphi, a) \in G} a + \sum_{(\varphi, b) \in G'} b \right) \mid \varphi \in \text{For}(G \cup G') \right\}$$

is their formula-wise sum, where $\text{For}(G) = \{\psi \mid (\psi, w) \in G\}$.

Definition 5 (Optimal k -Sized Models). The set $\text{opt}_k(G)$ is the set of optimal k -sized models according to the goalbase G :

$$\text{opt}_k(G) = \underset{\substack{M \subseteq \mathcal{PS} \\ |M|=k}}{\text{argmax}} u_G(M).$$

Definition 6 (k -Seat Goalbase Summation Rule). The k -Seat Goalbase Summation Rule (GSR) is the voting rule where voters $v \in \mathcal{V}$ submit goalbases as ballots and the winning committees are the members of $\text{opt}_k(\bigoplus_{v \in \mathcal{V}} G_v)$.

Our first observation about the GSR is that when $k = 1$, we have a voting rule which generalizes a wide variety of standard single-winner rules in the sense that it accepts all of their ballots (after a bijective mapping to goalbases) as valid ballots itself. For example, plurality ballots correspond to goalbases $\{(c, 1)\}$ where c is the voter’s top choice; approval ballots correspond to goalbases $\{(c, 1) \mid c \in \mathcal{A}\}$, where $\mathcal{A}$ is the set of the voter’s “approved” candidates. In fact, there is an obvious mapping for all m -vote ℓ -support top- k voting rules:

Proposition 7. Let $\mathcal{B}$ be a set of m -vote ℓ -support ballots. For each ballot $b = v_{c_1}, \dots, v_{c_n} \in \mathcal{B}$, represented as a vector of votes for each candidate, define the goalbase $G_b = \{(c_i, v_{c_i}) \mid 1 \leq i \leq n\}$, collecting all $G_b \in \mathcal{G}$. Then k -GSR with ballots $\mathcal{G}$ will choose the same winners as the m -vote ℓ -support top- k rule with ballots $\mathcal{B}$.

On its face, this is just a baroque way of rewriting the ballots of existing voting rules, so is not interesting by itself; rather, the interest is in what it points to: Voting rules we can *simulate* using GSR we can also *extend*. The problem we identified in Section 2 was that multi-winner extensions of single-winner voting rules have too little expressive power when voters have non-independent preferences. The ballots used by each of these rules are a proper subset of all possible goalbase ballots; therefore, relaxing the restrictions on allowable ballots is a direct way of increasing the expressivity of such rules.

This relaxation of ballot restrictions may be done in both the single- and multi-winner cases. We briefly consider the properties of 1-GSR:

Recall that cumulative vote is the single-winner voting rule where each voter is permitted to distribute some number of votes m over the candidates however he sees fit. (That is, cumulative vote is the m -vote m -support top-1 rule.)

Fact 8. If no restrictions are placed on ballot goalbases, then 1-GSR is cumulative voting without vote limits.

Cumulative voting without vote limits is not a practical voting method, since for voters it is equivalent to playing the game of which voter can write down the largest number for his favored candidate. (Unrestricted 1-GSR is the finite-vote, finite-support top-1 rule.) However, if we restrict the weights appearing in the goalbase ballots to some range of $\mathbb{N}$, then we have the following correspondence:

Fact 9. *If each goalbase G_i uses weights from $0, \dots, n \in \mathbb{N}$ only and $\sum_{(\varphi, w) \in G_i} w \leq m$, then 1-GSR corresponds to m -vote cumulative voting.*

The following is an example of how relaxing restrictions on goalbases can yield new voting rules:

Example 10. Call Property Approval Voting (PAV) the voting rule in which *properties of the outcome* (rather than individual candidates) are the objects of approval or disapproval. Any goalbase G having 1 as the sole permissible weight for any formula constitutes an admissible PAV ballot. However, some formulas will be useless: Any formula which implies a conjunction of positive literals longer than the intended number of winners, and any formula which implies a conjunction of negative literals longer than the intended number of losers, will effectively be equivalent to $\perp$. A significant difference between AV and PAV is the range of preorders of which they permit representation. Every AV ballot induces a dichotomous order, while PAV supports much more. In the case where there are three candidates a, b, c , the PAV ballot $\{(a, 1), (a \vee b, 1)\}$ induces the (non-dichotomous) order $a > b > c$, since the state $\{a\}$ receives two points, $\{b\}$ one point, and $\{c\}$ zero points. (Only singleton states are relevant here, since we are considering the single-winner case.)

In fact, there is a general way of representing any strict linear order $a_1 > a_2 > \dots > a_n$ with a PAV ballot:

$$\begin{aligned} & (a_1 \vee \dots \vee a_{n-2} \vee a_{n-1}, 1) \\ & (a_1 \vee \dots \vee a_{n-2}, 1) \\ & \vdots \\ & (a_1 \vee a_2, 1) \\ & (a_1, 1) \end{aligned}$$

The clause which ends with a_i is the one which causes a_i to be ordered strictly above a_{i+1} , so by omitting that clause we can get a ballot where $a_i \sim a_{i+1}$. This is sufficient to induce any weak linear order over the candidates. Thus, in the single-winner case, PAV is a variant of the Borda rule.

4 The Complexity of Deciding Winning Slates

If we are to use goalbases as ballots, then in the multi-winner case finding winning committees is an optimization problem. The problem of deciding whether some utility threshold may be reached given a goalbase representation of a utility function arises in other applications of goalbases (e.g., when using goalbases as combinatorial auction bidding languages as in [20, Chapter 6]). Following [21], we give a formal definition of this decision problem, known as MAX-UTIL:⁶

Definition 11 (MAX-UTIL). *The decision problem MAX-UTIL has as input a goalbase G and $K \in \mathbb{Z}$, and outputs 1 if some $M \in 2^{\mathcal{PS}}$ has $u_G(M) \geq K$, and 0 otherwise.*

Determining the winner of an election where goalbases are ballots is related to solving MAX-UTIL for the sum of those goalbases. MAX-UTIL for goalbase languages with straightforward definitions tends to be either trivial or NP-complete (see [21, Section 5] and [20, Section 5.5]).

Because we are concerned here with electing committees of a size fixed prior to the election (as opposed to the open-ended committees

⁶ In [20, 21] a parametrized version of MAX-UTIL is used. We treat the parameters informally here.

discussed by Brams *et al.* [6]), we cannot apply MAX-UTIL directly to the sum of voters' goalbases in order to determine the winners of the election. Doing that might yield a model with the wrong number of winners. We must do something to ensure that only models which fill k seats are potentially optimal. One approach to adapting our winner determination problem to MAX-UTIL is to augment the sum of the voters' goalbases with formulas which increase the utility of k -sized models (or decrease the utility of non- k -sized models).

First, some notation is required. Define the following formulas:

$$\begin{aligned} \varphi_{\geq k} &= \bigvee \left\{ \bigwedge X \mid X \subseteq \mathcal{PS} \text{ and } |X| = k \right\} \\ \varphi_{\leq k} &= \neg \varphi_{\geq k+1} \\ \varphi_{=k} &= \varphi_{\leq k} \wedge \varphi_{\geq k} \end{aligned}$$

and the quantity

$$\delta = \sum \left\{ |w| \mid (\varphi, w) \in \bigoplus_i G_i \right\},$$

where G_i is the goalbase of voter i . The formula $\varphi_{=k}$ is such that a model $M \models \varphi_{=k}$ iff $|M| = k$. The quantity δ is a (not necessarily tight) upper bound on utility change between arbitrary models for $u_{\bigoplus_i G_i}$. Note that $\varphi_{\geq k}$ has $\binom{n}{k}$ disjuncts, and so is potentially a very long formula.⁷ However, in the context of committee elections n and k —the numbers of candidates and seats—will tend to be small, and, as will be seen below, $\varphi_{=k}$ appears exactly once in the goalbase which represents the voters' preferences.

Suppose that $\bigoplus_i G_i$ is the sum of voter goalbases in a k -seat committee election. Let

$$G = \left(\bigoplus_i G_i \right) \oplus \{(\varphi_{=k}, \delta + 1)\}.$$

Since δ is an upper bound on utility change between models for $u_{\bigoplus_i G_i}$, we can say the following: If M, N are models such that $M \models \varphi_{=k}$ and $N \not\models \varphi_{=k}$, then $u_G(M) > u_G(N)$, as the greatest possible utility loss of moving from N to M in $u_{\bigoplus_i G_i}$ is δ , and making $\varphi_{=k}$ true results in a gain of $\delta + 1$. Thus, since any model of size k is strictly better than every model of any other size, we are guaranteed that all models which yield maximal utility are of size k . Moreover, since $\varphi_{=k}$ is true on *every* model of size k , it does not affect their utility relative to one another, so augmenting $\bigoplus_i G_i$ with $(\varphi_{=k}, \delta + 1)$ preserves the ordering of (relevant) models.

Therefore, we may easily adapt the input to force size- k models to the top of the ordering and use an off-the-shelf algorithm for deciding MAX-UTIL to determine winners—though this may be impractical due to the complexity of MAX-UTIL for unrestricted goalbases. If $\bigoplus_i G_i$ is confined to something less than the full goalbase language, however, we may be able to use that to our advantage. The following proposition shows that when solving MAX-UTIL we may always reduce a goalbase outside a given language $\mathcal{L}$ to a goalbase inside the language by solving a simpler version of MAX-UTIL at most an exponential number of times:

⁷ While it is not possible to shorten $\varphi_{\geq k}$ using standard Boolean connectives, we can be more concise if we are willing to augment our language with a cardinality operator. For example, Benhamout *et al.* [1] consider a variant of propositional logic in which there are *pair formulas* $(\rho, \mathcal{L})$, where $\mathcal{L}$ is a multiset of literals and ρ specifies how many elements of the multiset must be true in order for the $(\rho, \mathcal{L})$ to be true. Clearly $(|\mathcal{PS}|/2, \mathcal{PS})$ is equivalent to $\varphi_{\geq |\mathcal{PS}|/2}$, but exponentially shorter. Hoos and Boutilier [13] propose a similar, though less powerful, k -of operator—less powerful due to the fact that their (bidding) language lacks negation, and so any k -of operates on atoms only.

Proposition 12. *If $G \in \mathcal{L}$, $G' \notin \mathcal{L}$, and $\mathcal{L}$ is closed under substitution of logical constants for atoms, then MAX-UTIL for $G \oplus G'$ can be solved with $\leq 2^{|\text{Var}(G')|}$ calls to a MAX-UTIL oracle for $\mathcal{L}$.*

Proof. There are $2^{|\text{Var}(G')|}$ models on just the variables occurring in formulas in G' . For each model over the variables in G' , we substitute $\top$ and $\perp$ into $G \oplus G'$ as per the model and carry out MAX-UTIL on the modified $G \oplus G'$. $\square$

If $\text{Var}(G')$ is small and does not depend on $\mathcal{PS}$, decomposing a goalbase containing alien formulas in this way is potentially feasible. However, the formula $\varphi_{=k}$ contains every atom in $\mathcal{PS}$ at least once and hence the upper bound we get is exponential in $|\mathcal{PS}|$, which is unhelpful. (For a discussion of the substitution closure condition, see [20, Section 5.7.1].)

An alternative approach is to modify the decision problem instead of the goalbase. Perhaps MAX-UTIL is not the right decision problem unless we have the same number of candidates as seats—in which case, why vote? Instead, we define a variant of MAX-UTIL where exactly k atoms must be true in any solution:

Definition 13 (k -MAX-UTIL). *The decision problem k -MAX-UTIL has as input a goalbase G and $K \in \mathbb{Z}$, and outputs 1 if some $M \in \text{opt}_k(G)$ has $u_G(M) \geq K$, and 0 otherwise.*

k -MAX-UTIL is the decision-problem version of finding members of $\text{opt}_k(G)$, just as MAX-UTIL is the decision-problem version of finding members of $\text{opt}(G) = \bigcup_{i=0}^{|\mathcal{PS}|} \text{opt}_i(G)$.

Fortunately, having a fixed number of seats to fill dramatically reduces the complexity of finding a voter's preferred ballot:

Proposition 14. k -MAX-UTIL $\in \text{P}$, for fixed $k \in \mathbb{N}$.

Proof. For any given k and $\mathcal{PS}$, there are $\binom{|\mathcal{PS}|}{k}$ models of size k to check. It is always the case that $\binom{n}{k} \leq \frac{n^k}{k!}$, which grows polynomially in $n = |\mathcal{PS}|$ for any fixed k . $\square$

This makes *whatever* goalbase language we want for representing our voters' ballots computationally tractable (though not necessarily trivial) so long as the number of seats and candidates is not too large. In particular, it is well within the capabilities of contemporary desktop computers to determine the winners in committee elections of a size similar to that conducted by Game Theory Society in 2003 (see Section 2), where there would be only 2.7 million models to check. In practice, an off-the-shelf integer program solver could be used to find winners, since any set of ballots can be converted to an integer program.

5 Discussion & Future Work

Many paths are yet to be explored. In this section we give an overview of those of which we are aware.

In order to use goalbase ballots for multi-winner voting, we must place some restriction on the weights which are available to voters. As noted after Fact 8, cumulative voting without point limits is not a sensible voting method. Having established that restrictions are needed, we are faced with the problem of selecting some—it is not presently obvious which restrictions are most suitable. The restriction which cumulative voting itself suggests is to limit the sum of weights in any goalbase: $\sum_{(\varphi, w) \in G} w \leq K$.⁸ This is a limit on the input

⁸ If we permit negative weights, then we would need to place an upper bound on the sum of the absolute values of the weights instead of on the sum of the weights. In this way we avoid ballots like $\{(a, -2^{1000}), (b, 2^{1000} + 10)\}$ which the voter could otherwise claim is a 10-point ballot.

space. Another approach is to restrict the output space: For example, we might limit the utility of any admissible state: $u_G(M) \leq K$ for all $M \subseteq \mathcal{PS}$ where $|M| = k$.

There are advantages and disadvantages to both methods. If our voter is a person, he will find it easier to cast a valid sum-limited ballot than a valid state-limited one. Input limits are not uncommon: For example, in the U.S., the State of Illinois used single-winner cumulative voting with a 3-point limit (i.e., 3-vote 3-support top-1) within districts for electing members of its House of Representatives from 1870 to 1980 [19, 22]. Corporate boards of directors are usually elected using cumulative voting, where the point limit for each voter is the number of shares he owns. We know of no uses of output limits. Presumably this is because it is hard to see when working in the input space whether output limits are being respected; output limits expect too much of the average voter. However, output limits on elections of the size human voters are likely to face will not be difficult for machines to enforce, so might be practical if voters use a computer-aided voting system. This is a user-interface issue.

Point limits also raise a fairness issue. For simplicity, we use a single-winner example, though the problem it illustrates is general. The sum-limit $\sum_{(\varphi, w) \in G} w \leq K$ will not always produce utility functions having equal sums for singleton models. E.g., consider the goalbase ballots $G_1 = \{(a, 10)\}$ and $G_2 = \{(a \vee b, 10)\}$. The latter has a singleton state sum of 20 ($u_{G_2}(\{a\}) = 10, u_{G_2}(\{b\}) = 10$), while singleton states for the former sum only to 10 ($u_{G_1}(\{a\}) = 10$). The sum-limit gives voters with top-heavy preferences more influence on the outcome than voters with balanced or bottom-heavy preferences. We could try to mitigate this by “normalizing” formulas based on the number of states they affect, e.g., $(a \vee b, 10)$ could be translated to $(a, 5), (b, 5)$, but this would seem to disadvantage voters who have top-heavy preferences. If $(\bigvee \mathcal{PS} \setminus \{a\}, 10)$, after normalization, gives one point to everyone but candidate a , that is not likely to be very effective for voters who dislike a but otherwise do not distinguish among the other candidates. Or we could try other ways of normalizing—Lafage and Lang [14, Section 3.2.3] suggest postprocessing (dis)utilities to equalize entropy across agents—which will potentially have some other differential effect on voters.

The basic question here seems to be how to set the value of preferences which are not over single states against those which are. What is an appropriate measure of voting power here? Input limits seem to favor top-heavy voters, output limits seem to favor bottom-heavy voters. One way of quantifying the effect that a proposed weight limit could have is by considering the *efficacy* of voters with different preferences under that weight limit. (The efficacy of a ballot for a voter is a measure of how often that voter will be pivotal if he casts that ballot.) Ideally, all voters would have equally efficacious ballots to cast. Brams and Fishburn [3, Chapter 5] calculate the efficacy of ballots for approval voting and find that not all ballots are equally effective. If we assume that our voters are truthful, what this means is that approval voting is advantageous for voters with some kinds of preference orders and disadvantageous for others. A similar analysis could be done for cumulative voting with goalbase ballots, with an eye to which weight restrictions treat voters most equitably.

With any voting system, there are questions about whether it encourages or discourages strategic voting. The manipulability of a voting system must always be considered in the context of a notion of sincerity, for we cannot say whether a voter is *misrepresenting* his preferences if we cannot first say what it would be for a voter to represent his preferences accurately.

Consider, first, voting systems with ordinal ballots. Many standard systems—e.g., plurality, approval, Borda—use ballots which contain

purely ordinal information. In the case where there is an allowable ballot which induces the same preorder over outcomes as the voter's true preorder, then any reasonable notion of sincerity should deem that ballot sincere (and any ballot which does not induce that same preorder, insincere). This means, for example, that for voters with strict linear orders, there will always be unique sincere plurality and Borda ballots; and similarly, for voters whose preferences are dichotomous (and not wholly indifferent), there will be a unique sincere approval ballot. However, this will not be the case for voters with other kinds of preorders. There are no approval ballots which express nondichotomous preferences (e.g., $x > y > z$); standardly, Borda does not permit ties, so voters with weak (instead of strict) orders will have no ballots which express their preferences exactly.

What should count as sincere in the space of cardinal ballots is not immediately obvious. A voter's preferences may be inexpressible as a result of restrictions on the ballot language, and this can result in the existence of multiple sincere ballots which the voter could cast. Endriss [9] explores the existence of multiple sincere ballots for approval voting and shows that the Gibbard-Satterthwaite Theorem is avoidable in that context; Endriss *et al.* [10] present several measures of sincerity for languages where ballots are preorders, and examine the consequences for strategyproofness under these. This line of research could be continued for goalbase ballots, first by developing reasonable notions of sincerity, and secondly by determining which language restrictions induce sincerity in rational voters. Meir *et al.* [17] avoid the problem of sincerity in multi-winner voting altogether by defining manipulation as an optimization problem asking whether, given the ballots of some other voters, there is a ballot which the manipulating voter may cast which yields him at least t utility. The question of whether a better ballot exists is more general than, and serves as a proxy for, the question of whether a better insincere ballot exists—though this still leaves open the possibility that some ballot which is optimal is nonetheless also sincere, and so does not exactly capture classical manipulability.

Finally, we might consider questions about the difficulty of finding a sincere ballot given a voter's preferences. It would not be surprising to learn that for some languages, it is always in the voter's best interests to cast a sincere ballot, but nonetheless quite difficult for him to determine which ballots are sincere for him. Strategyproofness is not worth much in this case. A method for constructing sincere ballots will be essential for any language intended for human voters.

6 Conclusion

In this paper we considered some methods for electing committees and demonstrated that they lack properties which are desirable when conducting multi-winner elections. In particular, single-winner voting methods lack the expressivity to extend well to the multi-winner case. The observation that it is possible to simulate many single-winner voting methods using the Goalbase Summation Rule, which takes goalbases as ballots, suggests one way of extending the expressivity of existing voting methods for use in a multi-winner setting. Because multi-winner elections tend to have the number of winners fixed beforehand, the complexity of determining winners (using the decision problem MAX-UTIL) is limited, even when the goalbase language is unrestricted. Along these lines, we suggest as an example a multi-winner extension of approval voting, which we call Property Approval Voting. Finally, we discuss the possibilities for future work: the need to find useful limits on weights in goalbase ballots; the fairness of these limits, since they may differentially affect voters with dissimilar preferences; and issues related to sincerity and strategic voting.

ACKNOWLEDGEMENTS

This work is not entirely disjoint with [20, Chapter 7], which was supported by a GLoRiClass fellowship funded by the European Commission (Early Stage Research Training Mono-Host Fellowship MEST-CT-2005-020841).

REFERENCES

- [1] B. Benhamou, L. Sais, and P. Siegel, 'Two proof procedures for a cardinality based language in propositional calculus', in *STACS 94, 11th Symposium on Theoretical Aspects of Computer Science*, eds., P. Enjalbert, E. W. Mayr, and K. W. Wagner, pp. 71–82, Springer, (1994).
- [2] J.-P. Benoît and L. A. Kornhauser, 'Social choice in a representative democracy', *American Political Science Review*, **88**(1), 185–192, (1994).
- [3] S. J. Brams and P. C. Fishburn, *Approval Voting*, Springer, second edn., 2007.
- [4] S. J. Brams, D. M. Kilgour, and M. R. Sanver, 'A minimax procedure for negotiating multilateral treaties', in *Reasoned Choices: Essays in Honor of Academy Professor Hannu Nurmi*, ed., M. Wiberg, 108–139, Finnish Political Science Association, Turku, Finland, (2004).
- [5] S. J. Brams, D. M. Kilgour, and M. R. Sanver, 'How to elect a representative committee using approval balloting', in *Mathematics and Democracy, Studies in Choice and Welfare*, 83–95, Springer, (2006).
- [6] S. J. Brams, D. M. Kilgour, and M. R. Sanver, 'A minimax procedure for electing committees', *Public Choice*, **132**(3–4), 401–420, (2007).
- [7] J. R. Chamberlin and P. N. Courant, 'Representative deliberations and representative decisions: Proportional representation and the Borda rule', *The American Political Science Review*, **77**(3), 718–733, (1983).
- [8] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet, 'Preference handling in combinatorial domains: From AI to social choice', *AI Magazine*, **29**(4), 37–46, (2008).
- [9] U. Endriss, 'Vote manipulation in the presence of multiple sincere ballots', in *Proc. 11th Conf. on Theoretical Aspects of Rationality and Knowledge (TARK-2007)*, ed., D. Samet, pp. 125–134, Presses Universitaires de Louvain, (2007).
- [10] U. Endriss, M. S. Pini, F. Rossi, and K. B. Venable, 'Preference aggregation over restricted ballot languages: Sincerity and strategy-proofness', in *Proc. 21st Intl. Joint Conf. on Artificial Intelligence (IJCAI-2009)*, ed., C. Boutilier, pp. 122–127, (2009).
- [11] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley, second edn., 1994.
- [12] C. Haub, 'How many people have ever lived on earth?', *Population Today*, **30**(8), 3–4, (November/December 2002).
- [13] H. H. Hoos and C. Boutilier, 'Solving combinatorial auctions using stochastic local search', in *Proc. 17th Conf. on Artificial Intelligence (AAAI 2000)*, pp. 22–29, AAAI Press, (2000).
- [14] C. Lafage and J. Lang, 'Logical representation of preferences for group decision making', in *Proc. 7th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR-2000)*, eds., A. G. Cohn, F. Giunchiglia, and B. Selman, pp. 457–468, Morgan Kaufmann, (2000).
- [15] J. Lang, 'Logical preference representation and combinatorial vote', *Annals of Mathematics and Artificial Intelligence*, **42**(1–3), 37–71, (2004).
- [16] J. Lang and L. Xia, 'Sequential composition of voting rules in multi-issue domains', *Mathematical Social Sciences*, **57**(3), 304–324, (2009).
- [17] R. Meir, A. D. Procaccia, J. S. Rosenschein, and A. Zohar, 'Complexity of strategic behavior in multi-winner elections', *Journal of Artificial Intelligence Research (JAIR)*, **33**, 149–178, (2008).
- [18] B. L. Monroe, 'Fully proportional representation', *The American Political Science Review*, **89**(4), 925–940, (1995).
- [19] B. F. Moore, *The History of Cumulative Voting and Minority Representation in Illinois, 1870–1908*, volume III.3 of *The University Studies*, University Press, Urbana-Champaign, Illinois, 1909.
- [20] J. Uckelman, *More Than the Sum of Its Parts: Compact Preference Representation Over Combinatorial Domains*, Ph.D. dissertation, University of Amsterdam, 2009. ILLC Publication DS-2009-12.
- [21] J. Uckelman, Y. Chevaleyre, U. Endriss, and J. Lang, 'Representing utility functions via weighted goals', *Mathematical Logic Quarterly*, **55**(4), 341–361, (2009).
- [22] 'Alternative voting systems as remedies for unlawful at-large systems', *The Yale Law Journal*, **92**(1), 144–160, (1982).

Preferences, Utilities, and Quantitative Judgments

Richard J. Wallace and Gordon Rios¹

Abstract. This paper is concerned with a major hiatus in prescriptive decision analysis, which of course includes portions of the AI preferences field when used in interactions with human beings. In such settings, all multi-attribute decision aids must elicit some sort of preference ordering, and this entails quantitative judgments. Therein lies the rub: since all quantitative judgments involve cognitive processes, they have certain properties that may or may not match the properties assumed by the corresponding model. In addition, such judgments are subject to biasing effects, such as range or centering effects, in varying degree depending on the nature of the judgment. Fortunately, there is a field of psychology, called psychophysics, that deals specifically with such judgments and their properties. This paper surveys the field and explains how it can be applied to assess the soundness of quantitative judgments made in the context of constructing preference orderings, and what general lessons might be learned. Note: this is distinct from issues of descriptive adequacy of the models, which may be entirely irrelevant in the prescriptive setting.

1 INTRODUCTION

In the area of preference handling in artificial intelligence, one important concern is the design of systems to support human decision making. This falls under the larger heading of prescriptive decision analysis. The latter term is used to designate the application of formal or semi-formal procedures, based in part on assumptions about rationality, in order to aid decision makers (DM) [2]. In the middle of the last century, it split off from the older area of normative decision analysis, which is concerned with characterising rational decision making according to first principles as well as with formal properties of certain forms of orderings. Both areas should be distinguished from descriptive decision analysis, which is concerned with decision making as an empirical phenomenon. (Some authors have suggested adding “constructive decision analysis” to this triad [7]. However, the term “constructive” in this context is not commensurate with the other three categories, since it refers to a certain perspective on prescriptive – and descriptive – decision making, rather than a distinct form of endeavour.)

Within the prescriptive framework, we are generally concerned with situations in which there are several or many criteria by which the DM might judge his/her choices. (In simpler situations it is expected that the DM can order choices unaided.) We are therefore in the domain of multi-criteria decision analysis (MCDA). There are numerous formal or semi-formal methods in MCDA; all involve an analytical phase, where a complex alternative is decomposed into a set of attributes, and a compositional phase, where the ordered or scaled preferences for single attributes are combined or aggregated

into a single ordering or scale of value, the latter usually being designated as a “utility scale”.

Ordering or scaling involves judgments made by the human decision maker. In decision aiding of this form there is, therefore, an unavoidable psychological element. This is because human judgments are psychological phenomena, with certain properties that must be elucidated by empirical investigation. As a result, any prescriptive method involves the assumption that the particular kind of judgment carried out under that method has properties commensurate with those specified by the model of preferences that the method assumes. Deciding whether the assumption is met, therefore, depends in part on empirical tests of the features of the judgment made by the DM.

Note that we are not concerned with (nor will we discuss) the descriptive adequacy of these models. So we will not present a Kahnemann-and-Tversky-like discussion of the vagaries of human decision making [39]. In our view, this is a red herring since MCDA models are not meant to be psychological models of choice. Following Roy [30], we consider these methods as decision aids, and we take the view that MCDA models are built to serve that purpose. This means that the orderings or utility scales derived in the prescriptive context are artificial, i.e. they are prosthetic devices of a sort. In fact, they constitute a form of *human artificial intelligence*, that is, artificial intelligence in which a person is an essential component rather than a device like a computer. (This is discussed at greater length elsewhere; see [41].)

The question, then is, can we interrogate the psychological literature to determine whether the judgments in question have been studied, and if so, can we determine whether the features of these judgments are consistent with the features implied by the decision model? Since all decision models involve “preference elicitation”, which comes down to psychological scaling, the issue can be phrased succinctly as: what is the nature of the human DM’s scaling abilities?

We have two objectives in this endeavour. One is to avoid introducing bias in DM’s judgments. The other is to avoid asking DM for judgments that he/she can’t really carry out – at least not in the manner prescribed by the model.

For brevity, in this paper we largely restrict ourselves to riskless choice. This is in some ways unfortunate because of the importance of expected utility models and the special questions that arise concerning judgments made in this context. However, the basic issues we address can all be discussed within this restricted arena.

One last introductory note before we start. This paper is not based on a reading of a single textbook or a few well-known papers. It is derived from a study (by the first author) of a large swath of the literature on decision making as well as several hundred papers covering most of the field known as “psychophysics” (see below). The significance of this will be made clearer toward the end of the paper.

¹ Cork Constraint Computation Centre and Department of Computer Science, University College Cork, Cork, Ireland. {r.wallace, g.rios}@4c.ucc.ie

2 WHAT JUDGMENTS ARE WE TALKING ABOUT?

Every preference or utility model, when it is applied to human decision making, involves some sort of quantitative judgment. However, such judgments differ depending on the model, particularly in the field of MCDA. Here, we will consider several prominent models and the judgments required. We assume that the attributes and values have already been decided on; we are concerned with the judgments made by DM in the course of constructing a preference ordering, usually via a utility *scale*. We assume the standard vector-of-values model of alternatives throughout, with various kinds of aggregation specified by different models. We also assume a single DM, although collective decisions can often be made with the same formalisms and judgments, cf. [17].

Here, we also point out a textual notation used throughout the text: single quotes are scare quotes, for figurative or otherwise problematic terms, while double quotes around single words or compound terms are quotation marks meant to indicate standard terminology.

The classical multi-attribute utility model is based on the notion of an additive utility function, which involves scaling utilities of separate attributes and then combining them according to an additive model [17]. Since we are talking about riskless choice, we will defer to [17] and call this the multi-attribute value model, with acronym MAVT, although the first author prefers to use the term “utility” in its generic sense, as a “subjective scale of values” [25], and “expected utility” to designate utility based on prospects. Aggregation of this sort requires coefficients associated with each individual utility, which along with the individual utilities must be based on some kind of human judgment. This judgment consists of specifying an interval on one utility scale that matches a given interval on another. That the judgment is meant to reflect the idea of comparing marginal rates of substitution is not of concern here. What is important is that the judgments is one of comparing two differences in intensity; moreover, these are differences assumed to be associated with two functions derived from the same non-sensory continuum.

Although the basic MAVT procedure is that just described, there are a number of reports in the literature regarding additive models, where “surrogate” judgments are used to estimate single-dimensional utilities and attribute coefficients, or weights. Among the non-probabilistic methods, these include bisection, direct ratings, and “ordered metrics”; the first uses a numerical interval, the second usually involves a discrete integer scale such as $[-3, +3]$, the third involves comparison of differences [13].

Perhaps the most prominent surrogate MAVT method is SMART [9]. In SMART, after ranking the attributes, their weights are determined by assigning a standard value of 10 to the lowest-ranked attribute (a modulus) and assigning a number to each successive attribute in the ranking that reflects the *ratio* of importance between it and the modulus (a “multiplication” judgment). Values are ordered by estimating their positions on a 0-to-100 scale.

The Analytical Hierarchy Process (AHP) model involves ideas of ratio scaling. In line with this idea, DM’s compare values of an attribute as well as attributes themselves by making ‘ratio’ judgments of the following sort: given two attribute-values (or attributes), the greater of the two is given a value of 1; the DM then has to give a value to the other that is some fraction representing the ratio in utility (or, in the case of two attributes, importance) between the two items. As we will see in the next section, in this case DM’s are making “fractionation” judgments.

Verbal Decision Analysis (VDA) was developed by Larichev and

his collaborators at the Institute for Systems Analysis in Moscow [19]. The VDA method that we will consider belongs to a group of methods collectively referred to as ZAPROS [20]. We assume a set of attributes and n attribute values; in addition, we assume a weak preference ordering over the values of each attribute. Given this, we can represent attribute vectors by vectors of ranks; e.g. (1,2,2) means the highest-ranked value for attribute 1, the second-highest for attributes 2 and 3. ZAPROS methods are based on a particular kind of paired comparison of attribute vectors. The comparison process is based on the idea of a reference vector, which is either (1,1,...,1) or ($n, n, \dots, n$), i.e. all of the best (highest-ranked) or all of the worst values. DM first compares pairs of near-reference vectors, where all but one attribute has the reference value; e.g. (1, 1, 2) versus (1, 2, 1). From this we can infer further preferences, e.g. (1, 1, 2) $\succ$ (1, 3, 1). There is, of course, more to the story, but this is sufficient for our purposes. Although it may not be immediately obvious, the judgment required here is the same as that for MAVT, i.e. a comparison of differences in value.

Outranking models, as the name indicates, involve assessing alternatives in terms of their ranking within each attribute (these are the “arguments in favour of the statement ‘ a outranks b ’” [40], p. 59). In the ELECTRE I model, the user must, therefore, compare values within each attribute and also assign attribute weights [29]. The former can be accomplished by paired comparison judgments. Although it is not clearly specified in the model, it appears from recent work that ratio judgments are used in determining weights [12].

In contrast to the MCDA systems mentioned above, some prominent preference models in AI, such as CP-Nets [6], require only simple ranking judgments. Although it is not entirely clear from descriptions such as [8], it appears that paired comparison judgments would be sufficient across each domain, for each combination of parent values.

Some readers may be thinking at this point, “Well, of course people can make these kinds of judgments. So what’s the problem?” The problem is this. People will carry out a task such as fractionation (e.g. “ X is one-half the intensity of Y ”) or interval comparison and may even return results that are more-or-less consistent (although it is usually not clear from reported cases how often they are asked to make the *same* judgment during a decision analysis session, or to what degree inconsistencies are ‘corrected’). This in itself does not mean that their judgments actually correspond to the often high-powered assumptions entailed by the model. As we will see, people are very good at using whatever cues are available in order to “meet expectations”, in this case the expectations of the decision analyst. As a rule, careful investigation is required to determine the nature of the judgments made in these situations, and to determine whether human DMs even have the abilities that are being assumed. This will become clear in the next section. To put the point another way, the psychological evidence concerning human quantitative judgments, including models, may allow us to evaluate this issue on the basis of empirically-derived *principles*. And this will give us a better idea of whether a particular MCDA procedure is justified than we can get from ‘intuition’ or experience in the decision analysis situation.

3 PSYCHOPHYSICS AND SCALING

3.1 Basic ideas and findings

We have now seen something of the judgments that the DM is expected to make in the decision analysis situation. Next, we will try to determine whether human beings can actually make these judg-

ments in the manner required by the decision model. For this purpose, we turn to an area of psychology that has been concerned with these questions, but in more general terms, a field known as “psychophysics”.

Psychophysics can be thought of as the study of the ability to make quantitative judgments. The name given to the field reflects its original goal, which was to delve into the relation between the physical and mental worlds. As such, psychophysics began as a study of certain judgments related to sensory input, in particular, the relative intensities of two stimuli, for example, two lights presented together or in succession or two weights lifted in succession. In one case, the human “Observer” (O) might be asked to tell the experimenter which of the two lights was the brightest; in the latter case, O might be asked to say whether the second weight was heavier or lighter than the first. Toward the end of the 19th century, some investigators began to study more complicated judgments. An example is the bisection judgment: given two tones, say, of different intensities, O’s task would be to adjust a third tone so its loudness was midway between the other two.

One of the enduring obsessions of this field, dating from Fechner, who can be said to have founded psychophysics as a distinct area of inquiry, is the idea of deriving sensory *scales* that bear a functional relation to intensities along various physical continua [11]. Examples are scales of brightness, based on variations in light intensity (here, we are being purposely vague regarding the actual physical input), scales of loudness based on sound intensity, scales of heaviness based ultimately on differences in weight.

Fechner did not believe that “sensation” could be measured directly. Instead, he derived his scales from judgments of “just noticeable differences” (JNDs) between the intensities of two stimuli presented for comparison. A JND is defined as the smallest difference that leads to a judgment of difference ‘in sensation’. Here, Fechner used a simplification, which he called “Weber’s Law”, after an earlier investigator of these judgments. This ‘law’ states that the ratio of the JND to the intensity of the “standard” used for comparison is a constant, i.e.

$$k = \Delta I / I$$

where I is the intensity of the standard and $I + \Delta I$ is minimum intensity necessary or perceive/judge that the two stimuli are different along the given sensory dimension. By considering this relation in the limiting case and integrating across all JNDs in the limit, Fechner derived a logarithmic relation, “Fechner’s Law” or Massformel,

$$S = k * \log I$$

where S is the intensity of the sensation. This general procedure is now referred to as an indirect method of scaling sensations or sensory judgments.

Beginning in the 1930’s, but only gathering real momentum in the 1950’s, an alternative approach was developed, which used “direct methods” of scaling. These are related to earlier techniques like the bisection method mentioned above. In fact, the latter can be generalised to equisection methods (where the interval defined by the two boundary stimuli is to be divided into k equal parts), which are now usually called “category judgments”. Note that this form of judgment involves comparing differences. In addition to the conceptually straightforward equisection judgments, other scaling techniques are possible that are based on comparing differences.

In addition, there is a class of methods based on ‘ratio’ judgments. These can be either “estimation” or “production” methods; in the first case, O gives a numerical response to indicate the ratio between two stimuli, while in the second case O adjusts the intensity of a stimulus

so that it stands in a certain ratio to the standard. Examples of direct estimation include

1. judging the intensity-ratio between a pair of stimuli (“ratio estimation”),
2. given a standard stimulus intensity which is assigned a given number (the “modulus”), O judges a succession of stimuli, giving each a numerical value that represents the ratio in intensity between it and the standard along the prescribed continuum (“magnitude estimation”),
3. a variant of magnitude estimation in which no standard is given; instead, after familiarising himself with the range of stimuli presented, O gives numerical responses that are meant to reflect their relative intensities along the given continuum.

Using direct methods, S. S. Stevens and others have derived psychological scales for a large variety of perceptual continua [34]. In contrast to Fechnerian scaling, these scales are “power laws” of the form:

$$S = aI^n$$

More generally, (due in large part to disparities with the simple form of the exponential function that occur with low stimulus intensities) the ‘law’ has the form:

$$S = a(I - I_0)^n$$

where I_0 is called an “effective threshold”. Typically, in the expression on the right-hand side, the exponent has a value that is characteristic of the continuum being examined. For example, the exponent given by Stevens for loudness measured in decibels is 0.6.

An important development in psychophysics that occurred in first third of the 20th century, was the proposal by Thurstone that psychophysical methods could be applied to *non-sensory* phenomena, and that scaling procedures could also be used in these cases [37]. Among the attributes studied are attitudes (especially racial attitudes) [37], judgments of pleasantness and unpleasantness [1], and judgments of occupational prestige [15]. Obviously, this extension also takes in the notion of utility, and in fact Thurstone carried out some of the earliest studies that were meant to assess subjective value.

3.2 Issue 1: variability of judgment

In the decades that followed Fechner’s work, there was a certain amount of critical discussion concerning the notion of a sensory threshold, as revealed by psychophysical judgments. The main problem is that in psychophysical tests there is no clear separation between intensities that bring forth one kind of judgment, e.g. “lighter” or “uncertain”, and another, e.g. “heavier”. In fact, taking judgments of heaviness as an example, for a standard stimulus of 100g and a series beginning with 90g and ending with 110g, it is found that the proportion of “heavier” judgments increases steadily throughout the series. Thus, there is no evidence for a ‘transition point’ where there is a qualitative change in the perception of relative heaviness. Because of this, certain conventions had to be used in specifying a JND. Repeated judgments of the same stimulus intensity or difference in intensity were made, and the JND was defined as that difference where the proportion of “different” judgments equalled some specified fraction of the total, e.g. 0.5.

This basic finding eventually led to a new theoretical framework for judgments of this type, known as the Theory of Signal Detection (TSD). TSD was based on concepts from the field of signal processing, and was introduced as a model of psychological judgements in

the mid-50's [35]. The fundamental idea in TSD is that the channel involved in detecting a stimulus is noisy, so the stimulus, or signal, always occurs against a background of noise. The task of the observer is then to determine whether a given psychological event represents a signal or not. Both signal and noise are modelled as probability distributions along the intensity dimension (usually assumed to be Gaussian), and under 'threshold' conditions, these distributions overlap, perhaps to a considerable degree. The Observer must therefore use some kind of criterion to decide whether a signal occurred.

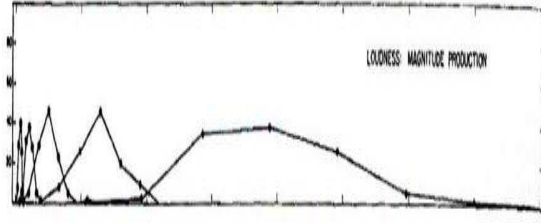


Figure 1. Distributions of judgments to specific stimuli in magnitude production task based on loudness.

Considerable variability is also found for direct estimation judgments. In some cases it can be quite large, and it increases with the intensity of the comparison stimulus [32, 21]. An example is shown in the figure above, which shows frequency distributions of loudness settings for one individual for different numerical values (each connected curve is for one such value); the abscissa is decibels ([32, top panel of Figure 8]). This is why in most studies central tendency statistics are used, in particular the median or the geometric mean.

Magnitude estimation judgments also show considerable inter-Observer variability. Even under the same experimental conditions, the exponents for different Observers can vary by at least 2:1; this was found for a variety of continua. The most reasonable explanation is that different Os are using different criteria for judgment in these cases. That some of this difference is due to different use of numbers in judging intensity has been shown by comparing power-law exponents for the same Os across modalities (in this case, weight, taste and smell). Positive correlations were found, which indicated that Os tended to use the same numerals in each case [26]. This suggests that Observers do not have a sound foundation for making such judgments, so they cannot apply numbers in these situations in a consistent fashion.

3.3 Issue 2: biases and contextual factors

Context effects in quantitative judgments take many forms (cf. [28]). At the same time, they can be divided into two rough groups: range and centering effects and sequential effects.

A classic example of range effects is by [14]. The task was to decide whether the stimuli within a certain range of intensities were more or less than half as loud as a standard (1000 Hz) tone of 90 db. There were three experimental groups which differed only in the range of intensities that they heard: a 55-65 db group, a 65-75 db group, and a 75-85 db group. A value for half-loudness was obtained by interpolation. The main findings were, (i) for each group of Os

the intensity value for half-loudness was within the range of intensities presented, (ii) despite large differences amongst themselves, even within one range, judgments for individuals were quite stable. The author interpreted the results as showing that Ss could not really judge whether a tone was half as loud as another (in his words, "Os have little or no idea what they are doing in this type of judgment" [14, p. 222]), and so they arbitrarily chose an intensity within the given range to represent this. Having made this selection, they are able to remember the value and make stable judgements. Similar results have been found with other sensory continua.

Sequential effects, where the current judgment is affected both by the prior stimulus and response are found for direct as well as indirect judgments. In general, they either take the form of assimilation (to the previous judgment) or contrast. For example, in magnitude estimation tasks there is a pronounced degree of assimilation to the previous presentation and response, and a weaker contrast effect with respect to presentations just before the last [16]. Partly as a result of such effects, researchers in the field have come to regard all such judgments as being 'relative' rather than absolute.

3.4 Issue 3: nature of 'ratio' judgments

Here, we briefly review two kinds of study. The first is concerned with testing properties that 'ratio' judgments should have if true ratio operations are involved. An example concerns tests of a property of ratio scales called the "product axiom". Given ϕ_i , the "psychological magnitude of Stimulus i " and $\phi_{ij} = \phi_i / \phi_j$, the product axiom states that for all triples i, j, k ,

$$\phi_{ij} \times \phi_{jk} = \phi_{ik}$$

The experimental evidence does not support this relation [10].

The second kind of study is concerned with the question of whether ratio and difference judgments are based on the same mental operation(s). It deals directly with the issue of whether Os in direct estimation experiments actually do what they are instructed to do. This work was originally inspired by a problem that arose when scales based on magnitude judgments were compared with scales based on category (involving equisection and therefore differences). The two kinds of scale seemed to have a nonlinear relation to each other. Torgerson [38] argued that there was one operation that underlay both kinds of judgment, and that it was probably a kind of difference operation. To state this more precisely, while the two-operation model says:

$$R_{ij} = J^*(s_j/s_i) \quad D_{ij} = J(s_j - s_i)$$

the one-operation model says:

$$R_{ij} = J^*(s_j - s_i) \quad D_{ij} = J(s_j - s_i)$$

where R and D are 'ratio' and 'difference' estimates, and s_j and s_i are subjective values of stimuli j and i [23].

Evidence supporting the one-operation model consists of demonstrations that certain context effects (e.g. stimulus spacing and distribution) affect 'difference' and 'ratio' judgments in the same way [24, 22], and that when Os perform both kinds of judgment their scales are monotonically related [4, 15]. Despite the fact that some studies have adduced evidence for the two-operation model, the preponderance of evidence supports the one-operation model, especially for non-sensory judgments. (For further discussion, see [18].)

3.5 Discussion

This short summary of some aspects of a very complicated field serves to indicate some pitfalls in asking people to make quantitative judgments. To quote the title of a paper by an authority in the area, "Quantitative subjective assessments are almost always biased, sometimes completely misleading" [27]. As a result, such judgments perhaps always require independent corroboration. In Lamming's view, Stevens' main mistake was to overlook this [18].

Stevens, in fact, made the remarkable assumption that naive Os could 'read out' some value of their sensations, analogous to taking a physical measurement, although he admitted that in practice his methods required various forms of randomization and averaging (which as Lamming says, removed "the net effect of those other judgmental factors which contributed more than 90 per cent of the variability" [18], p. 123). But is this not the same assumption made by decision analysts when they try to construct utility scales or "elicit preferences"? Except that here there is not even the possibility of varying an external stimulus intensity independently of judgments.

4 STUDIES OF UTILITY ASSESSMENT

Down through the years there have been studies that compared different MCDA methods using the same sets of alternatives. More often than not marked discrepancies are found in the results produced by different methods. Although it is difficult to pinpoint the reasons for this, if we consider the findings discussed in the previous section, such results are not surprising, and in some cases we can point out connections. To take one example, a comparison of AHP and MAVT (using surrogate direct rating judgments based on a 0-100 scale) showed systematic deviation from the line of perfect correlation [3]. This may be due in part to a range effect, since the range of numbers used with MAVT was much larger (AHP uses numbers between 1 and 9 inclusive).

Since it is perhaps the most problematic aspect of additive utility scaling, some studies have dealt specifically with the weighting of attributes. In one study the original MAVT method was compared with several surrogate judgments, including ratio, "swing weights", and "pricing out" [5]. Another study included comparison between two commercial systems, one based on tradeoff/indifference judgments, the other based on ratings using a slider anchored with the best and worst-ranked values [20]. In both studies there were marked inconsistencies among methods; again, this is not surprising once one appreciates the fact that different kinds of judgments can give different results and that often different methods involve different biasing effects, as noted above. An important conclusion is that 'surrogate' judgments (here ratings) are not genuine substitutes in practice.

Interestingly, the study by [5] also found that tradeoff/indifference judgments were less reliable than the others. This is an important point that accords with the earlier discussion of variability in judgment. At the same time, it should also be clear that, per [14], greater reliability does not imply greater validity; it may just be that in one case, previous judgments were easier to remember.

5 SOME INITIAL CONCLUSIONS

5.1 Prescriptions for prescriptive decision analysis

We conclude that, despite their elaborate conceptual machinery, MCDA techniques are basically heuristic in nature. Scaling 'utility' or even forming a preference ordering is, therefore, not akin to measurement in the physical sciences. This is a trap that has enticed

many, from Von Neumann and Morgenstern to S. S. Stevens. This does not mean that formal approaches are misguided per se. For one thing, they are the main vehicle we have to express normative concepts usually placed under the heading of principles of rationality. And, of course, they allow us to examine ideas like preference consistency in great depth.

Among the phenomena demonstrated in psychophysical experiments, some are particularly relevant here:

- Variability of judgments, which translates into unreliability in the present context (cf. [5]).
- Biasing effects, which include perceptual and response effects, plus the willingness on the part of the DM to make judgments even when there is evidence that the judgments don't actually match the assumptions behind them.
- Non-substitutability of surrogate judgments.
- The problematic nature of 'ratio' judgments.

In previous work, we have attempted to begin formalising these under the rubric of "soundness-in-practice", for example, using the idea of relational homomorphisms holding between the model and empirically based representations of the actual judgments [41].

Some readers may have wondered about the relevance of satisficing in this context. In our view, it is probably a red herring. It can be argued that this idea stems from a failure to appreciate the situation that organisms often find themselves in. That is, it fails to acknowledge the cost of information gathering, as well as the fact that we cannot spend all our time on one judgment, and the corollary, which is that there are many situations where one cannot realistically determine which alternative is better. In fact, this is why we are interested in developing decision aids (and if we were mere satisficers, it would be difficult to explain this interest). The purpose of this paper, then, is not to promote satisficing, but to promote realism, which means using ideas of optimisation in a more thoughtful and appropriate manner.

5.2 Other considerations

Utility 'scaling' is telling us something about the processes involved in psychological scaling. It is an indication that such scaling is always basically inferential in nature, even in the case of so-called sensory scales. For if a human observer can construct a scale without any sensory underpinning, as when supposedly comparing two different rates of increase in some abstract dimension of preference-value, the process must be largely centralised or internal.

In the Introduction, we noted that this paper is based on a study of a large number of papers. Perusal of this literature serves to bring home the fact that in psychology every principle, nay every 'fact', comes with a raft of conditions and caveats. Because of this, popular works on psychology (including the now-popular works on behavioural and neuro-economics) are always misleading, to put it mildly. Here is one example. We all know about the "magic number 7, plus or minus 1"; in fact, Saaty refers to it explicitly in justifying some features of AHP [31]. But how many know that the basic experiments leading to this idea contained an important confounding factor? This is that the amount of information was confounded with the average interval between presentations of a stimulus, an interval during which O was occupied with other judgments. This makes this experimental paradigm a variant of the classical paradigm for testing short-term memory: present a stimulus, followed by mental activities to prevent rehearsal and transfer to long-term memory, and after an interval prompt for recall. So the fabled "single channel" may in this

case be a memory effect [33] For the user of psychological knowledge, then, the message is always “caveat emptor”.

6 NEW ATTITUDES, NEW APPROACHES

Among the implications for preference and utility analysis (within the prescriptive framework) are these:

- (i) It is likely that judgments should be considered as random variables. This is not just a matter of identifying “error”, variability seems inherent in such judgments.
- (ii) Both the stimulus (perceptual) and response aspects of judgments must be considered in evaluating these methods.

More generally, although the argument in this paper is not meant to imply that one should avoid deriving preference orderings or utility scales altogether, it suggests that other approaches should be explored. It also implies that utility cannot be viewed as a measure commensurate with physical measures (which we then go on to ‘optimise’). In the prescriptive context, it should perhaps be treated more as a goal. That is to say, the goal of prescriptive analysis should be a consistent preference ordering (which implies a utility function), but this should not be imposed top-down. That is, it should be a standard for judging the quality of output, especially since a utility function embodies principles of rationality, in particular, transitivity. Putting this another way, we want a system that acts to achieve consistency, but we don’t want to impose this property in the first place. Consistency then becomes something that is maximised rather than something that is enforced.

Currently, we are beginning to develop models and methods commensurate with this approach. In this connection, constraint satisfaction as a means of maximising coherence may be a useful orientation [36]. In addition, we think that the ‘Bayesian style’ of updating prior probabilities could be used in a more thorough-going way than it is presently. In this case, aggregated (constructed) utilities could be replaced by probability-of-satisfaction measures. Unfortunately, space precludes delving into these topics further in the present paper.

Acknowledgments This work received support from Science Foundation Ireland under Grant 05/IN/1886 and Marie Curie Grant MTKD-CT-2006-042563.

REFERENCES

- [1] J. G. Beebe-Center, ‘The law of affective equilibrium’, *Amer. J. Psychol.*, 54–69, (1929).
- [2] D. E. Bell, H. Raiffa, and A. Tversky, ‘Descriptive, normative and prescriptive interactions in decision making’, in *Decision Making. Descriptive, Normative and Prescriptive Interactions*, eds., D. E. Bell, H. Raiffa, and A. Tversky, 9–30, Cambridge, (1988).
- [3] V. Belton, ‘A comparison of the analytic hierarchy process and a simple multi-attribute value function’, *Eur. J. Operat. Res.*, 26, 7–21, (1986).
- [4] M. H. Birnbaum, ‘Controversies in psychological measurement’, in *Social Attitudes and Psychophysical Measurement*, ed., B. Wegener, 401–485, Earlbaum, (1982).
- [5] K. Borchering, T. Eppel, and D. von Winterfeldt, ‘Comparison of weighting judgments in multiattribute utility measurement’, *Management Sci.*, 37, 1603–1619, (1991).
- [6] C. Boutilier, R. I. Brafman, C. Domshlak, H. Hoos, and D. Poole, ‘CP-nets: A tool for representing and reasoning with *ceteris paribus* preference statements’, *J. Artif. Intell. Res.*, 21, 135–191, (2004).
- [7] L. Dias and A. Tsoukiás, ‘On the constructive and other approaches to decision aiding’, in *Proc. 56th Meeting Euro MCDA Working Group*, pp. 13–28, (2004).
- [8] C. Domshlak, R. I. Brafman, and S. E. Shimony, ‘Preference-based configuration of web page content’, in *Proc. Seventeenth International Joint Conference on Artificial Intelligence*, pp. 1451–1456. Morgan Kaufmann, (2001).
- [9] W. Edwards, ‘Use of multiattribute utility measurement for social decision making’, in *Conflicting Objectives in Decisions*, eds., D. E. Bell, R. Keeney, and H. Raiffa, 247–276, Wiley, (1977).
- [10] R. F. Fagot and M. Stewart, ‘Tests of product and additive scaling axioms’, *Perception and Psychophysics*, 117–123, (1969).
- [11] G. Fechner, *Elements of Psychophysics. Volume 1 (orig. 1860)*, Holt, Rinehart & Winston, 1966.
- [12] J. Figueira and B. Roy, ‘Determining the weights of criteria in the ELECTRE type methods with a revised simos’ procedure’, *Eur. J. of Operat. Res.*, 139, 317–326, (2002).
- [13] P. C. Fishburn, ‘Methods of estimating additive utilities’, *Management Sci.*, 13, 435–453, (1967).
- [14] W. R. Garner, ‘Context effects and the validity of loudness scales’, *J. Exper. Psychol.*, 48, 218–224, (1954).
- [15] C. Hardin and M. H. Birnbaum, ‘Malleability of the “ratio” judgments of occupational prestige’, *Amer. J. Psychol.*, 103, 1–20, (1990).
- [16] W. Jesteadt, R. D. Luce, and D. M. Green, ‘Sequential effects in judgments of loudness’, *Journal of Experimental Psychology: Human Perception and Performance*, 3, 92–104, (1977).
- [17] R. L. Keeney and H. Raiffa, *Decisions with Multiple Objectives. Preferences and Value Tradeoffs*, Cambridge, 1993 (orig. 1976).
- [18] D. Lammington, *The Measurement of Sensation*, Oxford, 1997.
- [19] O. I. Larichev and H. Moshkovich, *Verbal Decision Analysis for Unstructured Problems*, Kluwer, 1997.
- [20] O. I. Larichev, D. L. Olson, H. M. Moshkovich, and A. J. Mechitov, ‘Numerical vs cardinal measurements in multiattribute decision making: How exact is enough?’, *Org. Behav. Hum. Decis. Proc.*, 64, 9–21, (1995).
- [21] R. D. Luce and D. M. Green, ‘Ratios of magnitude estimates’, in *Sensation and Measurement*, eds., H. R. Moskowitz, B. Scharf, and J. C. Stevens, 99–111, Reidel, (1974).
- [22] B. A. Mellers and M. H. Birnbaum, ‘Loci of contextual effects in judgment’, *J. Exper. Psychol.: Human Percept. Perform.*, 582–601, (1982).
- [23] B. A. Mellers, D. M. Davis, and M. H. Birnbaum, ‘Weight of evidence supports one operation for “ratios” and “differences” of heaviness’, *J. Exper. Psychol.: Human Percept. Perform.*, 216–230, (1984).
- [24] H. Montgomery, ‘Direct estimation: Effect of methodological factors on scale type’, *Scand. J. Psychol.*, 16, 19–29, (1977).
- [25] F. Mosteller and P. Nogee, ‘An experimental measure of utility’, *J. Polit. Economy*, 59, 371–404, (1951).
- [26] F. N. and M. J. Marcus, ‘The subject effect in judgments of subjective magnitude’, *J. Exper. Psychol.*, 61, 40–44, (1961).
- [27] E. C. Poulton, ‘Quantitative subjective assessments are almost always biased, sometimes completely misleading’, *Brit. J. Psychol.*, 68, 409–425, (1977).
- [28] E. C. Poulton, *Bias in Quantifying Judgments*, Earlbaum, 1989.
- [29] B. Roy, ‘The outranking approach and the foundations of the ELECTRE methods’, *Theory Decis.*, 31, 49–73, (1991).
- [30] B. Roy, ‘Decision science or decision aid science?’, *Eur. J. Oper. Res.*, 66, 184–203, (1993).
- [31] T. L. Saaty, ‘The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making’, in *Multiple Criteria Decision Analysis*, eds., J. Figueira, S. Greco, and M. Ehrgott, chapter 9, 345–407, Springer, (2005).
- [32] B. Schneider and H. Lane, ‘Ratio scales, category scales, and variability in the production of loudness and softness’, *J. Acoust. Soc. Amer.*, 35, 1953–1961, (1963).
- [33] W. Siegel, ‘Memory effects in the method of absolute judgment’, *J. Exper. Psychol.*, 121–131, (1972).
- [34] S. S. Stevens, *Psychophysics*, Transaction, 1975.
- [35] W. P. Tanner and J. A. Swets, ‘A decision-making theory of visual detection’, *Psychol. Rev.*, 61, 401–409, (1954).
- [36] P. Thagard and K. Verbeurgt, ‘Coherence as constraint satisfaction’, *Cognit. Sci.*, 22, 1–24, (1998).
- [37] L. L. Thurstone, *The Measurement of Values*, University of Chicago, 1959.
- [38] W. S. Torgerson, ‘Distances and ratios in psychological scaling’, *Acta Psychol.*, 201–205, (1961).
- [39] A. Tversky and D. Kahneman, ‘Judgment under uncertainty: Heuristics and biases’, *Science*, 185, 1124–1130, (1974).
- [40] P. Vincke, *Multicriteria Decision-Aid*, Wiley, 1992.
- [41] R. J. Wallace and I. Ashikhmin, ‘Soundness conditions for prescriptive decision analysis’, in *Fortieth Annual Meeting of the Decision Sciences Institute*, pp. 4401–4410, (2009).

A Database Approach to Dynamic Group Formation

Florian Wenzel and Werner Kießling¹

Abstract. Modern communication technologies, in particular social network services and mobiles Internet applications, have generated a tremendous hype enabling new forms of human interaction patterns. Millions of users voluntarily give away their preference profiles and want to share common interests in groups. Thus, the proper and efficient management of group preferences becomes a challenging area. Since any approach towards its solution should be scalable to even very large user groups, we propose a database-centered framework in this paper. We extend a well-known constructor-driven approach for modeling preferences as strict partial orders towards modeling and optimization capabilities for dynamic group formation. Since the quality of group formations can be judged by semantic aspects like e.g. group homogeneity, group optimization techniques must be capable to properly take this aspect into consideration. In more detail we describe how to model the quality of group formations depending on individual group member preferences. Then we propose several optimization heuristics for subgroup formation, based on laws of preference algebra and also on statistical knowledge. In summary we are confident that the preliminary results presented here can be extended towards intuitive, robust and scalable algorithms for the dynamic management of even very large user groups with sophisticated preference profiles.

1 Introduction

Modern communication techniques and social networking services in particular lead to a shift from personal contact to virtual interaction. For traditional interaction patterns the concept of homophily states that social network ties and hence groups are based on similarities in common traits such as age, race or religion and also on common interests and beliefs [10]. Studies show that this phenomenon is not restricted to conventional social networks but can also be observed in corresponding virtual counterparts such as MySpace [13]. Consequently, just like groups based on personal contact, virtual groups are defined by common preferences. Features such as creation and administration of groups are the status quo of social networking services. Users can find existing groups via full-text search or create a new group profile. However, these profiles merely provide additional functions besides a forum for group member interaction. The determination of a common group preference based on the interests of its members hence would be a leap in technology for social networking services. Such a group preference can be used to determine a common outcome for the group based on single member preferences. In a fictional social network platform for outdoor enthusiasts, a group representing the local hiking club could determine a suitable tour for the annual hiking trip from a data set of tour candidates based on the preferences of its members.

Introductory Example: Annual hiking trip

In order to determine a common tour for the annual hiking club trip, members Tom, Billy, Sam and Tina state their individual preferences concerning tour attributes. The task now is to find suitable tours based on these stated preferences. This procedure induces the notion of quality of the group result for a single member. Since Billy and Sam are advanced hikers, the creation of subgroups such as a beginner and advanced group becomes an additional issue. $\square$

While the modeling of single user preferences is a widely studied research area, the formation of group preferences as well as the optimization of single user allocation to subgroups based on a preference framework is a new aspect that is addressed in this paper. Several related research areas define problem specifications which are based on groups and consensus decisions. In [1], a database driven method has been introduced to combine single user ratings into a common group recommendation by collecting or estimating single user ratings for a movie and subsequent numerical combination. Within the field of operations research, various contributions approach the problem of group decisions in the context of decision support systems. To find a group consensus, numerical means such as utility functions [11] and methodologies accounting for fuzziness of preferences [7] as well as database-aided solutions [14] have been developed. In the context of multi-agent systems, preference aggregation is often achieved by voting techniques that find consensus winners for a group decision [2]. The topic of interest-matching in this context provides metrics to determine the similarity of preferences between agent pairs [16] which could in turn be used to group agents according to similarity. These approaches are all acting on the assumption that a consensus has to be found for a single group with a predefined set of members. However, even individuals with generally similar interests differ in specific preferences, e.g. individuals might differ in their preference about tour difficulty while sharing a common interest in hiking. Hence, it is more intuitive to create multiple subgroups such as a beginner and an advanced group of an initial group. In addition, groups incorporate a hierarchy that is indispensable for the determination of a group consensus. This fact makes a pairwise comparison of single members based on preference similarity only partly applicable.

These fundamental observations give raise to a framework to group formation that is presented in this paper. It provides a quality measure for group preference results which is vital to control the homogeneity of a group. In cases in which the initial group quality proves to be unsatisfying, heuristics are introduced that optimize the overall group quality by forming higher quality subgroups. To handle large group sizes which are common in application areas such as social network services with millions of users, a database centered approach is pursued. The selection of suitable heuristics follows the tradition of operator tree and query optimization which is an area of expertise of database systems.

¹ University of Augsburg, 86135 Augsburg, Germany,
email: {wenzel, kiessling}@informatik.uni-augsburg.de

The remainder of this paper is organized as follows: Section 2 outlines the technical foundations that are necessary to formally describe individual preferences. Then, explicit ways to combine these single valuations into a common group preference are described followed by a depiction of numeric means to determine the quality of the corresponding group result in Section 3. Based on quality calculations, Section 4 presents heuristics designed to improve group quality through dynamic creation of subgroups. The laws of preference algebra defined by [8, 9] as well as statistical data provided by the database serve as background knowledge for this optimization process. Finally, Section 5 sums up our preliminary results and points out areas of ongoing research.

2 The Preference Model

In databases, a preference $P = (A, <_P)$ is usually expressed as a strict partial order $<_P$ over a set of attributes A [8, 3]. This approach supports quantitative preferences on numerical attributes as well as qualitative preferences on categorical domains. Moreover, it is the key to efficient query processing through algebraic optimization [6, 3] and fast evaluation algorithms such as Hexagon [12] or LESS [5]. Preferences can be constructed inductively, starting from base preferences on single attributes.

Base Preferences

In order to express preferences on single attributes, the preference framework defined in [8, 9] provides a base preference hierarchy for both numerical and non-numerical domains. The following preferences are presented exemplarily:

- $POS(A, pos)$: values contained in the set pos are preferred.
- $NEG(A, neg)$: values not contained in the set neg are preferred.
- $AROUND(A, x)$: numeric values with closest distance to value x are preferred.
- $BETWEEN(A, [low, up])$: numeric values inside the numeric interval $[low, up]$ are preferred. If no such values exist, values with closest distance to the interval are preferred.

All base preferences described in this paper can be considered as subconstructors of the SCORE preference. Given a scoring function $f: dom(A) \rightarrow \mathbb{R}$, P is a SCORE preference on attribute A and is formally expressed by $SCORE(A, f)$, if for $\forall x, y \in dom(A)$: $x <_P y$ iff $f(x) < f(y)$.

Preference Terms

Base preferences $P_1, \dots, P_k$ are further combined to complex preference terms with the help of complex preference constructors which again represent strict partial orders. These include amongst others:

- Pareto $P_1 \otimes P_2$: P_1 and P_2 are equally important.
- Prioritization $P_1 \& P_2$: P_1 is more important than P_2 .

The Pareto constructor values participating preferences equally while in contrast the Prioritization constructor favors the preference provided as first attribute. Only in cases in which two tuples are equally important concerning the first preference, the second preference is evaluated by the Prioritization constructor.

Preference terms describe soft constraints that should preferably be fulfilled in contrast to hard constraints as specified by relational algebra that have to be fulfilled without exception. These constraints are expressed as queries on tuples of database sets that represent a part of the real world. Consequently, preference queries return a

Best-Matches-Only (BMO) result containing perfect matches if such tuples exist or the best matches possible but nothing worse. For this purpose, a preference term P can easily be embedded into relational queries on a relation R by the use of a *winnow* operator [3], which is identical to the *preference selection operator* $\sigma[P](R)$ of [8]. The presented model explicitly allows the expression of multiple base preferences on the same attribute which lays the foundation for complex group preferences.

Group Preferences

The formation of group preferences is performed in three consecutive steps:

- A group member U_i representing a social network service user states preferences on single attributes A with finite domains $dom(A)$.
- Base preferences $P_{1U_i}, \dots, P_{kU_i}$ are combined to a user preference term T_{U_i} using complex preference constructors.
- Preference terms $T_{U_1}, \dots, T_{U_n}$ of n users are combined using aforementioned complex constructors to form a common group preference term T_{group} .

Group Hierarchies

The last step is the accumulation of single user preference terms into a common group preference term T_{group} . This process again uses the introduced complex constructors. Depending on the choice of Pareto or Prioritization in formation of T_{group} , different group hierarchies are induced.

$$T_{group} = T_{U_1} \otimes \dots \otimes T_{U_n} \quad (1)$$

The group preference expressed by Equation 1 defines a group in which all group members are of equal importance. In contrast, Equation 2 describes a hierarchy with one group leader U_1 among otherwise equally valued group members $U_2, \dots, U_n$.

$$T_{group} = T_{U_1} \& (T_{U_2} \otimes \dots \otimes T_{U_n}) \quad (2)$$

While these two group hierarchies are prevalent constellations, virtually all combinations of complex constructors are possible.

3 Quality Measures

Obviously, a group preference can be constructed out of any combination of single user preferences. However, groups vary in quality depending on the homogeneity of single member preferences. This behavior of the BMO set resembles group formation processes in real life situations. If a group varies significantly in individual opinions, it becomes less likely that a common compromise is going to satisfy each single member. Therefore, this finding is consistent with the concept of homophily stated in Section 1. In order to determine the overall quality of a BMO set for a group, measures on three different layers have to be assessed. The first step is to determine the quality of each individual BMO tuple for a single member U_i . Next, these single valuations have to be combined to an overall quality estimate for member U_i regarding the total of tuples of the BMO set. Finally, this overall value of user U_i has to be compared to all other users $U_1, \dots, U_{i-1}, U_{i+1}, \dots, U_n$.

Numeric Representations

In a first step, the quality of a tuple has to be evaluated for a single user. In the context of e-commerce, precedent findings concerning this task have been published by [4]. To extend single user quality values to establish a common group quality, it is beneficial that each base preference can be considered as a subconstructor of the SCORE

preference. This fact provides the necessary transition to a numerical domain through the definition of specific SCORE functions f_P for each base preference constructor P . A function for the POS constructor $POS(A, pos)$ is described in equation 3:

$$f_{POS}(v) = \begin{cases} 1, & \text{if } v \in pos \\ 0, & \text{else} \end{cases} \quad (3)$$

For NEG, the function is defined to return zero if $v \in neg$ and otherwise one. Furthermore, a function for the AROUND constructor $AROUND(A, x)$ is depicted in Equation 4 for infimum inf_A , supremum sup_A of $dom(A)$ and $\Delta d := \max(x - inf_A; sup_A - x)$:

$$f_{AROUND}(v) = \frac{1}{\Delta d}(\Delta d - |x - v|) \quad (4)$$

All functions are normalized to return values within the interval [0,1] with 1 being the best value.

Single Tuple Quality

In order to obtain an overall numeric value for a preference term T_{U_i} on a tuple t of a relation R with attributes $A_1, \dots, A_n$, base preferences $P_1, \dots, P_k$ of T_{U_i} have to be inspected. Since all described base preferences are subconstructors of the SCORE preference, each preference $P_i = (A, <_{P_i})$ can be expressed as $SCORE(A_i, f_{P_i})$. In turn, $f_{P_i}(A_i)$ is the numerical representation of that base preference. Finally, these numerical values are combined using a weighted sum function F :

$$F_{T_{U_i}}(t) := \sum_{i=1}^k w_i \cdot f_{P_i}(t[A_i]) \quad (5)$$

The result provides a close approximation of the value of a tuple t for user U_i depending on T_{U_i} . One remaining task is the exact specification of single weights w_i . Similar to terms in relational algebra, preference terms can be depicted as operator tree with complex constructors as inner nodes and base preferences as leaves.

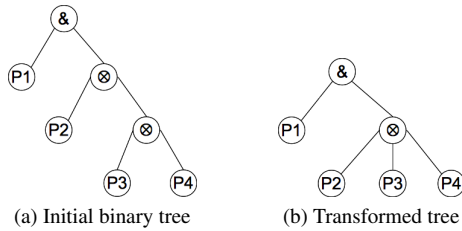


Figure 1: Operator tree transformation

Initially, this tree is binary as shown in Figure 1a but taking the associativity of the Pareto operator into account it can be transformed as depicted in Figure 1b. This transformation is utilized as input for the following Preference Weight Determination (PWD) Algorithm:

Algorithm 1: PWD-Algorithm

input: finite set of operator tree nodes, input weight $w_{init} > 0$, λ
output: $w_1, \dots, w_n > 0$
init: inspectNode(root, w_{init} , 0.99)

INSPECTNODE(NODE, WEIGHT, λ)

```
switch(node.type):
  case Pareto:
    case Pareto:
      for  $i \leftarrow 1$  to node.childCount
        do inspectNode(child $_i$ , weight / childCount);
      case Prioritization:
        inspectNode(leftChild,  $\lambda \cdot$ weight);
        inspectNode(rightChild,  $(1 - \lambda) \cdot$ weight);
      case leaf:
        node.weight := weight;
```

The task of the weights is to mimic the behavior of these complex preference constructors as precisely as possible. The algorithm distributes weight according to a 99 to 1 ratio as specified by λ for Prioritization nodes between the left and the right child node, deeming the left subtree as more important. No λ value can in general achieve the same behavior as a Prioritization [3, 9]. However, by tweaking λ the cases in which the right subtree affects the result even with the left subtree value not being equal can be minimized which is the measure of precision in this case. For Pareto nodes, the weight of the parent node is distributed equally among children. It can be easily verified that the algorithm terminates for finite operator trees with non-zero weights w_i for $w_{init} > 0$.

Example 1: Weight determination

For the operator tree of Figure 1b of preference term $P1 \& (P2 \otimes (P3 \otimes P4))$ the algorithm is initialized with the root node and $w_{init} = 100$. According to the Prioritization rule, the left child node is assigned a weight of 99 and is treated next. The recursive call determines that a leaf is reached and a final weight of $w_1 = 99$ is assigned for $P1$. Next, the right child of the initial Prioritization node is inspected with a weight of 1. According to the Pareto rule, this weight is distributed equally across child nodes. The recursive call for the children again determines that a leaf has been reached and assigns weights $w_2 = 0.33$ for $P2$, $w_3 = 0.33$ for $P3$ and $w_4 = 0.33$ for $P4$. With no more inner nodes left, the algorithm terminates. $\square$

The arbitrary choice of w_{init} calls for a normalization of the initial weighted sum function F to return values in the interval [0,1] with 1 being the desired outcome. Furthermore, a transformation into a quality measure is performed which characterizes the quality deviation (Q) of a tuple with regard to the desired optimum:

$$Q_{U_i}(t) := 1 - \frac{1}{w_{init}} F_{T_{U_i}}(t) \quad (6)$$

BMO Set Quality

Based on the deviation measure for single tuples, the overall quality of a BMO set $\tilde{B}$ provided by T_{group} for a user U_i can be calculated as the average of deviations to single tuples $t_1, \dots, t_j$ of $\tilde{B}$:

$$Q_{U_i}(\tilde{B}) = \frac{1}{j} \sum_{i=1}^j Q_{U_i}(t_i) \quad (7)$$

Equation 7 describes the normalized total deviation of a user U_i to a group BMO set $\tilde{B}$. The lower the quality deviation, the better the BMO set satisfies the user's true wishes. Thus, the presented measure provides an approximation of the quality of a BMO set for a specific user.

Example 2: BMO set quality for a single user

Table 1 displays a group BMO result set determined according to [15]. Further a single user preference for user U_1 Tina is provided:

$$T_{U_1} := POS(difficulty, \{easy\}) \otimes AROUND(distance, 5)$$

Table 1: Sample group BMO set $\tilde{B}$

id	distance	difficulty
1	90	hard
2	5	easy
3	40	medium
4	20	easy
5	70	hard

A numeric value for POS preferences can be calculated via Equation 3, a value for AROUND via Equation 4. The *PWD-Algorithm* returns $w_1 = 50$ and $w_2 = 50$ for $w_{init} = 100$. $Q_{U_1}(t_1)$ for the tuple with *tour id* = 1 is calculated as follows:

$$Q_{U_1}(t_1) = 1 - \frac{1}{100} \cdot (50 \cdot 0 + 50 \cdot \frac{85 - 85}{85}) = 1$$

Accordingly, deviations to tuples with *id* 2, 3, 4 and 5 can be calculated. An overall deviation to $\tilde{B}$ is finally given as:

$$Q_{U_1}(\tilde{B}) = \frac{1}{5} \sum_{i=1}^5 Q_{U_1}(t_i) = 0.54$$

Group Quality

The overall quality of a group BMO set $\tilde{B}$ for the entire group is composed of quality measures of each of the n group members:

$$Q_G(\tilde{B}) = \frac{1}{n} \sum_{i=1}^n Q_{U_i}(\tilde{B}) \quad (8)$$

The resulting quality measure also returns values in the range of [0,1]. The goal for optimization processes therefore is to form groups in such a way that Q_G is close to zero. However, the average of single user quality evaluations can't be the only criterion for the performance of a group. With reference to the concept of homophily, the homogeneity of the group becomes of importance, too. A good average quality can still contain a set of individuals with extremely different quality values for the common group BMO set. Therefore, the variance of quality values also has to be considered. Equation 8 and the variance measure together provide an effective means to calculate the quality of a BMO result set for an entire group.

Example 3: Group BMO set quality and variance

Coming back to the introductory example, preferences of U_2 Tom, U_3 Billy and U_4 Sam are stated additionally to the preferences of U_1 Tina from Example 2, on a database relation *tourinfo*(ID, distance, difficulty, description) which is not shown here but which can be very large in practise. A group preference is formed by a Pareto combination of $T_{U_1}, \dots, T_{U_4}$ as described by Equation 1. The resulting BMO set $\tilde{B}$ is depicted in Table 1.

- $T_{U_2} := NEG(difficulty, \{hard\}) \otimes AROUND(distance, 10)$
- $T_{U_3} := NEG(difficulty, \{easy\}) \otimes AROUND(distance, 90)$
- $T_{U_4} := POS(difficulty, \{hard\}) \otimes AROUND(distance, 70)$

Similar to Example 2, single user deviations are determined:

- $Q_{U_1}(\tilde{B}) = 0.54$
- $Q_{U_2}(\tilde{B}) = 0.42$
- $Q_{U_3}(\tilde{B}) = 0.46$
- $Q_{U_4}(\tilde{B}) = 0.49$
- $Q_G(\tilde{B}) = 0.48$
- $Var(Q_G(\tilde{B})) = 0,002$

□

The subsequent section is explaining how these quality assessments can be used to generate higher quality groups.

4 Dynamic Subgroup Optimization

Acting on the assumption that group heterogeneity leads to poor group quality with regard to a common group preference, the

formation of subgroups leads to more homogeneous groups and defines the subsequent optimization problem.

Optimization Problem: Find partitions $g_1, \dots, g_q$ of the initial group G with minimal $Q_G(\tilde{B}_{g_j}), \forall j \in \{1, \dots, q\}$.

In order to perform this partition, thresholds for group quality have to be defined. Once these are violated, partitioning processes are started. For this purpose, the semantics of preference terms provide valuable information about starting points for group formation. In particular, semantics can be used as background knowledge for heuristics that determine the most efficient subgroup composition.

Laws of Preference Algebra

The preference framework of [8, 9] is both an algebraic means for group term optimization as well as a first indicator for the efficient formation of subgroups by predicting group quality changes under certain conditions. Here we provide additional algebraic laws which are particularly useful for our purposes. In the following, the proof for Lemma 1 is pretty straight forward while the proof for Lemma 2 follows from the *Non-Discrimination Theorem* of [8]. Let P_1, P_2 be preferences on the same attribute A:

Lemma 1

$$P_1 \& P_2 \& P_1 \equiv P_1 \& P_2$$

This elimination rule can be extended to an arbitrary number of preferences. Lemma 1 formally describes the fact, that in the case of Prioritization the addition of similar preferences to an existing group doesn't change the overall group outcome. Considering the aspect of group quality this means that adding similar users to the same group in this case doesn't negatively affect the group quality for a single member. Consequently, users that share the same preferences should be allocated to the same subgroup.

Furthermore, initial group preference terms T_{group} can be investigated to gain starting points for subgroup formation. Contrary preferences on the same attribute by different individuals are such an indicator, finding their formal equivalent in dual preferences. The dual preference P^δ of preference P reverses the order on P , so that $x <_{P^\delta} y$ only holds if $y <_P x$. The intuitive assumption that group members that have dual preferences on the same attribute might be better off being placed in separate groups is backed up by findings in preference algebra for preferences on an attribute A:

$$P \otimes P^\delta = A^{\leftrightarrow} \quad (9)$$

The Pareto combination of dual preferences on the same attribute leads to an Antichain preference $A^{\leftrightarrow}$ for which all attribute values are indifferent [8].

Finally, Lemma 2 defines semantic constructs for which transformations help to clarify the outcome of a corresponding preference term. In part b) of the Lemma, adding another preference P_1 to the pre-existing term $P_1 \& P_2$ doesn't influence the group outcome at all.

Lemma 2

- $(P_1 \& P_2) \otimes P_2 = P_1 \otimes P_2$
- $(P_1 \& P_2) \otimes P_1 = P_1 \& P_2$
- $(P_1 \& P_2) \otimes (P_2 \& P_1) = P_1 \otimes P_2$

Several indicators for the prediction and optimization of group outcomes are provided by underlying preference terms, giving the presented approach an advantage over purely numerical solutions. Based on an initial group preference term T_{group} the presented Lemmas help to identify indicators for subgroup formation and

provide a means to predict group quality changes through the grouping process under certain preconditions.

Algebraic and Statistical Heuristics

Once a violation of quality thresholds is detected, several heuristics can be applied to find a decomposition of the initial group into higher quality subgroups. These heuristics preserve group hierarchies since subgroup formation is achieved by defining a filter preference P_{filter} that is inserted in front of T_{group} via a Prioritization constructor. Thus, the hierarchy is still defined by T_{group} , only the range of possible results is narrowed down by the filter.

While the defined preference algebra of [8, 9] in general and the presented Lemmas as well as Equation 9 in particular provide a rich background for heuristics triggered by semantic indicators, statistical heuristics help in cases where semantics don't provide such information. This holds in particular for the Pareto constructor ' $\otimes$ '. E.g. constructors like $AROUND(A, x)$ are weak orders, but terms like $AROUND(A, x_1) \otimes AROUND(A, x_2)$ are no longer weak order preferences, hence cannot be simplified by an equivalent $AROUND(A, z)$ for some z . Next we propose two heuristics which demonstrate the possibility of implementing both semantic as well as statistical optimization strategies.

▷ **Dual Preference Heuristics:** As already stated before, dual preferences lead to a poor group quality. The presented heuristics attenuates these effects by splitting the group contingent on the value of the critical attribute. Considering a preference P on attribute A , the dual preference P^δ on the same attribute and the group preference T_{group} , two groups are formed according to the following scheme:

- $T_{G1} := P \ \& \ T_{group}$
- $T_{G2} := P^\delta \ \& \ T_{group}$

The two groups defined by preference terms T_{G1} and T_{G2} differ in allowed values for the critical attribute A . Based on the BMO sets $\tilde{B}_{G1}$ and $\tilde{B}_{G2}$ for both subgroups defined by T_{G1} and T_{G2} , users are assigned. A user U_i is accordingly designated for the subgroup with higher single user quality, e.g. if $Q_{U_i}(\tilde{B}_{G1}) < Q_{U_i}(\tilde{B}_{G2})$ then U_i is assigned to the subgroup represented by T_{G1} .

Example 4: Application of the Dual Preferences Heuristics

Since T_{group} of Example 3 contains dual preferences $POS(difficulty, \{easy\})$ and $NEG(difficulty, \{easy\})$ of Tina and Billy the *Dual Preferences Heuristics* can be applied, leading to the following subgroup terms:

- $T_{G1} := POS(difficulty, \{easy\}) \ \& \ T_{group}$
- $T_{G2} := POS(difficulty, \{easy\})^\delta \ \& \ T_{group}$

The two resulting BMO sets $\tilde{B}_{G1} = \sigma[T_{G1}](R)$ and $\tilde{B}_{G2} = \sigma[T_{G2}](R)$ are shown in Figure 2a and 2b.

id	distance	difficulty
2	5	easy
4	20	easy

(a) BMO set for T_{G1}

id	distance	difficulty
1	90	hard
3	40	medium
5	70	hard

(b) BMO set for T_{G2}

Figure 2: Dual Preferences Heuristics: BMO sets for subgroups

With $Q_{U_1}(\tilde{B}_{G1}) = 0.05$, $Q_{U_2}(\tilde{B}_{G1}) = 0.05$, $Q_{U_3}(\tilde{B}_{G1}) = 0.96$ and $Q_{U_4}(\tilde{B}_{G1}) = 0.54$ for subgroup one and $Q_{U_1}(\tilde{B}_{G2}) = 0.86$, $Q_{U_2}(\tilde{B}_{G2}) = 0.67$, $Q_{U_3}(\tilde{B}_{G2}) = 0.13$ and $Q_{U_4}(\tilde{B}_{G2}) = 0.27$ it is obvious that U_1 Tina and U_2 Tom are allocated to the first "beginner"

group and U_3 Billy and U_4 Sam belong to the second "advanced" group. The implementation of the heuristics hence improved group qualities to $Q_G(\tilde{B}_{G1}) = 0.05$ and $Q_G(\tilde{B}_{G2}) = 0.20$. $\square$

▷ **Growing Intervals Heuristics:** Another evidence for inconsistent groups can be gained from AROUND preferences on the same attribute by different group members. A preference $AROUND(A, x)$ determines a preferred numeric value x for an attribute A . Collecting values x from n users leads to a distribution containing values $x_1, \dots, x_n$ from which three possible patterns emerge as shown in Figures 3a through 3c.

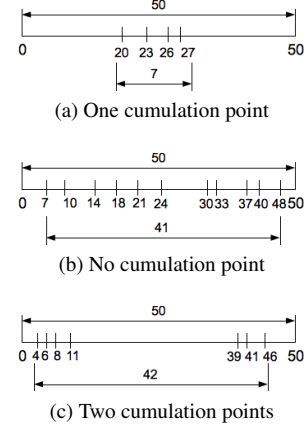


Figure 3: Different distributions of preferred values

Figure 3a depicts a case in which preferred values are cumulated in one delimited area of the domain between the arbitrary infimum 0 and supremum 50, predicting a homogeneous group preference with regard to the examined attribute. A contrary scenario is depicted in Figure 3b where no cumulation point can be identified. The most interesting case for subgroup formation is shown in Figure 3c because it contains multiple cumulation points. Given the definition of sup_A and inf_A for the domain of an attribute A as well as the maximum max_x and minimum min_x of values x_i , these three cases can be determined by Equation 10.

$$q := \frac{max_x - min_x}{(sup_A - inf_A)} \quad (10)$$

To be more precisely, the quotient helps to discriminate between the desired case of Figure 3a in which it is below some threshold of 0.4 and the suboptimal cases of Figure 3b and 3c with $q > 0.4$. To distinguish between the two last mentioned scenarios, the maximum distance d_{adj} between adjacent values x_i and x_j , $i, j \in \{1, \dots, n\}$ has to be calculated. This distance is higher than one quarter of the difference $sup_A - inf_A$ in Figure 3c and lower in Figure 3b.

With means to differentiate between possible distributions of values x_i in place, countermeasures for undesired distributions can be implemented. The case depicted in Figure 3c is ideal for the formation of separate subgroups which in turn only contain a single cumulation point each. The presented heuristics targets exactly this kind of distribution and is designed to form two subgroups. It is called *Growing Intervals Heuristics* because clusters are created in an iterative fashion by defining initial cluster cores as intervals that grow with each iteration step. In a first step, values x_i each form single intervals $[x_1, x_1], \dots, [x_n, x_n]$. With every iteration, these intervals grow by one unit into each direction, e.g. $[x_1 - 1, x_1 + 1], \dots, [x_n - 1, x_n + 1]$. By and by, the clusters overlap and merge until in the end only two clusters are left. This process can also be specified as follows:

- Determination of two adjacent values x_i, x_j with $d_{adj} > \frac{1}{4}(sup_A - inf_A)$.
- Formation of two cluster intervals $[inf_A, \lceil x_i + \frac{1}{8}(sup_A - inf_A) \rceil]$ and $[\lceil x_j - \frac{1}{8}(sup_A - inf_A) \rceil, sup_A]$.
- Specification of the two clusters by determining the minimal and maximal values min_{C_i} and max_{C_i} of cluster $C_i, i \in \{1, 2\}$ and the maximal distance d_{C_i} of arbitrary values of each cluster.

The described specification step leads to two final cluster intervals:

- first cluster: $[max(min_{C_1} - \lceil \frac{d_{C_1}}{2} \rceil, inf_A); max_{C_1} + \lceil \frac{d_{C_1}}{2} \rceil]$
- second cluster: $[min_{C_2} - \lceil \frac{d_{C_2}}{2} \rceil; min(max_{C_2} + \lceil \frac{d_{C_2}}{2} \rceil, sup_A)]$

The presented formal procedure determines the number of needed iterations by taking the maximal distance between values of each cluster into account that has to be overcome. With the two crucial cluster intervals $C_1 = [x_1, y_1]$ and $C_2 = [x_2, y_2]$ defined, two subgroups can be formed according to the following scheme:

- $T_{G1} := BETWEEN(A, [x_1, y_1]) \& T_{group}$
- $T_{G2} := BETWEEN(A, [x_2, y_2]) \& T_{group}$

The allocation of single users U_i to subgroups represented by T_{G1} and T_{G2} is performed by the evaluation of single user quality for BMO sets $\tilde{B}_{G1}$ and $\tilde{B}_{G2}$ as already described for the *Dual Preference Heuristics*.

Example 5: Application of the Growing Intervals Heuristics

Based on previous examples, the intervals $C_1 := [5, 18]$ and $C_2 := [63, 90]$ are determined for $inf_A = 5$ and $sup_A = 90$ which are afterwards refined to $C_1 := [2, 13]$ and $C_2 := [60, 100]$, leading to the following subgroups:

- $T_{G1} := BETWEEN(distance, [2, 13]) \& T_{group}$
- $T_{G2} := BETWEEN(distance, [60, 100]) \& T_{group}$

The corresponding BMO sets $\sigma[T_{G1}](R)$ and $\sigma[T_{G2}](R)$ are depicted in Figures 4a and 4b.

id	distance	difficulty
2	5	easy

(a) BMO set for T_{G1}

id	distance	difficulty
1	90	hard
5	70	hard

(b) BMO set for T_{G2}

Figure 4: Growing Intervals Heuristics: BMO sets for subgroups

Examining single user distances $Q_{U_1}(\tilde{B}_{G1}) = 0$, $Q_{U_2}(\tilde{B}_{G1}) = 0.03$, $Q_{U_3}(\tilde{B}_{G1}) = 1$ and $Q_{U_4}(\tilde{B}_{G1}) = 0.88$ for the first and $Q_{U_1}(\tilde{B}_{G2}) = 0.94$, $Q_{U_2}(\tilde{B}_{G2}) = 0.91$, $Q_{U_3}(\tilde{B}_{G2}) = 0.06$ and $Q_{U_4}(\tilde{B}_{G2}) = 0.06$ for the second subgroup it again becomes obvious that U_1 Tina and U_2 Tom should be assigned to group one and U_3 Billy and U_4 Sam to group two. Accordingly, $Q_G(\tilde{B}_{G1}) = 0.02$ and $Q_G(\tilde{B}_{G2}) = 0.06$. $\square$

5 Summary and Outlook

The presented database driven approach provides a quality measure for group preferences as well as heuristics that dynamically optimize the group outcome by generating subgroups at decision time, thus being sensitive to changes of the underlying data set. Based on the preference algebra of [3, 8] it defines a semantic background for group preferences that is missing in purely numerical approaches. Additionally, complex group hierarchies can be incorporated into preference terms through complex constructors. Known efficient database algorithms permit a very fast calculation of Pareto sets which are an

integral part of group preference terms. As a result, we claim that our group preference formation approach has the potential to display a high performance concerning scalability, which is important in application areas such as social network services with millions of users. Algorithmic strategies that explore very large numbers of possible combinations could also be implemented efficiently.

Our next efforts will include the evaluation of preliminary results as well as intensive benchmarks to gain more experience. For this purpose, both actual data of hiking tours in Germany provided by Alpstein Tourismus GmbH (<http://www.alpstein-tourismus.de>) as well as artificial data will be used. Additionally, other allocation methods based on preference similarity will be evaluated. To confirm predictions for scalability, tests will include the integration of group preferences into a social networking service to create a realistic test environment. Concurrently, algebraic identities in preference terms will be examined to find more semantic triggers for subgroup heuristics.

REFERENCES

- [1] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawla, Gautam Das, and Cong Yu, ‘Group Recommendation: Semantics and Efficiency’, *PVLDB*, **2**(1), 754–765, (2009).
- [2] *Handbook of social choice and welfare*, eds., Kenneth J. Arrow, Amartya K. Sen, and Kotaro Suzumura, Elsevier, Amsterdam, 2002.
- [3] Jan Chomicki, ‘Preference formulas in relational queries’, *ACM Transactions on Database Systems (TODS)*, **28**(4), 427–466, (2003).
- [4] Stefan Fischer, Werner Kießling, and Timotheus Preisinger, ‘Preference based Quality Assessment and Presentation of Query Results’, in *Flexible Databases Supporting Imprecision and Uncertainty*, eds., Gloria Bordogna and Giuseppe Psaila, pp. 91–122, Secaucus, NJ, USA, (2006). Springer-Verlag New York.
- [5] Parke Godfrey, Ryan Shipley, and Jarek Gryz, ‘Maximal vector computation in large data sets’, in *VLDB ’05: Proceedings of the 31st international conference on Very large data bases*, pp. 229–240. VLDB Endowment, (2005).
- [6] Bernd Hafenrichter and Werner Kießling, ‘Optimization of relational preference queries’, in *ADC ’05: Proceedings of the 16th Australasian database conference*, pp. 175–184, Darlinghurst, Australia, (2005). Australian Computer Society, Inc.
- [7] Hsi-Mei Hsu and Chen-Tung Chen, ‘Aggregation of fuzzy opinions under group decision making’, *Fuzzy Sets and Systems*, **79**(3), 279 – 285, (1996).
- [8] Werner Kießling, ‘Foundations of Preferences in Database Systems’, in *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, Hong Kong, China*, pp. 311–322, (2002).
- [9] Werner Kießling, ‘Preference Queries with SV-Semantics’, in *Advances in Data Management 2005, Proceedings of the Eleventh International Conference on Management of Data, Goa, India*, pp. 15–26, (2005).
- [10] Miller McPherson, Lynn Smith-Lovin, and James M Cook, ‘Birds of a Feather: Homophily in Social Networks’, *Annual Review of Sociology*, **27**(1), 415–444, (2001).
- [11] Ojelanki K. Ngwenyama and Noel Bryson, ‘Eliciting and mapping qualitative preferences to numeric rankings in group decision making’, *European Journal of Operational Research*, **116**(3), 487 – 497, (1999).
- [12] Timotheus Preisinger and Werner Kießling, ‘The Hexagon Algorithm for Pareto Preference Queries’, in *Proceedings of the 3rd Multidisciplinary Workshop on Advances in Preference Handling*, (2007).
- [13] Mike Thelwall, ‘Homophily in MySpace’, *Journal of the American Society for Information Science and Technology*, **60**(2), 219–231, (2009).
- [14] Rudolf Vetschera, ‘Integrating databases and preference evaluations in group decision support : A feedback-oriented approach’, *Decision Support Systems*, **7**(1), 67 – 77, (1991).
- [15] Florian Wenzel and Werner Kießling, ‘Group Preferences in Social Network Services’, in *Proceedings of the 22nd Workshop ”Grundlagen von Datenbanken 2010” (GvD-2010), Bad Helmstedt, Germany*, eds., Wolf-Tilo Balke and Christoph Löff, (2010).
- [16] Andrew W. Wicker and Jon Doyle, ‘Interest-Matching Comparisons using CP-nets’, in *Proceedings of the 22nd AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada*, pp. 1914–1915. AAAI Press, (2007).