6th Multidisciplinary Workshop on Advances in Preference Handling (MPREF-12)

Co-located with ECAI 2012 in Montpellier (France)

August 27th

***

Accepted Papers

Markus Endres, Patrick Roocks, Florian Wenzel, Alfons Huhn and Werner Kießling. *Handling of NULL Values in Preference Database Queries.*

Helene Fargier, Jerome Lang, Jerome Mengin and Nicolas Schmidt. *Issue-by-issue voting: an experimental evaluation.*

Thomas Kalinowski, Nina Narodytska, Toby Walsh and Lirong Xia. *Elicitation-free Protocols for Allocating Indivisble Goods.*

Christophe Labreuche, Nicolas Maudet, Vincent Mousseau and Wassila Ouerdane. *Explanation of the robust additive preference model by even swap sequences.*

Alberto Maran, Maria Silvia Pini, Francesca Rossi and Kristen Brent Venable. *Resistance to bribery when aggregating soft constraints, and exploitation of bribery cost schemes in preference compilation and optimization.*

Nina Narodytska and Toby Walsh. *Manipulating Two Stage Voting Rules.*

Davide Navarro, Maria Silvia Pini, Francesca Rossi and Kristen Brent Venable. *Magenda: Doodle with Preferences.*

Sergei Obiedkov. *From Preferences over Objects to Preferences over Concepts.*

Magnus Roos, Jörg Rothe, Joachim Rudolph, Björn Scheuermann and Dietrich Stoyan. *A Statistical Approach to Calibrating the Scores of Biased Reviewers: The Linear vs. the Nonlinear Model.*

Piotr Skowron, Piotr Faliszewski and Arkadii Slinko. *Proportional Representation as Resource Allocation: Approximability Results.*

Tran Cao Son, Enrico Pontelli and Chitta Baral. *A Non-Monotonic Goal Specification Language for Planning with Preferences.*

Paolo Viappiani and Christian Kroer. *Optimization and Elicitation with the Maximin Utility Criterion.*

Wietske Visser, Koen Hindriks and Catholijn Jonker. *Explaining Qualitative Preference Models.*

Nic Wilson. *Balancing Occupants' Comfort in Shared Spaces: A Preliminary Investigation.*

Handling of NULL Values in Preference Database Queries

Endres Markus and **Roocks Patrick** and **Wenzel Florian** and **Huhn Alfons** and **Kießling Werner** [1]

Abstract. In the last decade there has been much interest in preference query processing for various applications like personalized information or decision making systems. Preference queries aim to find only those objects that are most preferred by the user. However, the underlying data set may contain NULL values which represent *unknown* or *incomplete* data. Most of the existing algorithms for preference query evaluation do not know how to treat these NULL values and consider them worse than any other value. Other algorithms do not allow NULLs in their input data set. However, NULL values are common in data sets and must be considered in preference query evaluation. In this paper we introduce an approach to handle NULL values in preference queries which extends preference algebra, a formal model for preference specification. Our approach can be adopted by all preference query algorithms which rely on strict partial orders, because it does not violate the transitivity relation as other methods do.

1 Introduction

Preferences in databases – as shown by a recent survey [18] – as well as preferences in artificial intelligence and social choice theory (cp. [17]) are a well established framework to create personalized information systems. By using well designed preference models, users can be provided with just the information they need, thereby overcoming the dreaded empty result set and flooding effect [10].

However, the data set behind these information systems may contain *unknown* data, known as NULL values in database systems. NULL is a special marker to indicate that a data value does not exist in the database and therefore represents *missing and inapplicable information*. In standard SQL the handling of NULL values has been the focus of controversy for more than 30 years resulting in a three-valued logic [6]. Hence, comparisons with NULL can never result in either *true* or *false*, but always in a third logical result, *unknown*.

However, the discussion of NULLs in preference database queries is an open issue. Almost all algorithms for preference evaluation (e.g., [1, 5, 7, 14, 16]) rely mainly on two assumptions: First, all preference algorithms assume transitivity in the dominance relation, and second, data is complete, i.e., all dimensions are available for all data objects. The first assumption of dominance transitivity is one of the most used properties in preference algorithms. If a data tuple t_1 dominates tuple t_2 while t_2 dominates t_3, then t_1 dominates t_3, too. Using transitivity, preference query processing algorithms exploit various ways of data pruning and indexing. Obviously, the second assumption of completeness is not practical in a real world database, where NULL values frequently occur, cp. [13].

[1] University of Augsburg, Germany, email: {endres, roocks, wenzel, huhn, kiessling}@informatik.uni-augsburg.de

Table 1. Sample table of hiking tours.

tours	**id**	**length**	**difficulty**	**rating**	**vista**
	1	23.5	medium	4	excellent
	2	NULL	easy	5	bad
	3	NULL	NULL	2	bad
	4	13.1	hard	2	good
	5	7.3	NULL	1	excellent

For example, in a hiking tour database (cp. Table 1) it is highly unlikely that all data for all attributes of a tour are always known. The column *length* contains two NULL entries, because it was not possible to determine the length of the tours. Furthermore users may fill a global database with their own hiking tours. If a hiking tourist wants to set the length but doesn't know it or does not want to rate the difficulty of the tour, he omits the input value. Thus the missing data has to interpreted correctly by setting this value to NULL instead of default value, e.g. 0.

If there are NULLs in the database, how should one compare the unknown to the known values in preference queries? For example, if a users' preference is to find hiking tours with a length of 20 km, how are the given values $\{23.5, 13.1, 7.3\}$ compared to NULL?

One may state that NULL should be always worse than all other values. This would be good for a hiking tourist which is a cautious and accurate person and plans all tours in detail. However, for an user who is adventurous, ready to tackle new challenges and who likes to get surprised by new tours, NULL values in the result of the preference query would be a welcome variety. Hence, this user prefers tours with unknown values over fully documented tours.

The same question also arises for Pareto preference (Skyline) queries [1, 10], where two or more preferences are equally important. In a Pareto preference a tuple t_1 is said to dominate a tuple t_2 if t_1 is better than or equal to t_2 in all dimensions and is strictly better than t_2 in at least one dimension. Unfortunately, with the existence of some incomplete dimensions, we cannot simply use the traditional definition of the dominance relation as it is not immediately clear how to compare an incomplete dimension with a corresponding complete dimension. For example, consider the wish for a tour having a difficulty of 'hard' and a rating of 2. We cannot judge which tuple of $t_1 = (\text{hard}, 2)$ and $t_2 = (\text{NULL}, 2)$ is superior in the first dimension.

The aim of this paper is to extend preference queries to cope with the existence of incomplete data. We provide an approach to handle NULL values in preference queries such that the transitivity relation will be preserved and the assumption of data completeness is not necessary for preference evaluation algorithms. Furthermore we suggest a syntax extension for Preference SQL queries [12] to specify the treatment of NULLs in our preference database system.

The rest of this paper is organized as follows: Section 2 highlights related work. An overview of the used preference model is given in Section 3. Section 4 introduces our NULL value handling in preference algebra. Afterwards, we extend the syntax of the Preference SQL query language in Section 5. Finally, we conclude in Section 6.

2 Related Work

Preference queries are more general than the well known Skyline queries introduced more than ten years ago by Börzsönyi et al. [1]. Skyline queries are a special kind of Pareto preference queries and aim to find tuples which are not dominated by others. Since then, several algorithms have been proposed for preference and Skyline query evaluation that include index-based solutions, pre-sorting and no pre-processing, cp. [1, 5, 7, 16] to name a few. Unfortunately, all these algorithms consider only the case of complete data, i.e. data where all values are known. However, NULL values occur frequently in real life data sets.

Several papers have studied the evaluation of Skyline queries over uncertain (probabilistic) data [15]. Uncertain data in those works is generally caused by data randomness, incompleteness, limitations of measuring equipment, etc. Due to the importance of those applications and the rapidly increasing amount of collected and accumulated data containing uncertainty, analyzing large collections of uncertain data has become an important task. However, how to conduct advanced analysis on uncertain data remains an open problem at large [15].

One of the first works on incomplete data and NULL values was done by Chan et al. [2]. They consider a tuple to dominate another tuple only if a subset of a given size of the dimensions dominates the corresponding dimensions in another tuple. Under this definition, the dominance relation becomes non-transitive. Therefore, traditional preference algorithms cannot be applied.

The closest work to ours is the Skyline querying in the presence of incomplete data [9], which is based on the former mentioned paper. In this work for any two *incomplete* tuples only the common dimensions that are known in both tuples are considered. Among these common dimensions only, they apply the traditional dominance relation to decide which tuple dominates the other, if any. However, this fails if there are no common dimensions. Furthermore, Chomicki rightly asks "What is the right logic for defining such preference relations?", cp. [4].

We introduce an approach of NULL value handling which maintains the transitivity of the dominance relation. Therefore every preference algorithm requiring transitivity can be applied to evaluate preferences on incomplete data.

3 Preferences in Database Systems

Preference modeling has been in focus for some time, leading to diverse approaches, e.g. [3, 10, 11]. We follow the preference model from [11] which is a direct mapping to relational algebra and declarative query languages, e.g., Preference SQL which is discussed in Section 5. It is semantically rich, easy to handle and very flexible to represent user preferences which are ubiquitous in our life.

Formally, a *preference* P on a set of attributes A is defined as $P := (A, <_P)$, where $<_P$ is a strict partial order on the domain of $\mathrm{dom}(A) \times \mathrm{dom}(A)$. For $x, y \in \mathrm{dom}(A)$ the term $x <_P y$ is interpreted as "I like y more than x". We say x and y are *indifferent*, if $\neg(x <_P y) \wedge \neg(y <_P x)$, i.e., neither x is better than y nor y is better than x. Note that the preference order $<_P$ is irreflexive and transitive.

The *Best-Matches-Only*-set (BMO-set) of a preference contains all tuples from a data set that are not dominated w.r.t. the preference. Best-Matches-Only offers a cooperative query answering behavior by automatic matchmaking: The BMO query result adapts to the quality of the data in the database, defeating the empty result effect and reducing the flooding effect by filtering out worse results.

To specify a preference, a variety of intuitive base preference constructors together with some complex preference constructors has been defined. Subsequently, we present some selected preference constructors used in this paper. More preference constructors as well as their formal definition can be found in [10, 11, 12].

3.1 Base Preference Constructors

Preferences on single attributes are called *base preferences*. There are base preference constructors for *discrete* (*categorical*) and for *continuous* (*numerical*) domains. Figure 1 shows the taxonomy of several frequently occurring base preferences [12].

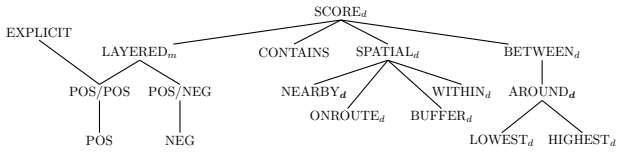


Figure 1. Taxonomy of base preference constructors

Subsequently we describe some numerical base preferences.

Definition 1 (SCORE$_d$ Preference)**.** *Given a scoring function* $f : \mathrm{dom}(A) \to \mathbb{R}_0^+$*, and some* $d \in \mathbb{R}_0^+$*. Then* P *is called a* SCORE$_d$ *preference, iff for* $x, y \in \mathrm{dom}(A)$*:*

$$x <_P y \iff f_d(x) > f_d(y)$$

where $f_d : \mathrm{dom}(A) \to \mathbb{R}_0^+$ *is defined as:*

$$f_d(v) := \begin{cases} f(v) & \text{if } d = 0 \\ \left\lceil \frac{f(v)}{d} \right\rceil & \text{if } d > 0 \end{cases}$$

Note that in the case of $d = 0$ the function $f(v)$ models the *distance* to the best value. That means $f_d(v)$ describes how far the domain value v is away from the optimal value. A d-parameter $d > 0$ represents a discretization of $f(v)$, which is used to group ranges of scores together. The d-parameter maps different function values to a single number. Choosing $d > 0$ effects that attribute values with identical $f_d(v)$ value become indifferent.

The BETWEEN$_d$ preference is a sub-constructor of SCORE$_d$. It expresses the wish for a value between a *lower* and an *upper* bound. A deviation of d does not matter. For BETWEEN$_d(A, [low, up])$ we have $f(v) = \max\{low - v, 0, v - up\}$. Specifying $low = up$ (=: z) in BETWEEN$_d$ we get the AROUND$_d(A, z)$ preference, where the desired value should be z, i.e. $f(v) = |z - v|$. Furthermore, the LOWEST$_d(A)$ and HIGHEST$_d(A)$ constructors prefer the minimum and maximum of the domain of A.

***Example* 1.** The $P_1 := \mathrm{AROUND}_2(\mathrm{rating}, 4)$ preference on Table 1 expresses the wish for a tour rating around 4 where a difference of 2 does not matter. Obviously, the tuple with ID 1 is the most preferred value.

All categorical preferences are sub-constructors of LAYERED$_m$.

Definition 2 (LAYERED$_m$ Preference)**.** *Let $L = (L_1, \cdots, L_m)$ be an ordered list of m sets forming a partition of $\mathrm{dom}(A)$ for an attribute A. The preference P is a* LAYERED$_m(A, (L_1, \ldots, L_m))$ *preference if it is a* SCORE$_d$ *preference with the following scoring function: $f(v) := i - 1 \iff x \in L_i$. For convenience, one of the L_i may be named "*OTHERS*", representing the set $\mathrm{dom}(A)$ without the elements of the other subsets. This implies* OTHERS *contains also* NULL*, if* NULL *is not contained in any other layer.*

Furthermore, sub-constructors of LAYERED$_m$ for frequently occuring cases exist, e.g. POS(A, *POS-set*), which is equal to LAYERED$_2$(A, *POS-set*, OTHERS). It expresses that a user has a set of preferred values, the *POS-set*, in the domain of A. There is also a NEG-preference NEG(A, *NEG-set*). Moreover, it is possible to combine these preferences to POS/POS or POS/NEG. For the POS/POS(A, *POS1-set*, *POS2-set*) preference a desired value *should be* amongst a finite set *POS1-set*. Otherwise it *should be* from a disjoint finite set of *alternatives POS2-set*. If this is also not feasible, better than getting nothing any other value is acceptable. There are many more base preference constructors (cp. Figure 1), all described in [10, 11, 12, 19].

***Example* 2.** Let $P_2 := \mathrm{POS}(\textit{vista}, \{\mathrm{excellent}, \mathrm{good}\})$. That means that we are looking for tours having an excellent or good vista. From Table 1 we get the BMO-set with IDs {1,4,5}.

3.2 Complex Preference Constructors

If one wants to combine several preferences into more *complex preferences*, one has to decide the relative importance of these given preferences. Intuitively, people speak of "this preference is more important to me than that one" or "these preferences are all equally important to me". Equal importance is modeled by the so-called *Pareto preference*.

Definition 3 (Pareto Preference)**.** *In a Pareto preference $P := P_1 \otimes P_2 = (A_1 \times A_2, <_P)$ all preferences $P_i = (A_i, <_{P_i})$ on the attributes A_i are of equal importance, i.e., for two tuples $x = (x_1, x_2)$, $y = (y_1, y_2) \in \mathrm{dom}(A_1) \times \mathrm{dom}(A_2)$ we define:*

$$(x_1, x_2) <_P (y_1, y_2) \textit{ iff}$$
$$(x_1 <_{P_1} y_1 \wedge (x_2 <_{P_2} y_2 \vee x_2 =_P y_2)) \vee$$
$$(x_2 <_{P_2} y_2 \wedge (x_1 <_{P_1} y_1 \vee x_1 =_P y_1))$$

The *Prioritization preference* allows the modeling of combinations of preferences that have different importance.

Definition 4 (Prioritization)**.** *Assume preferences $P_1 = (A_1, <_{P_1})$ and $P_2 = (A_2, <_{P_2})$, then prioritization denoted by $P := P_1 \,\&\, P_2$ is defined as:*

$$(x_1, x_2) <_P (y_1, y_2) \textit{ iff } x_1 <_P y_1 \vee (x_1 =_P y_1 \wedge x_2 <_{P_2} y_2)$$

***Example* 3.** Reconsider the preferences P_1 and P_2 from Example 1 and 2. In the Pareto preference $P := P_1 \otimes P_2$ both preferences are equally important. Tuple 1 dominates tuple 2 and 3, because it is better in both dimensions. Tuple 1 is better than tuple 5 concerning the rating and equal in the vista. Therefore tuple 5 is dominated by tuple 1. Tuple 4 and tuple 2 are indifferent. Tuple 4 is better concerning the rating, but incomparable concerning the vista (*excellent* is not equal to *good*). Therefore, the BMO-set is given by the IDs $\{1, 4\}$.

3.3 Preferences with SV-Semantics

There are situations where indifferent objects should be treated as *substitutable*. That means for base preferences that *all* objects v with equal $f_d(v)$ function value can be designated as *equally good*. This behavior is called *regular Substitutable-Values-Semantics* (SV-semantics). Using regular SV-semantics, all objects with the same $f_d(v)$ value are positioned on the same *level*. Obviously, level 0 contains the perfect matches, higher levels are worse. Having *trivial* SV-semantics only *equal* values are considered as equally good. Following [11] we write $P = C(A, <_P, \cong_P)$ for a preference having any SV relation. We use $\sim_P$ for regular and $=_P$ for trivial SV-semantics.

For base preferences regular SV-semantics does not affect $<_P$ itself, but expresses that it is admissible to substitute values for each other. A complex constructor using $\sim_P$ instead of $=_P$ in its definition (cp. Def. 3 and 4) does affect $<_P$, as we can see in the next example.

***Example* 4.** Consider the Pareto preference $P := P_1 \otimes P_2$ from Example 3. From this example we know that the result of P using trivial SV-semantics is given by the IDs $\{1, 4\}$. Using regular SV-semantics for vista the values *excellent* and *good* are equally good. Since tuple 1 is better than tuple 4 concerning the rating and *excellent* is *substitutable* to *good*, tuple 1 is preferred over tuple 4.

4 NULL Values in Preference Database Queries

In this section we formally introduce the handling of NULL values in preferences. In our proposed approach, NULL is fully integrated in the preference order, i.e. comparisons of NULL and any other value of the domain are possible. To this end we define the NULL-*extended* domain by

$$\mathrm{dom_N}(A) := \mathrm{dom}(A) \cup \{\mathrm{NULL}\}$$

Note that in standard SQL NULLs are not special domain values. A three-valued logic is used, where comparisons with NULL return the third truth value *unknown*. We will use a two-valued boolean logic. An expression $x <_P$ NULL or NULL $<_P x$ with $x \in \mathrm{dom_N}(A)$ is either *true* or *false*. Additionally we require the NULL-extended preference relation $<_p$ to be transitive. Due to these requirements we can use traditional algorithms for the evaluation of preferences.

In the following sections we adapt the preference constructors from Section 3 to the NULL-extended domain. SV-semantics (Section 3.3) are also extended to NULL-values, i.e. the user may specify for which values of x the expression $x \cong_P$ NULL is *true*.

4.1 Insertion Strategies

One possibility to extend preferences to $\mathrm{dom_N}(A)$ is to treat the NULL-value like a value of the original domain, i.e. NULL is *inserted* into the order at the same place as a value from $\mathrm{dom}(A)$. In the case of base preferences, we distinguish between categorical and numerical preferences: For a categorical preference, NULL can be written in the POS-set, OTHERS-set, one of the LAYERED-sets, etc. while for numerical preferences one can either define a NULL-equivalent value (NULL equals 4.5) or place NULL at the top or bottom of the preference order.

Another approach to handle NULL-values is to make the NULL incomparable to all other values, i.e. the expression $x <_P$ NULL is *false* for all x. This models the *missing information* character of the NULL-values: If one knows nothing about a given value, one does not assume any *better than* relations to other values.

Incomparable NULL values are not dominated by any value of $\mathrm{dom}(A)$ and do not dominate any of these values. Hence tuples with NULL-values in the respective attribute always occur in the BMO-set of the preference.

4.2 Extended Categorical Preference Constructors

In the categorical preference constructors, NULL can be used like a usual domain value as shown in the following example:

***Example* 5.** Consider the LAYERED$_m$-preference on attribute A. NULL can be contained in one of the L_i, e.g.

$$\text{LAYERED}_4(\textit{difficulty}, (\{\text{'easy'}\}, \{\text{'medium'}\}, \{\text{'hard'}, \text{NULL}\}, \text{OTHERS}))$$

which means that NULL in the difficulty attribute of the hiking tour is equally disliked as *hard*.

Analogously POS, NEG, POS/POS, etc. are extended in the same manner, i.e. NULL may be written in the POS-set, NEG-setc, etc.

To specify that NULL is incomparable or NULL is placed in the worst layer we introduce a NULL-*handling parameter* for the constructors. Thereby $\text{N}_?$ means NULL *is incomparable to all other values* whereas $\text{N}_{\max}$ places NULL in the worst layer, formally:

Definition 5. *Let C be a preference constructor, A an attribute, X an parameter (Layered-sets, POS-set, etc.) for C and $\cong$ the SV-relation. Then for a preference $P = C(A, X, \cong_P)$ we define:*

1. *Let $P' = C(A, X, \cong_P, \text{N}_?)$, then $<_{P'}$ is given by:*

$$x <_{P'} y = \begin{cases} \text{false} & \text{if } x = \text{NULL} \ \vee \ y = \text{NULL} \\ x <_P y & \text{otherwise} \end{cases}$$

2. *Let $P' = C(A, X, \cong_P, \text{N}_{\max})$. We set the* SCORE*-function (Def. 1) for* NULL *to the maximum of the other values of the domain:*

$$f(\text{NULL}) = \max\{f(v) \mid v \in \mathrm{dom}(A)\}$$

4.3 Extended Numerical Preferences Constructors

For the numerical preference constructors we introduce a constructor which assigns a level or a distance to the NULL-value; additionally NULL may be incomparable, as defined before.

Definition 6. *For a numerical preference constructor C, attribute A, SV-Relation $\cong$ and an optional d-Parameter d and parameter X we define the preference $P = C_d(A, X, \cong_P, N)$, where N may be:*

- $N = \text{N}_?$*: cp. Def. 5, i.e.* NULL *is incomparable to all other values*
- $N = \text{N}_v^{\text{dist}}$*:* NULL *is on distance v.*
- $N = \text{N}_v^{\text{level}}$*:* NULL *is on level v – only if d-Parameter is set with $d > 0$ and regular SV-semantics are used.*

where $v = \max$ means that the $f(\text{NULL})$ is set to the highest level or distance which occurs in $\mathrm{dom}(A)$, cp. Def. 5.

We have the special cases:

- $N = \text{N}_0^{\text{dist}}$*:* NULL *is as good as the best values.*
- $N = \text{N}_\infty^{\text{dist}}$*:* NULL *is the worse than all values of* $\mathrm{dom}(A)$

Thereby *distance* refers to the f-function in Def. 1 whereas *level* refers to the f_d function. In this case we have the equivalences $\text{N}_0^{\text{dist}} \equiv \text{N}_0^{\text{level}}$, $\text{N}_\infty^{\text{dist}} \equiv \text{N}_\infty^{\text{level}}$ and $\text{N}_{\max}^{\text{dist}} \equiv \text{N}_{\max}^{\text{level}}$, where $\equiv$ means that the corresponding preference order is the same.

***Example* 6.** Let $P_3 = \text{AROUND}_{10}(\text{length}, 20, \sim_P, N)$ and consider the tours attribute in the sample data in Table 1. There is no perfect match, i.e. no tour with length 20. Thus for $\text{N}_1^{\text{level}}$ only NULL is in the BMO-set. The length values 23.5 and 13.1 are on level 1 and they are the best matches in $\mathrm{dom}(\textit{length})$, hence for $\text{N}_1^{\text{level}}$ and $\text{N}_?^{\text{level}}$ they are together with NULL in the BMO-set. As the maximum level for P_3 is 2, for $\text{N}_{\max}^{\text{level}}$ and $\text{N}_v^{\text{level}}$ with $v \geq 2$ the NULL value is less preferred than 23.5 and 13.1. In summary we have:

N	BMO-set of values for "length"
$\text{N}_0^{\text{level}}$	$\{\text{NULL}\}$
$\text{N}_1^{\text{level}}, \text{N}_?^{\text{level}}$	$\{\text{NULL}, 23.5, 13.1\}$
$\text{N}_{\max}^{\text{level}}, \text{N}_2^{\text{level}}, \text{N}_3^{\text{level}}, ..., \text{N}_\infty^{\text{level}}$	$\{23.5, 13.1\}$

4.4 Complex Preferences and SV-Semantics

We defined how NULL values are placed in the preference order. Now we consider SV-semantics and complex preferences.

NULL is now a part of the domain and the NULL-extended preferences are still strict partial orders. Therefore the composition of complex preferences can be straight-forward applied to preferences with domain $\mathrm{dom}_{\text{N}}(A)$. For the SV-semantics the same holds: For trivial SV-Semantics $x =_P x$ holds while $x =_P y$ is *false* for $x \neq y$. Note that this implies that NULL $=_P$ NULL is always *true* (in contrast to the trivalent logic in standard SQL). For regular SV-semantics NULL becomes substitutable with all values v having the same level (for $\text{N}_v^{\text{level}}$) or the same distance (for N_v^{dist}). If $N = \text{N}_?$ is used, NULL is not substitutable with any value.

The grouping preference P grouping A evaluates the preference P for all groups with the same value of A separately. It is also extended to NULL values: For P grouping A a group with A = NULL is also considered. To avoid this, P grouping$_{\neg N}$ A is the grouping preference, where a NULL-group is only considered if no other values for A exist.

5 NULL Values in Preference SQL

While previous sections describe a formal framework for NULL handling in preference queries, we now present the extension of the Preference SQL query language. First, we summarize basic features of Preference SQL before describing the extended syntax. Finally, a use case scenario illustrates the applicability of the novel approach.

5.1 Preference SQL

Preference SQL [12] is a declarative extension of standard SQL by strict partial order preferences, behaving like soft constraints under the BMO query model. The BMO-set as result of a preference query contains all database tuples which are not dominated by any other tuple concerning the users preferences, cp. [10]. Syntactically, Preference SQL extends the SELECT statement of SQL by an optional *PREFERRING* clause leading to the following schematic design:

```
SELECT       ...  <selection>
FROM         ...  <table_reference>
WHERE        ...  <hard_conditions>
 PREFERRING  ...  <soft_conditions>
 GROUPING    ...  <attribute_list>
 BUT ONLY    ...  <but_only_condition>
 TOP         ...  <number>
GROUP BY     ...  <attribute_list>
HAVING       ...  <hard_conditions>
ORDER BY     ...  <attribute_list>
LIMIT        ...  <number>
```

The keywords SELECT, FROM, WHERE, GROUP BY, HAVING, and ORDER BY are treated as standard SQL keywords. The *PREFERRING* clause specifies a preference by means of the preference constructors given in Section 3. Furthermore, the Pareto preference can be expressed using the *AND* keyword in the *PREFERRING* clause, *PRIOR TO* expresses a Prioritization. Keywords such as *GROUPING* are provided to modify preference evaluation, *BUT ONLY* for the definition of post-filter or *TOP* and *LIMIT* to regulate the number of results.

A specified preference is evaluated on the result of the hard conditions stated in the WHERE clause. Therefore, preference queries can be cleanly composed with standard SQL queries, even if the standard SQL handling of NULL values uses a three-valued logic in contrast to the two-valued boolean logic used in our preference queries.

***Example* 7.** The preferences $P_1 \otimes P_2$ from Example 4 can be expressed in Preference SQL as follows:

```
SELECT ID FROM tours
PREFERRING length AROUND 4, 2
 AND vista IN ('excellent', 'good');
```

5.2 Extended Preference SQL Syntax

Following the formal framework presented in Section 4, the Preference SQL syntax has to be intuitively extended to allow the expression of newly defined NULL handling possibilities.

For NULL-insertion into the layers of categorical preferences this is straight-forward, as shown in the following example:

***Example* 8.** We translate Example 5 into Preference SQL:

```
... PREFERRING difficulty LAYERED
    (('easy'), ('medium'),
     ('hard', NULL), OTHERS)
```

For the other placements of NULL the syntax

[Attribute] [Constructor] [Parameter] [NULL-handling]

is used. The first three parts of the term are interpreted as usual, say that they are formally $P = C_d(A, X, \cong_P)$. If the optional [NULL-handling] term is set, then a preference $P = C_d(A, X, \cong_P, N)$ is constructed, where N is assigned as follows:

- **`AVOID NULL`**: NULL becomes least preferred, i.e. $N = \mathrm{N}_{\infty}^{\mathrm{dist}}$.
- **`WITH NULL AT BMO`**: NULL is incomparable, i.e. $N = \mathrm{N}_{?}$. Note that an incomparable NULL implies that NULL always occurs in the BMO-set, because incomparable values cannot be dominated by any other value.
- **`WITH NULL AT DISTANCE`** v: NULL is placed at distance v from optimal value, i.e. $N = \mathrm{N}_{v}^{\mathrm{dist}}$.
- **`WITH NULL AT LEVEL`** v: NULL is placed at level v, i.e. $N = \mathrm{N}_{v}^{\mathrm{level}}$.
- **`WITH NULL WORST`**: NULL is placed at the same distance as the worst value of $\mathrm{dom}(A)$, i.e. $N = \mathrm{N}_{\max}^{\mathrm{dist}}$.

If the [NULL-handling] term is omitted, the placement $N = \mathrm{N}_{\max}^{\mathrm{dist}}$ is used, i.e. **`WITH NULL WORST`** is the default NULL-handling.

To avoid the NULL-group in the grouping preference, i.e. to use "P grouping$_{\neg N}$ A", the syntax `[P] GROUPING [A]` **`AVOID NULL`** is used. Then a NULL-group is only considered if no other values for A exist.

5.3 Use Case

Each of the presented NULL handling strategies can be assigned to a user type. Given the database relation in Table 1 we can define four different types of user:

- **experienced user:** Sue is an experienced tour guide, knowing a lot of tours by heart. Hence, she wants to substitute unknown values with concrete values from her experience.
- **indifferent user:** Bob is quite spontaneous and doesn't care about the functionality of database systems. He knows nothing about unknown values and just wants to get best matching tours with no strings attached.
- **cautious user:** Mark is a cautious and accurate person who plans all tours in detail. He prefers tours that give him all the information to make a conscious decision. Thus, unknown values are the last thing that he wants.
- **adventurous user:** Tina is adventurous and ready to tackle new challenges. She likes to get surprised by new tours and to correct missing values with her own hiking records. Hence, she prefers tours with unknown values over fully documented tours.

Given the extended Preference SQL syntax, all these users are now able to express their individual opinions concerning NULL values.

***Example* 9.** Sue as experienced user knows that the average tour length in the desired area is about 35 kilometers and that tours with unknown difficulty level are rarely difficult tours. Since she generally prefers tours with a length around 50 kilometers and a hard difficulty level she poses the following Preference SQL query:

```
SELECT * FROM tours PREFERRING
 length AROUND 50 WITH NULL AT DISTANCE 15
 AND
 difficulty IN ('hard') with NULL AT LEVEL 1;
```

Sue specified an explicit distance value that should be used for comparisons with NULL. Additionally, she placed NULL at level 1 of a POS-preference, thus putting it into the same level as *easy* and *medium*. As result the following tuples are returned from Table 1:

id	length	difficulty	rating	vista
2	NULL	easy	5	bad
3	NULL	NULL	2	bad
4	13.1	hard	2	good

Because NULL is put at distance 15, thus equally preferred as the the value $50 - 15 = 35$ for the length attribute, the tuples with id 2 and 3 are best matches w.r.t. the stated AROUND preference. Furthermore, NULL is in the same level as *easy*, hence both tuples are retrieved. Additionally, the tuple with id 4 best matches the preference for tours of difficulty *hard*. Consequently, tuples with NULL values can be part of the BMO-set in Sue's case.

***Example* 10.** Bob as indifferent user prefers tours with excellent vista and lowest tour length:

```
SELECT * FROM tours PREFERRING
 vista IN ('excellent') AND length LOWEST;
```

Since Bob doesn't know much about NULL values, he posed a query without explicit NULL handling, hence the default behavior is in place. Here, NULL values are treated as being equally preferable to the worst known attribute values, similar to *NULL WORST*. As result the following tuples are returned from Table 1:

id	length	difficulty	rating	vista
5	7.3	NULL	1	excellent

The tuple with id 5 is a best match considering the POS preference and has the lowest length of all tours. For other preferences terms, NULL values might still occur but less frequently compared to Example 9 or 12.

***Example* 11.** Mark as cautious user is looking for tours of difficulty *very easy* and thus poses the following Preference SQL query:

```
SELECT * FROM tours PREFERRING
 difficulty IN ('very easy') AVOID NULL;
```

Mark choses to avoid NULL values in the difficulty attribute if possible, hence NULL is treated as worse than the worst known attribute values. As result the following tuples are returned from Table 1:

id	length	difficulty	rating	vista
1	23.5	medium	4	excellent
2	NULL	easy	5	bad
4	13.1	hard	2	good

Mark didn't get any tuples containing NULL values in the difficulty attribute. Even in the absence of perfect matches for his preference, the best alternatives that are not of value NULL are returned.

***Example* 12.** Tina as adventurous user likes tours with a length between 15 and 20 kilometers with a tolerance of 5 kilometers. With a lower priority she is also interested in a medium difficulty:

```
SELECT * FROM tours PREFERRING
 length BETWEEN 15,20,5 WITH NULL AT BMO
 PRIOR TO
 difficulty IN ('medium') WITH NULL AT BMO;
```

She specifies NULL to be a best match for each base preference. As result the following tuples are returned from Table 1:

id	length	difficulty	rating	vista
1	23.5	medium	4	excellent
3	NULL	NULL	2	bad

Without having the NULL-handling parameter Tina would get the tuple 1. Since she is adventurous the tuple 4 having NULLs in both attributes is also returned. She may decide now.

The presented examples illustrate that the different possibilities of treating NULL values in Preference SQL have a direct impact on the returned BMO-sets. In contrast to hard constraints, none of the presented possibilities for NULL handling can guarantee that no NULL values enter the BMO-set. Even by selecting "AVOID NULL" as handling strategy, complex preference queries might still return a BMO-set containing NULLs, e.g. if a tuple with NULL in one dimension is a perfect match considering another dimension in a Skyline query.

6 Summary and Outlook

In this workshop paper we have addressed the problem of preference database queries over incomplete data, i.e., data having NULL values. We introduced a NULL handling which extends preference algebra and can easily be integrated in preference query languages. We have proposed an insertion strategy for NULL in common preference constructors and extended the syntax of Preference SQL to handle incomplete data. In contrast to other approaches for incomplete data, the transitivity relation among data tuples is preserved, thus all existing techniques for preference or Skyline query evaluation are still applicable. However, we observed that some preference optimization laws [3, 8] – independent of our NULL handling approach – cannot be applied if NULL values exist in a database relation. Although we proposed a model for NULL handling, its benefit must be evaluated in an practical use-case. For this we will extend Preference SQL with our NULL handling behavior and will do some case-studies. Of course, our approach for NULL handling is not restricted to database queries. It can also be adopted by other preference models, e.g., in the wide area of artificial intelligence and social choice theory.

REFERENCES

[1] S. Börzsönyi, D. Kossmann, and K. Stocker, 'The Skyline Operator', in *Proceedings of the 17th ICDE*, pp. 421–430, Washington, DC, USA, (2001). IEEE Computer Society.

[2] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, 'Finding k-Dominant Skylines in High Dimensional Space', in *Proceedings of the 2006 ACM SIGMOD*, pp. 503–514, New York, NY, USA, (2006). ACM.

[3] J. Chomicki, 'Preference Formulas in Relational Queries', in *ACM TODS*, volume 28, pp. 427–466, New York, NY, USA, (2003). ACM Press.

[4] J. Chomicki, 'Logical Foundations of Preference Queries', *IEEE Data Eng. Bull.*, **34**(2), 3–10, (2011).

[5] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, 'Skyline with Presorting', in *Proceedings of the 19th ICDE*, pp. 717–816, (2003).

[6] E. Franconi and S. Tessaris, 'On the Logic of SQL Nulls.', in *Proceedings of the 6th AMW on Foundations of Data Management*, volume 866 of *CEUR Workshop Proceedings*, pp. 114–128. CEUR-WS.org, (2012).

[7] P. Godfrey, R. Shipley, and J. Gryz, 'Maximal Vector Computation in Large Data Sets', in *Proceedings of the 31st VLDB*, pp. 229–240. VLDB Endowment, (2005).

[8] B. Hafenrichter and W. Kießling, 'Optimization of Relational Preference Queries', in *Proceedings of the 16th ADC*, pp. 175–184, Darlinghurst, Australia, (2005). Australian Computer Society, Inc.

[9] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, 'Skyline Query Processing for Incomplete Data', in *Proceedings of the 2008 IEEE 24th ICDE*, pp. 556–565, Washington, DC, USA, (2008). IEEE Computer Society.

[10] W. Kießling, 'Foundations of Preferences in Database Systems', in *Proceedings of the 28th VLDB*, pp. 311–322, Hong Kong, China, (2002). VLDB Endowment.

[11] W. Kießling, 'Preference Queries with SV-Semantics', in *Proceedings of the 11th COMAD*, eds., Jayant R. Haritsa and T. M. Vijayaraman, pp. 15–26, Goa, India, (2005). Computer Society of India.

[12] W. Kießling, M. Endres, and F. Wenzel, 'The Preference SQL System - An Overview', *Bulletin of the Technical Commitee on Data Engineering, IEEE Computer Society*, **34**(2), 11–18, (2011).

[13] W. Kießling, M. Soutschek, A. Huhn, P. Roocks, M. Endres, S. Mandl, F. Wenzel, and A. Zelend, 'Context-Aware Preference Search for Outdoor Activity Platforms', Technical Report 2011-15, University of Augsburg, (2011).

[14] D. Papadias, Y. Tao, G. Fu, and B. Seeger, 'Progressive Skyline Computation in Database Systems', *ACM Trans. Database Syst.*, **30**(1), 41–82, (2005).

[15] J. Pei, B. Jiang, X. Lin, and Y. Yuan, 'Probabilistic Skylines on Uncertain Data', in *VLDB*, pp. 15–26. ACM, (2007).

[16] T. Preisinger and W. Kießling, 'The Hexagon Algorithm for Evaluating Pareto Preference Queries', in *Proceedings of the 3rd MPref*, (2007).

[17] Francesca Rossi, Brent Venable, and Toby Walsh, *A Short Introduction to Preferences: Between Artificial Intelligence and Social Choice*, volume n/a of *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan & Claypool, July 2011.

[18] K. Stefanidis, G. Koutrika, and E. Pitoura, 'A Survey on Representation, Composition and Application of Preferences in Database Systems', *ACM TODS*, **36**(4), (2011).

[19] F. Wenzel, M. Soutschek, and W. Kießling, 'A Preference SQL Approach to Improve Context-Adaptive Location-Based Services for Outdoor Activities', in *Advances in Location-Based Services*, 191–207, Springer Berlin Heidelberg, (2011).

Issue-by-issue voting: an experimental evaluation

Hélène Fargier[1] and **Jérôme Lang**[2] and **Jérôme Mengin**[1] and **Nicolas Schmidt**[1]

Abstract. Multiple referenda consists in making a common decision about each of a set of binary issues, given the preferences of a set of voters. Asking voters for their preferences on all combinations of values is not feasible in practice, because of the exponentially large number of such combinations; on the other hand, voting issue-by-issue on each of the issues can lead to strongly paradoxical outcomes. This paper proposes to measure experimentally to which extent it is suboptimal to vote issue-by-issue voting, in function of the voting rule to be implemented, and the nature of the voters' preferences (arbitrary, separable or additive). For this we use randomly generated separable profiles, which turns out to be a difficult problem.

keywords. Computational social choice; preferences; voting; combinatorial domains; random generation.

1 Introduction

In many practical group decision making contexts, voters have to agree on a value to be given to each of a set of variables, or issues. The simplest approach – which is the one generally used in practice – consists in decomposing the voting process into a set of elementary voting processes, each bearing on a single variable, and processed simultaneously (*i.e.*, in issue-by-issue). When voters have preferential dependencies, however, such a decomposition may (in theory) lead to counterintuitive results [5, 15, 1, 19]. Consider a first example, with 500 voters and two Boolean issues A and B, whose value domains are respectively $\{a, \bar{a}\}$ and $\{b, \bar{b}\}$. The voters' preferences are:

200 voters: $a\bar{b} \succ \bar{a}b \succ \bar{a}\bar{b} \succ ab$
200 voters: $\bar{a}b \succ a\bar{b} \succ \bar{a}\bar{b} \succ ab$
100 voter: $ab \succ a\bar{b} \succ \bar{a}b \succ \bar{a}\bar{b}$

The last group of voters prefer a to $\bar{a}$, whatever the (fixed) value of B, and *vice versa*: their preference order is *separable*. In contrast, the other 400 voters have nonseparable preferences: for example, the first two prefer $A = a$ to $A = \bar{a}$ if $B = \bar{b}$ and prefer $A = \bar{a}$ to $A = a$ if $B = b$. If one asks voters to vote issue-by-issue, and if they majoritarily behave optimistically (*i.e*, the first group majoritarily chooses $A = a$), then we get a majority for $A = a$ and a majority for $B = b$: the final decision (a, b) is the worst for 80% of the voters.

As remarked by Lacy and Niou [15], issue-by-issue voting is much less of a problem when voters' preferences are separable[3]. However, separability is a highly demanding condition; moreover, it does not allow to avoid all paradoxes, even in its strongest version, as shown on the following example, that issue-by-issue voting does not satisfy *Pareto efficiency*[1, 17, 19].

Example 1. *We have three Boolean issues and three voters whose preferences are:*

v_1: $ab\bar{c} \succ a\bar{b}\bar{c} \succ \bar{a}b\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ abc \succ a\bar{b}c \succ \bar{a}bc \succ \bar{a}\bar{b}c$
v_2: $a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}c \succ \bar{a}\bar{b}\bar{c} \succ abc \succ \bar{a}bc \succ ab\bar{c} \succ \bar{a}b\bar{c}$
v_3: $\bar{a}bc \succ \bar{a}\bar{b}c \succ \bar{a}b\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ abc \succ ab\bar{c} \succ a\bar{b}c \succ a\bar{b}\bar{c}$

Issue-by-issue voting leads to the decision abc, which is Pareto-dominated by $\bar{a}\bar{b}\bar{c}$, that is to say all voters prefer $\bar{a}\bar{b}\bar{c}$ to abc.

These two problems raise concerns on the social acceptability of issue-by-issue voting. On the other hand, voting on combinations (or "bundles") of values, which may be the only way to escape them, is practically impossible to implement, because of the combinatorial nature of the problem. As a matter of fact, issue-by-issue voting, as imperfect as it may be, is used in quite many contexts, and in particular in multiple referenda, as they are held for example in California [5]. Other solutions have been suggested, such as sequential voting [16, 18, 10], using compact representation languages such as in [7]; each of them shows to have some benefits but also some pitfalls.

In this paper we stick to issue-by-issue voting. We would like to know to which extent this procedure can approximate a voting rule on a combinatorial domain, comparing the outcome of this procedure to the one that we would have got if preferences over bundles had been elicited and aggregated using a given voting rule. Some rather negative results have been given in [8], who consider a few rules based on scores, and for each of them, give worst-case approximation bounds of the ratio between the score of the alternative chosen by issue-by-issue voting and the score of the alternative chosen by the voting rule, for separable profiles. However, these worst-case negative results do not give much information about the *average-case* of issue-by-issue voting. Here we address this question experimentally, via a random generation of profiles; however, generating separable profiles appear to be a difficult problem, because the ratio between the number of separable preferences and the number of arbitrary preferences is very low [11]; we address the problem by several random generation methods in Section 3. In Section 4 we use these methods to assess the average quality of issue-by-issue voting.

2 Background

2.1 Voting

Let $\mathcal{X}$ be a finite set of m alternatives. A *vote* over $\mathcal{X}$ is a linear order $\succ$ over $\mathcal{X}$. A *profile* $P = (\succ_1, \ldots, \succ_n)$ is a collection of n votes over $\mathcal{X}$, where $\succ_i$ is the vote of voter i. The vote of i represents her preferences, assuming that votes are sincere. A *voting rule* is a function r that associates to each profile P an alternative $r(P) \in \mathcal{X}$.

Several classical voting rules have been extensively studied (see e.g. [4] for a panorama of voting rules). In this paper, we are mainly interested in two groups of voting rules. The first group is that of scoring voting rules, that associate a score with each alternative, based on the ranks of the alternative in the votes. More precisely, given a vote $\succ$ and an alternative $x \in \mathcal{X}$, let $rk(\succ, x) \in \{1, \ldots, m\}$

[1] IRIT – University of Toulouse; {lastname}@irit.fr
[2] LAMSADE – University of Paris-Dauphine; lang@irit.fr
[3] Three forms of separability will be defined in Section 2; here it is enough to say that the argument holds for each form.

denote the rank of x in $\succ$. A scoring voting rule is defined by a vector of scores $\langle s_1, \ldots, s_m \rangle$, such that $s_1 \geq \ldots \geq s_m$, so that s_i is the score associated with rank i. Every time an alternative x is ranked ith for some voter, this vote contributes s_i to the overall score of x. Given a profile $P = \langle \succ_1, \ldots, \succ_n \rangle$, the score of x for P is therefore $s(P, x) = \Sigma_{i=1}^{m} s_{rk(\succ_i, x)}$. The alternatives are then ranked according to their global score $s(P, x)$, and $r(P)$ is the alternative x that maximizes $s(P, x)$. Three distinguished scoring rules that will be considered in this paper are:

Borda: $s_1 = m - 1, s_2 = m - 2, \ldots, s_m = 0$;
Plurality: $s_1 = 1, s_2 = \ldots = s_m = 0$;
$\frac{m}{2}$**-approval:** $s_1 = s_2 = \ldots = s_{\frac{m}{2}} = 1, s_{\frac{m}{2}+1} = \ldots = s_m = 0$

Given a profile P, x is a *Condorcet winner* if for every $y \neq x$, a majority of voters rank x ahead of y. A voting rule r is said to be *Condorcet-consistent* if, for every profile P for which there is Condorcet winner x, $r(P) = x$. It is well-known that no scoring rule is Condorcet-consistent. The second group of rules that we study in the sequel contains Condorcet-consistent rules. For most of these rules, the winner can be determined from the *majority graph* associated with profile P: this graph contains an oriented edge from alternative x to y if a majority of voters ranks x ahead of y. More generally, a weighted majority graph associated with P indicates, for every pair of alternatives (x, y), the number of voters $N_P(x, y)$ that rank x ahead of y. In particular, we will consider two Condorcet-consistent voting rules in the sequel:

Copeland: the alternative that wins the most duels wins;
Maximin: the alternative x that maximizes $\min_{y \neq x} N_P(x, y)$ wins.

2.2 Combinatorial domains

We consider a set $\mathcal{I} = \{A, B, C, ...\}$ of p *issues*, each issue being associated with a binary domain (the possible answers): $D(A) = D(B) = D(C) = \cdots = \{0, 1\}$. Then $\mathcal{X} = D(A) \times D(B) \times D(C) \times \ldots$ is the set of the possible alternatives, or, using voting terminology, *candidates*. The number of alternatives is thus $m = 2^p$. The elements of $\mathcal{X}$ are vectors $\vec{x}$, $\vec{x'}$; we will often concatenate the answers to describe a particular alternative. For instance, if $\mathcal{I} = \{A, B, C\}$, $(1, 0, 1)$ denotes the alternative that has answer 1 for issue A, answer 0 for issue B, and answer 1 for issue C. We will also use concatenation for vectors of answers for disjoint sequences of issues: for instance, if $\mathcal{I} = \{A, B, C, D\}$, $Y = (A, B)$, $Z = (C, D)$, $\vec{y} = (1, 0)$, $\vec{z} = (0, 1)$, then $\vec{y}.\vec{z}$ denotes the alternative $(1, 0, 0, 1)$. Lastly, for every $X \subseteq \mathcal{I}$, D_X is the set of assignments $\vec{x}$ of elements of X in their respective domains.

When considering orderings over combinatorial domains, there exist three definitions of separability.

Let $\succ$ be a vote over $\mathcal{X}$, it is:

weakly separable if for every variable $A \in \mathcal{I}$, every $v, v' \in D_A$, $\vec{x}, \vec{x'} \in D_{\mathcal{I} \setminus \{A\}}$: $v.\vec{x} \succ v'.\vec{x} \iff v.\vec{x'} \succ v'.\vec{x'}$

strongly separable if for every partition $\{X, Y\}$ of $\mathcal{I}$ and every $\vec{x}, \vec{x'} \in D_{\mathcal{X}}$, $\vec{y}, \vec{y'} \in D_{\mathcal{Y}}$: $\vec{x}.\vec{y} \succ \vec{x'}.\vec{y} \iff \vec{x}.\vec{y'} \succ \vec{x'}.\vec{y'}$

additively separable if for every issue $X_i \in \mathcal{I}$ there exists a function $u_i : D_i \to \mathbb{R}^+$ such that for every $\vec{x}, \vec{y} \in \mathcal{X}$ we have $\vec{x} \succ \vec{y}$ if and only if $\sum_i u_i(x_i) > \sum_i u_i(y_i)$.

Example 1. (cont.) *The voters on example 1 have strongly separable preferences. For instance, we have* $a \succ_1 \bar{a}, \forall B, C$.

Strong separability is sometimes called mutual preferential independence, as in [13]. It requires that preferences over combinations of values of any subset of variables do not depend on the values of other variables. Weak separability only requires that preferences over values of a single variable do not depend on the fixed values of other variables; it is met by preference relations associated with a CP-net with no edge in the dependency graph [2]. Additive separability implies strong separability, which in turn implies weak separability. As soon as $p \geq 5$, additive separability is strictly stronger than strong separability [14], whereas both notions coincide for $p \leq 4$ [3]. Therefore, we have only one notion of separability for $p = 2$, two distinct notions for $p = 3$ and $p = 4$. On continuous domains, strong and additive separability are equivalent [9].

2.3 Problematics

Our main objective is to compare the results of the strict application of a voting rule over a combinatorial domain with the results obtained when issue-by-issue voting is used. It is known that as soon as $p \geq 3$, issue-by-issue voting does not satisfy neutrality nor efficiency [1, 19], which implies that any voting rule on a combinatorial domain with more than two variables that satisfies any of these two properties (and all commonly used voting rules do) does not coincide with the issue-by-issue voting rule. For instance, in Example 1, the issue-by-issue winner is abc, whereas the Plurality cowinners are $ab\bar{c}$, $a\bar{b}c$ and $\bar{a}bc$, and the Borda winner is $\bar{a}\bar{b}\bar{c}$, which is also a Condorcet winner.

However, this impossibility theorem does not exclude the possibility that the outcomes do coincide *in general*. The rest of the paper addresses this question by trying to determine the probability that the outcomes coincide. Because of the difficulty of an analytical approach, we estimate this probability experimentally. Ideally it would be interesting to conduct these experiments with real-world data, but such voting data are rarely available, and, to our knowledge, there exist no available data of significant size for multiple referenda (in the data on multiple referenda used in [5], only the combination of the preferred values of each voter is given, not their entire preference relations). Therefore we choose to generate random samples.

3 Generation of separable profiles

The generation of random profiles in social choice is not a new problem: the *impartial culture assumption*, which assumes a uniform distribution over the set of possible profiles, is often made. In our case, we have $m = 2^p$ alternatives, thus there are $2^p!$ possible votes for each voter. Generating uniformly distributed profiles over such domains is not a problem as long as the number p of issues is low. However, the probability of obtaining a weakly separable (or, *a fortiori*, a strongly or additively separable) profile among the $2^p!$ possibilities is extremely weak. For instance, when $p = 4$, of the 16! possible orderings only 5376 are strongly (and additively) separable (a ratio of around $1/10^8$), and 26886144 are weakly separable (a ratio of around $1/10^6$). The exact numbers of orderings that are weakly (resp. additively, resp strongly) separable for $p > 5$ (resp. $p > 6$, resp. $p > 7$) are currently unknown – to our knowledge.

We have investigated several ways of generating random separable orders. A first, naive method, consists in picking random orders uniformly distributed and keeping only the (weakly/strongly/addtively) separable ones. Given the very low probability of picking a separable order, this method is practically not feasible as soon as $p > 4$.

Generating additively separable orders by random utility generation

When the value of p becomes too large for an explicit enumeration, we shall rely on multiattribute utility theory as a way of gen-

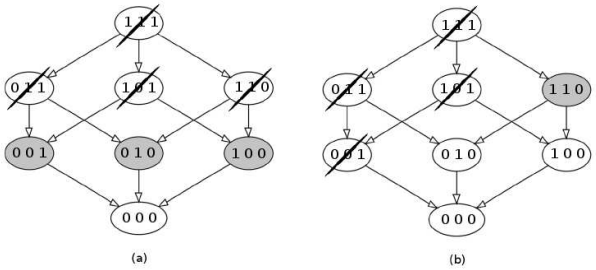


Figure 1. The lattice of options for $p = 3$, after the first four options have been chosen: (a) three possible choices, (b): one possible choice.

erating additively separable orders. The method consists in doing the following: for each variable X_i, we generate a random utility $u_i(x_i) \in (0, 1]$ for each of the possible values $x_i \in D(X_i)$; this results in a utility function on alternatives $u : \mathcal{X} \to \mathbf{R}$, and then we rank alternatives according to u. This method is a simplified version of the one developed in [6]. The distribution we get reaches every additively separable order with a positive probability, but is not uniform. It nevertheless has the advantage of being based on a well characterized model of rational decision makers.

Storing all normalized orders (weak and strong separability)

An ordering is normalized [11] if (i) the best alternative is $(1, ..., 1)$ and (ii) $(0, 1, 1, ..., 1) \succ (1, 0, 1, ..., 1) \succ \cdots \succ (1, 1, 1, ..., 0)$. Every separable order can be obtained from some normalized, separable order by permutation of some issues and inversion of some answers. Thus, in order to generate separable orders, we can first build a table of all normalized, separable orders, and then pick some of them at random and apply some permutations and inversions at random.

This method can be used for generating weakly separable orders if there are no more than $p = 4$ issues: for $p = 5$ issues our first experiments show that one would need a table of at least 5 terabytes (probably much more). For strongly separable orders, this method works well in practice for up to $p = 6$ issues

Generating weakly separable orders by lattice exploration

Normalized, weakly separable orders can be directly generated by considering the partial order $\succ^0$ that contains all pairs of the form $(1.\vec{x}, 0.\vec{x})$ for every $\vec{x} \in D_{\mathcal{I}\setminus\{A\}}$, for every $A \in \mathcal{I}$. All completions of $\succ^0$ are then normalized, weakly separable orders (and random permutations of issues and random inversions of answers will generate any weakly separable order). Such completions can be generated by random, top-down traversal of the graph that corresponds to $\succ^0$. The traversal must be such that no alternative is reached before preferred alternatives. The order in which the alternatives is reached is then a weakly separable order. More precisely, the first alternative $\vec{x}_1$ is the one that has the preferred value for every issue, $11 \ldots 1$ if we build a normalized order. We can then pick at random the second preferred alternative $\vec{x}_2$ among those that are dominated by $\vec{x}_1$ only. The third preferred alternative is then picked at random among those that are only dominated by $\vec{x}_1$ and $\vec{x}_2$, and so on.

However, a uniform distribution over all possible alternatives every time a new alternative must be picked does not guarantee that the resulting distribution over complete, normalized, weakly separable orders will be uniform. This is due to the fact that the number of possible alternatives at a given step depends on the already chosen alternatives. Figure 3 depicts two different possibilities to pick the first four alternatives when there are $p = 3$ issues: it shows that the number of possible alternatives at this step is not constant. The probabilities of the most frequent and of the least frequent complete orders generated in this way can be calculated. For $p = 4$ issues, they are $1/331776$ and $1/238878720$ respectively.

A better idea to complete the partial order $\succ^0$ is to proceed level by level, top-down. After all the alternatives above a given level in the lattice have been ranked, we consider all alternatives $\vec{x}$ at the next level: we determine the possible ranks for each of them in the current order, so as to respect the weak separability property: the rank of $\vec{x}$ will then be picked randomly among these possible ranks. The alternative $\vec{x}$ which has the lowest number of possible ranks is ranked first – since ranking the other alternatives will not change its set of possible ranks, whereas ranking an alternative highly constrained may diminish the number of possible ranks for alternatives less constrained. Once all alternatives at this level have been ranked, we proceed to the next level of the lattice of $\succ^0$.

For instance, consider the following order obtained after adding the two first levels; $111 \succ 011 \succ 101 \succ 110$. At the next level, the least restricted alternative is 001, since there are two possible choices for it (and only one for 100 and for 010). If we insert first 001, it will have a probability one half to be in the fourth position, and one half to be between the positions 5 and 7 (since 100 and 010 will be inserted afterwards). If 001 is inserted last, there is one $\frac{1}{4}$ to have it finally ranked at each of its possible locations (4, 5, 6 and 7).

Generating weakly separable orders by reparation of non-separable orders

Another method to generate weakly separable preferences consists in randomly generating (not necessarily separable) orders $\vec{x}_1 \succ \vec{x}_2 \succ \ldots \succ \vec{x}_m$ over $\mathcal{X}$ with a uniform distribution, and "reparing" each such order so as to make it weakly separable. The reparation works issue by issue, as follows: for every issue $A \in \mathcal{I}$, we first record the value that $\vec{x}_1$ has for A as being the preferred value for A; then we scan the entire ordering, making permutations every time we encounter pairs of alternatives that violate the weak separability condition with respect to A and the chosen preferred value. It can be proved that the resulting order, when it has been repaired w.r.t. all issues, is weakly separable. The order in which the issues are repaired has to be generated at random, since it has an effect on the resulting order over the alternatives.

However, using this algorithm to repair randomly, uniformly generated orders does not give a uniform distribution over weakly separable orders: this reparation procedure can be seen as a local search in the neighbourhood of the initial order; so if a weakly separable order has more not separable neighbours than another one, it has more chances of being obtained.

Evaluating the random generators

We thus get the utility-based generator (for that seems natural for additively separable preferences), the storing generator (for weakly separable preferences up to $p = 4$ and strongly separable preferences up to $p = 6$), and the reparation- and exploration- based generators, that have been designed to generate weakly separable preference for higher values of p in a way close to equiprobability.

In order to compare the quality of the random generators described above, we have tested them in the case of $p = 4$. We know in

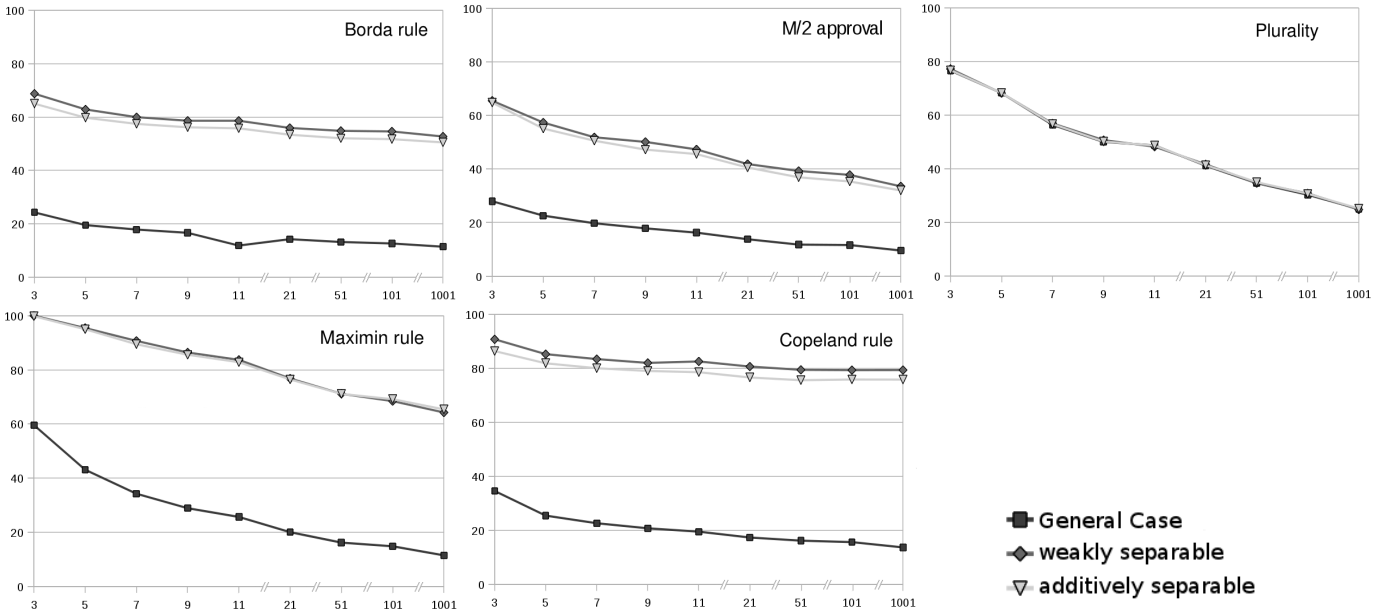


Figure 2. Success rate of issue-by-issue voting w.r.t. the number of voters; 4 issues

this case that there are 70016 normalized weakly separable orders (and therefore $70016 \times 2^4 \times 4! = 26886144$ weakly separable orders), and 14 strongly separable normalized orders (and therefore $14 \times 2^4 \times 4! = 5376$ strongly separable orders). Each generator was run at least 10^7 times so as to generate a normalized order, and we counted apparition of each preference order. The following table gives the frequencies of occurrence of the least frequent and most frequent orders, the ratio between these two frequencies, as well as an estimation as the entropy of each generator — the entropy measures the closeness of a probability distribution to equiprobability, and varies from 1 (equiprobability) to 0 (determinism).

	Storing	Utility	Reparation	Exploration
Weak sep.	X		X	X
Strong sep.	X			
additive sep.		X		
Max. freq.		4/25	1/3900	1/44050
Min. freq.		1/25	1/1400000	1/133333
$\frac{\text{max. freq.}}{\text{min. freq.}}$	1	4	360	3
Entropy	1	0,92	0,97	0,9994

Table 1. Entropy, minimal and maximal frequencies of apparition of the generated orders, by generator

Generating by storing is the only perfectly equiprobable generator. It can be used when the number of issues remains low: up to 4 for weakly separable orders, and up to 6 for strongly separable orders.

As we could expect, the utility-based generator has a bad entropy in the case of 4 issues. Nevertheless, it has the advantage of being simple, fast, and based on a realistic voter preference model, therefore this is the one we shall use to generate additively separable preferences for more than 6 issues. We do not currently know how to efficiently do this for more than 6 issues; therefore, this generator will also be used for strong separability, although it gives a zero probability to any strongly separable order which is not additively separable.

As to generating weakly separable preferences, our experiments suggest that the generator by lattice exploration is better than the reparation-based generator; we will use this argument to choose this generator for our experiments reported in Section 4.

4 Experimental study

The aim of the following experiments is to evaluate the interest of issue-by-issue voting for multiple referenda as an approximation of the application of a specific voting rule, applied to a profile over a combinatorial domain. We consider five voting rules: Borda, $\frac{m}{2}$-approval, Plurality, Maximin and Copeland, and we study the influence, on the quality of the approximation, of parameters such as the number of issues, the number of voters and the type of preference (weakly separable, strongly separable, additively separable).

Since the issues are binary, issue-by-issue voting leads to applying majority voting on each issue. When the profiles are not separable, we suppose that the voters adopt an optimistic attitude and prefer the values as prescribed by their preferred alternative. In order to limit the occurrence of ties, we assume the number of voters to be odd. The outcome of the issue-by-issue voting is then compared to the alternative chosen by the application of each specific voting rule r.

As for the generation of general profiles, without any assumption of separability, we use a uniform distribution over all profiles. For the generation of weakly/strongly/additively separable profiles, we use the storing-based generator for $p \leq 4$. When $p > 4$, we use the exploration-based generator and the utility based generator.

In the first experiment (Figures 2 to 5), we count the percentage of profiles that lead to the success of issue-by-issue voting (that is, the proportion of the generated profiles for which the original rule and issue-by-issue voting elect the same alternative; in case the application of the original voting rule gives a tie, we consider that issue-by-issue succeeds as soon as it elects one of the tied winners). Each of the following experiments repeats the test 10000 times (each point in the curve is computed on 10000 profiles).

4.1 Influence of separability

Unsurprisingly, the experimental results (cf. Figure 2; notice that for $p = 4$, strong and additive separability are equivalent, hence the figures draw only one curve for both concepts) are consistent with the theoretical results of [15] and [12]: issue-by-issue is sounder on separable profiles. For all the voting rules considered but Plurality, the success rate is better on separable samples than on the purely random samples. The result keeps holding when the number of voters increases (Figure 2) and when the number of issues increases (see Figure 3 for Borda; for the other rules studied, except Plurality, we get a similar behaviour). Notice that the success rate seems slightly better for weakly separable profiles than for strongly separable ones; we do not have a clear interpretation of this fact. For Plurality, the same results are obtained, whether or not separability is assumed. This can be easily explained by the fact that Plurality, like issue-by-issue voting, is tops-only, *i.e.*, the outcome is determined from the top of the votes, which implies that separability has no influence.

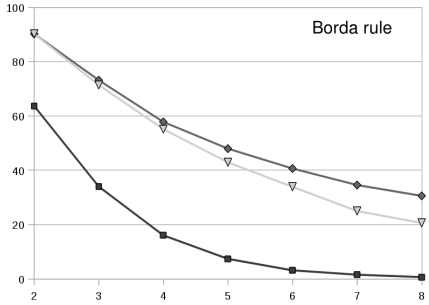


Figure 3. Borda's rule: success rate of issue-by-issue voting w.r.t. the number of issues; 11 voters.

From now on, we conduct the experiments on separable samples only: first, because purely random preferences lead to a quite bad success rate; and second, because a voter with non separable preferences can hardly give her preference issue by issue; we made the assumption that they report votes optimistically, which seems to be observed in practice, but this assumption can be questioned.

4.2 Comparison of voting rules

Let us first look at the case $p = 4$ (Figure 2; Figure 4 summarizes the 5 rules on the weakly separable sample).

We first observe that the three scoring rules are badly approximated; as soon a the number of voters reaches 7, the success rate of the approximation goes below the 60%, that is, for at least 6 cases among 10, issue-by-issue voting elects a winner that is different than the one designated by the application of the original voting rule. This rate is especially bad for Plurality and for $m/2$-approval.

Some rules, like plurality, $m/2$-approval or maximin generate many ties when the number of voters is low; this boosts the success rate of these rules for a few voters samples. That's why, for example, the success rate of maximin is 100% with 3 voters but decreases quickly when the number of voters increase.

It can moreover be noticed that the success rate gets worse as the number of voters increase. In a second series of tests, we measure the success rate for samples of 7 voters, letting the number of issues increase from 2 to 10, for the rules that were not too badly approximated according to the first experiment, namely Borda, Copeland and Maximin. Figure 5 reports our results for weakly separable profiles, generated by exploration (similar ones have been obtained for separable profiles and additively separable profiles): the success rate clearly depends on the number of issues (the more issues, the worst).

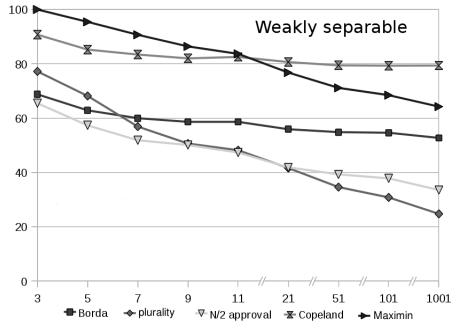


Figure 4. Success rate for Borda, Plurality, $m/2$-approval, Copeland, Maximin w.r.t. number of voters; weakly separable profiles, 4 issues

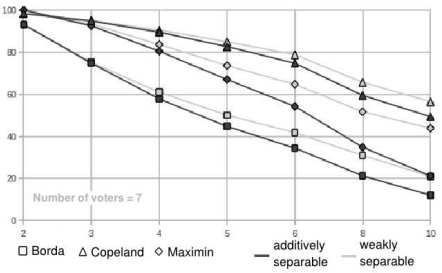


Figure 5. Success rate for Borda, Copeland, Maximin w.r.t. the number of issues, 7 voters

In summary, the success rate of issue-by-issue voting thus gets worse as the number of voters increases and, to a larger extent, as the number of issues increases. It quickly becomes very bad for Plurality and $\frac{m}{2}$-approval (once again). Results are better for Borda, but nevertheless falls below 50% for 5 issues (for 7 voters) or 10 voters (for 4 issues), which is disappointing. Finally, it is much better for Copeland and (to a lesser extent) Maximin; in both cases, a closer look to the sample reveals that bad results are highly correlated with the absence of a Condorcet winner.

4.3 Quality of the approximation

For 5 issues, the success rate of issue-by-issue falls below 50% for all considered scoring rules, and below 80% for Copeland and Maximin.However, the probability that both winners coincide is perhaps not the best way of measuring the approximation of a voting rule; as in [8], we may consider instead, for any voting rule r based on the maximization of some numerical score, the ratio between the score of the alternative elected by the issue-by-issue rule and the score of the winner of r. The following table gives these ratios. The first three lines of the table gives the average ratio taking all generated profiles into account, and the last one the average ratio when only the 'unsuccessful' profiles, that is, those for which the issue-by-issue winner and the winner for r differ. (Note that using these ratios for comparing different rules should be done with care, as these ratios depend heavily on the definition of the score.) For the sake of completeness we also give the success rate for separable profiles.

Once again, the best results are obtained for Borda, Copeland and Maximin: for these rules, the average approximation ratio is above 97%). The result are not as good for Plurality and $\frac{m}{2}$-approval.

$rule$:	Borda	plur.	$\frac{m}{2}$-app.	Cop.	maxim.
General case					
Average score of the real winner	72.71	1.9	5.73	14.22	3.47
Average score of the issue-by-issue winner	70.98	1.3	5.08	13.94	3.38
Ratio	0.976	0.684	0.887	0.98	0.974
Success rate	0.608	0.570	0.527	0.838	0.907
Unsuccessful profiles					
Average score of the real winner	70.59	1.99	5.77	13.36	3
Average score of the issue-by-issue winner	66.18	0.6	4.39	11.65	1.97
Ratio	0.938	0.302	0.761	0.872	0.657
Maximal distance	21	3	4	8	2
Minimal distance	1	1	1	1	1

Figure 6. Distance in the scores of the real and issue-by-issue winners , 7 voters and 4 issues, weakly separable profiles.

4.4 Pareto Efficiency

Recall that one of the drawbacks of issue-by-issue voting is its failure to satisfy efficiency as soon as $p \geq 3$. We give here the probability that the issue-by-issue winner is Pareto-dominated. An analytical study for $p = 3, m = 3$ gives a probability of $\frac{1}{2304}$ for strongly separable profiles and $\frac{1}{18432}$ for weakly separable profiles. For more than 3 issues or 3 voters, computing the probability analytically seems difficult, therefore we have once again proceeded to experiments with randomly generated profiles. For each couple $(\#issue, \#voters)$ we generated a set of 10^6 additively profiles, and for each of them, checked if the outcome is Pareto-dominated. Figure 7 give our results for additive separable profiles.

	$m = 3$	$m = 5$	$m = 7$	$m = 9$	$m = 11$
$p = 3$	436	20	4	0	0
$p = 4$	1699	111	6	0	0
$p = 5$	4211	329	23	4	0
$p = 6$	8268	671	34	4	0
$p = 7$	14052	1360	86	3	0
$p = 8$	21984	2217	150	14	0
$p = 9$	32324	3378	284	16	0

Figure 7. Pareto-dominated profiles for 10^6 profiles; additive separability

We can see that even if the probability of having a Pareto-dominated outcome is non-negligible when the number of voters is low (up to 3,2% for 3 voters and 9 issues), this probability decreases very quickly as the number of voters increases, and becomes negligible from 9 voters on – which is of course very unsurprising. Similar tests on weakly separable profiles give a probability of getting a Pareto-dominated outcome 10 to 100 times smaller than with additively separable profiles, the shape of the graphics being similar.

5 Conclusion

Although the initial motivation of this paper was an experimental comparison between issue-by-issue voting and common voting rules applied to separable profiles, it turned out that a surprisingly difficult issue that had to be addressed first was the random generation of separable profiles. As soon as there are at least 5 issues, we do not know how to generate weakly separable, separable nor additively separable profiles with an equiprobable distribution; we remedied this to some extent, by proposing two methods that generate distributions 'not too far' from equiprobability, and finally chose an algorithm based on the exploration of the lattice of alternatives for the generation of weakly separable profiles, and an algorithm relying on a utility-based representation of preferences, for strongly separable profiles. The first one performs well in terms of entropy (it is close to be equiprobable) and the second one has the advantage of being based on a well known (and well characterized) model of rational decision makers.

Concerning issue-by-issue voting, our result are rather negative, confirming the theoretical results in [8]. Although they show that the violation of efficiency is rare (which was expected), they also show that issue-by-issue voting is a bad approximation of scoring rules, in particular plurality. For r = plurality, Borda or $\frac{m}{2}$-approval, the issue-by-issue winner is different of the one elected by r for more than 70% of the profiles, even for only 4 issues and 5 voters. These results become worse when the number of issues or the number of voters increase. Copeland and (to a lesser extent) maximin do better: for instance, their winners coincide with the issue-by-issue winner for $80\%$of the profiles, for 4 issues. Unsurprisingly, our results also confirm that issue-by-issue behaves better on separable preference than on purely random profiles.

REFERENCES

[1] J.-P. Benoit and L.A. Kornhauser, 'Only a dictatorship is efficient or neutral', Technical report, NYU Law School, (2006).

[2] 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', *JAIR*, **21**, 135–191, (2004).

[3] W. Bradley, J. Hodge, and M Kilgour, 'Separable discrete preferences', *Mathematical Social Sciences*, **49**(3), 335–353, (2005).

[4] S. Brams and P. Fishburn, 'Voting procedures', in *Handbook of Social Choice and Welfare, Volume 1*, eds., K. Arrow, A. Sen, and K. Suzumura, 173–236, Elsevier, (2002).

[5] S. Brams, W. Zwicker, and M. Kilgour, 'The paradox of multiple elections', *Social Choice and Welfare*, **15**(2), 211–236, (1998).

[6] D. Braziunas and C. Boutilier, 'Local utility elicitation in gai models', in *Proceedings of UAI'05*, pp. 42–49, (2005).

[7] V. Conitzer, J. Lang, and L. Xia, 'Hypercubewise preference aggregation on multi-issue domains', in *IJCAI*, (2011).

[8] V. Conitzer and L. Xia, 'Approximation of common voting rules by sequential voting rules on multi-issue domains', in *KR*, (2012). to appear.

[9] W.M. Gorman, 'The structure of utility functions', *Review of Economic Studies*, **35**, 367–390, (1968).

[10] U. Grandi and U. Endriss, 'Aggregating dependency graphs into voting agendas in multi-issue elections', in *IJCAI*, pp. 18–23, (2011).

[11] J. Hodge, 'Permutations of separable preference orders', *Discrete Appl. Math.*, **154**, 1478–1499, (2006).

[12] J. Hodge and P. Schwallier, 'How does separability affect the desirability of referendum election outcomes?', *Theory and Decision*, **61**(3), 251–276, (2006).

[13] R. L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley and Sons, 1976.

[14] C. Kraft, J. Pratt, and A. Seidenberg, 'Intuitive probability on finite sets', *The Annals of Mathematical Statistics*, **30**(2), 408–419, (1959).

[15] D. Lacy and E. Niou, 'A problem with referenda', *J. of Theoretical Politics*, **12**(1), 5–31, (2000).

[16] J. Lang and L. Xia, 'Sequential composition of voting rules in multi-issue domains', *Mathematical Social Sciences*, 304–324, (2009).

[17] I. Oskal-Sanver and R. Sanver, 'Ensuring pareto optimality by referendum voting', *Social Choice and Welfare*, **27**(1), 211–219, (Aug. 2006).

[18] G. Dalla Pozza, F. Rossi, M.-S. Pini, and B. Venable, 'Multi-agent soft constraint aggregation via sequential voting', in *IJCAI*, pp. 172–177, (2011).

[19] L. Xia and J. Lang, 'A dichotomy theorem on the existence of efficient or neutral sequential voting correspondences', in *IJCAI*, (2009).

Elicitation-free Protocols for Allocating Indivisble Goods

Thomas Kalinowski[1] and **Nina Narodytska**[2] and **Toby Walsh**[3] and **Lirong Xia**[4]

Abstract. We study in detail a simple sequential procedure for allocating a set of indivisible goods to multiple agents. Agents take turns to pick items according to a policy. For example, in the alternating policy, agents simply alternate who picks the next item. A similar procedure has been used by Harvard Business School to allocate courses to students. We study here the impact of strategic behavior on the complete-information extensive-form game of such sequential allocation procedures. We show that computing the subgame-perfect Nash equilibrium is PSPACE-hard in general, but takes only linear time with two agents. Finally we compute the optimal policies for two agents in different settings, including when agents behave strategically and when agents can give away items.

1 Introduction

Suppose you are coaching a soccer team. To divide the players into two teams, you select the two best players as captains and then let them alternate at picking the remaining team members. Is this the best way to get an evenly matched game? Perhaps it would be better to reverse the order of their picks every round (so that the captain who picks first in the first round picks second in the second round)? This is an example of a problem in allocating indivisible goods. A number of real world problems involve allocating indivisible goods "fairly" between competing agents subject to possibly different preferences for these goods. For example, assigning courses to students at a business school is a problem of allocating indivisible goods. Students are competing for places on the popular courses, but have different preferences as to which courses to study. As a second example, the allocation of landing and take-off slots at an airport is a problem of allocating indivisible goods. Airlines are competing for popular landing and take-off times, but have different preferences as to precisely which slots they want. As a third and final example, sharing time slots on an expensive telescope is a problem of allocating indivisible goods. Astronomers are competing for observation time but have different preferences as to precisely which time slots are useful for their experiments.

Different properties might be demanded of a procedure for allocating indivisble goods. For example, we might look for allocations which are envy-free in the sense that every agent likes their allocation at least as much as the allocation to any other agent. However, envy-freeness by itself is not sufficient to ensure a "good" allocation. Not allocating any items is envy-fee, and there are also many situations where no envy-free allocation exists. We might consider other criteria including efficiency (e.g. Pareto optimality) and truthfulness (e.g. can agents profit by acting strategically?). There is, however, a tension between these properties. Svensson showed that the only strategy-proof, nonbossy[5] and neutral mechanism is a serial dictatorship in which agents take turns according to some order to pick their complete allocation of goods [5]. Unfortunately, a serial dictatorship can have a low efficiency in the utilitarian or egalitarian sense. In this paper, we focus on efficiency, and consider the impact on efficiency of such issues like the strategic behavior of the agents.

2 Existing methods

Several non-strategy proof procedures for allocating indivisible goods have been studied. For example, Brams, Kilgour and Klamler have proposed the undercut procedure for two agents [2]. This is the discrete analog of the "cut-and-choose" cake cutting procedure for divisible goods. The first agent divides the contested goods into two sets, and offers one set to the other agent. The second agent can either accept this set or take any strict subset of the goods in the complement set. They characterize when this procedure leads to an envy-free division. Whilst the undercut procedure is not strategy proof, the maximin strategy is truthfulness.

As a second example, the Harvard Business School has been using a mechanism called *Draft* to allocate courses to students [3]. The Draft mechanism generates a priority order over all students uniformly at random. Course are then allocated to students in rounds. In odd rounds, each student is assigned to their favorite course that still has availability using the priority order. In even rounds, the mechanism uses the reverse priority order. The Draft mechanism is not strategy-proof. Indeed, students at Harvard have been observed to behave strategically [3]. Such strategic behavior can be harmful to the ex post social welfare. However, the expected (ex ante) social welfare is higher than that of a strategy-proof mechanism like serial dictatorship. To obivate the need for certain types of manipulation, Kominers, Ruberry and Ullman [4] proposed a mechanism in which proxies play strategically. They prove that with lexicographical preferences, this proxy mechanism is Pareto efficient.

As a third example, Bouveret and Lang ([1]) consider a simple sequential allocation procedure which generalizes many aspects of the Draft mechanism (but ignores the inital randomization of the order of the students). The procedure is parameterized by a policy, the sequence in which agents take turns to pick items. This policy is fixed and assumed to be known to the agents in advance. For example, as in the Draft mechanism, with two agents and four items, the policy 1221 gives first and last pick to the first agent, and second and third pick to the second agent. This procedure has the advantage the preference of the agents do not need to be elicited. Bouveret and

[1] Universität Rostock, Rostock, Germany. email: thomas.kalinowski@uni-rostock.de
[2] NICTA and UNSW, Sydney, Australia. email: nina.narodytska@nicta.com.au
[3] NICTA and UNSW, Sydney, Australia. email: toby.walsh@nicta.com.au
[4] Harvard University, Cambridge, USA. email: xialirong@gmail.com

[5] A mechanism is nonbossy if when an agent submits different preferneces and their allocation does not change then the overall allocation does not change.

Lang assume agents have additive utilities given by a common scoring function (e.g. Borda, lexicographic or quasi-indifferent scores). They consider two extreme cases: full correlation in which preference orderings of the agents are identical, and full independence in which all preference orderings are equally probable. With full correlation, all policies give the same expected sum of utilities, and the sequential allocation procedure is strategy proof. With lexicographical scores, they show that the optimal strategy for an agent given a particular policy can be computed in polynomial time supposing other agents pick truthfully. The contribution of our paper is to study this sequential allocation procedure in more detail.

3 Preliminaries

Let $\mathcal{I} = \{c_1, \ldots, c_m\}$ denote a set of m indivisible goods, and $\mathcal{A} = \{A_1, \ldots, A_n\}$ denote a set of n agents. For any $j \leq n$, let $u_j : \mathcal{I} \rightarrow \mathcal{R}$ denote the utility function of agent A_j over $\mathcal{I}$. We assume $m \geq n$, and all agents have strict preferences. That is, for any $j \leq n$ and any pair of items $\{c, c'\}$, $u_j(c) \neq u_j(c')$. We suppose that an agent's utility function is *additive*. For any $j \leq n$ and any set of items $G \subseteq \mathcal{I}$, $u_j(G) = \sum_{c \in G} u_j(c)$. For any $j \leq n$, let O_j denote the ordinal preferences of agent j. That is, O_j is a total strict order over $\mathcal{I}$ and for any pair of items $\{c, c'\}$, $c \succ c'$ in O_j if and only if $u_j(c) > u_j(c')$. An agent has *Borda* utility, if for any $i \leq m$, the utility of the item ranked in i-th position in O_j is $m-i$. An agent has *lexicographic* utility, if for any $i \leq m$, the utility of the item ranked in i-th position in O_j is 2^{m-i}. An *allocation* is a function $f : \mathcal{I} \rightarrow \mathcal{A}$. For any agent $A \in \mathcal{A}$, $f^{-1}(A)$ denote the set of items allocated to A. A sequential allocation is a mechanism parameterized by a *policy* P. This can be represented by an ordering over m elements taken from $\mathcal{A}$ (e.g. $P = [A_1 \succ A_2 \succ A_1]$). Agents take turns to pick items according to this ordering.

4 Optimal Policies

Bouveret and Lang considered which policies maximize the social welfare of the agents supposing the preference of agents are independent and every preference ordering is equally likely [1]. They considered an utilitarian principle in which social welfare is measured by the expected sum of the utilities of the agents (ExpSumUtil). They demonstrated that the simple alternating policy $121212\ldots$ optimizes the social welfare when utilities are Borda score (i.e. where the ith ranked of m items has an utility of $m - i$) and up to 12 items. Interestingly, there exist situations where the policy that maximizes the sum of the utilities is not alternating. In fact, it need not even be balanced (that is, it might not assign an equal number of items to both agents).

Example 1. *Consider 8 items, a to h, 2 agents and utilities which are Borda scores. Suppose agent 1 has the preference order $a > \ldots > h$ whilst agent 2 has the order $a > h > b > c > d > e > f > g$. Then, supposing the agents pick items truthfully, the alternating policy* 12121212 *gives a social welfare of 22+16=38 but the optimal policy is* 22111111 *which gives a social welfare of 27+15=42. Note that the optimal policy does not Pareto dominate the alternating policy since, whilst the optimal policy increaes the utility for agent 1, the utility for agent 2 decreases slightly.*

Of course, an alternating policy can still be the best policy in expectation even if there are individual situations like the above where it is not the best. Bouveret and Lang also considered a rather unusual egalitarian principle in which social welfare is measured by the minimum of the expected utilities of the different agents (MinExpUtil). We consider two more "usual" measures of egalitarian social welfare: the expected minimum utility of the different agents (ExpMinUtil) and the minimum utility of the different agents over all possible worlds (MinUtil). In the economics literature, MinExpUtil is called the ex-ante egalitarian utility, whilt ExpMinUtil is called the ex-post egalitarian utility.

To illustrate the difference between the three measures, consider the following two protocols. In the first, we toss a coin. If it lands on heads, we assign all m items to agent 1, otherwise we assign all items to agent 2. In the second protocol, we assign $m/2$ items at random to agent 1 and the rest to agent 2. The second protocol is more egalitarian than the first since one agent is sure to get no items in the first protocol whilst each agent is allocated $m/2$ items in the second protocol. This is reflected in the expected minimum of the two utilities (which is zero for the first protocol and half the total utility for the second protocol), and in the minimum utility (which is zero for the first protocol, and the sum of utilities of the least valuable $m/2$ items for the second protocol). However, the minimum of the expected utilities hides this difference as both protocols have a minimum expected utility that is half the total. We have the following proposition, whose proof is straightforward and is omitted.

Proposition 1. *For any policy and any distribution over utility functions:* MinUtil<ExpMinUtil<MinExpUtil

Note that, whilst the minimum utility (MinUtil) often occurs in the full correlation case where agents utilities are identical [1], it can also occur when the utilities of the agents are different. For instance, suppose we are dividing just two items between two agents. Consider the protocol where the two agents declare which of the two items that they like most. If the two agents most prefer the same item, then we toss a coin to decide which agent gets this item, and assign the remaining, less preferred item to the other agent. On the other hand, if the two agents most prefers different items, we toss a coin and assign both items to an agent chosen at random. The minimum utility is now zero and occurs when the two agents most prefer different items. The full correlation case increases MinUtil to the smallest utility assigned to either object.

For the case of two agents, we computed the policies that maximize the three different egalitarian measures of social welfare using brute force search. Table 1 demonstrates that the optimal policies for maximizing `ExpMinUtil` and `MinExpUtil` differ. We conjecture that the optimal `ExpMinUtil` policy has the form: $(12)^k 2$ for $m = 2k+1$, $(12)^k(21)^k$ for $m = 4k$ and $(12)^k(21)^{k-1}$ for $m = 4k-2$. In addition, we conjecture that the optimal `ExpMinUtil` policy for an even number of items is also an optimal `MinUtil` policy.

To return to our soccer example, suppose there are ten players to divide into two teams, utilities are Borda scores, and we adopt an egalitarian position to help ensure a balanced match. We might then select the two best players as team captains and, based on the optimality of the policy 12122121, have the first team captain pick first, third, sixth and eigth, and the second team captain pick otherwise.

As in [1], we also considered two other scoring models: lexicographic scoring (where an item at position k is scored 2^{-k}) and quasi-indifferent (where an item at position k is scored $a - k$ for $a \gg n$). We consider both an egalitarian model (the ExpMinUtil and MinUtil policies in which we maximize the expected or actual minimum utilities) and an utilitarian model (the ExpSumUtil policy in which we maximize the expected sum of the utilities). In Tables 2 and 3, we report the optimal policies for lexicographical and quasi-indifferent scoring.

m	MINEXPUTIL	EXPMINUTIL	MINUTIL
1	1	1	1
2	12	12	12
3	122	122	122
4	1221	1221	1221
5	*11222*	12122	12122, 12212, 12211
6	121221	121221	121221, ...
7	*1122122*	1212122	*1212212*, ...
8	*12212112*	12122121	*11222122*, ...

Table 1. Optimal policies that maximize the minimum of the two expected utilities (`MinExpUtil`), the expected minimum of the two utilities (`ExpMinUtil`) and the minimun utility (`MinUtil`). In each case, we allocate m items, assign utilities using Borda scoring, and assume full independence between the two agents. *Emphasis* is added to highlight when policies start to differ.

m	EXPMINUTIL egalitarian	MINUTIL egalitarian	EXPSUMUTIL utilitarian
1	1	1	1
2	12	12	12
3	122	122	121
4	1221	1222	1212
5	12122	12222	12121
6	122121	122222	121212
7	1221211	1222222	1212121
8	12212112	12222222	12121212

Table 2. Optimal policies that maximize the expected minimum of the utilities (EXPMINUTIL), maximize the minimun utility (MINUTIL) and maximize the expected sum of utilities (EXPSUMUTIL). In each case, we allocate m objects, assign utilities using lexicographical scoring, and assume full independence between the two agents.

m	EXPMINUTIL egalitarian	MINUTIL egalitarian	EXPSUMUTIL utilitarian
1	1	1	1
2	12	12	12
3	122	122	121
4	1221	1221	1212
5	11222	11222	12121
6	121221	121221, ...	121212
7	1112222	1112222	1212121
8	12122121	11222211, ...	12121212

Table 3. Optimal policies that maximize the expected minimum of the utilities (EXPMINUTIL), maximize the minimun utility (MINUTIL) and maximize the expected sum of utilities (EXPSUMUTIL). In each case, we allocate m objects, assign utilities using quasi-indifferent scoring, and assume full independence between the two agents.

We make some observations about these results. First, in both scoring models, a simple alternating policy is optimal under the utilitarian assumption. It seems likely that the expected sum of utilities is maximized for a wide variety of scoring functions by this policy. Second, for the quasi-indifferent scoring function, the same policy is optimal for EXPMINUTIL and MINUTIL. This was not the case for the lexicographial scoring model. For Borda scoring, the same policy was optimal for EXPMINUTIL and MINUTIL only for even n.

5 Strategic Behavior

Another desirable property of an allocation procedure is strategy-proofness. A sequential allocation procedure is strategy-proof if for any utility functions, the agents are best off choosing their top ranked item still available at every step. Unfortunately, the sequential allocation procedure is not strategy-proof in general. For instance, the first agent to pick an item might not pick their most preferred item if this is the item least preferred by the other agent. The first agent might strategically pick some other item as the second agent will not pick this first item unless there is no other choice. Bouveret and Lang [1] argue that the sequential allocation procedure is strategy-proof when agents have the same preference rankings. They also gave a polynomial time method for a single agent to compute a manipulation supposing all other agents act truthfully and utilities are lexicographic. Supposing all agents but the manipulator act truthfully is a strong assumption. If one agent is acting strategically, why not the others?

The sequential allocation procedure naturally lends itself to a game theoretic analysis in which all agents can act strategically. Assuming that the agents know the utility functions of other agents, we can model the sequential allocation procedure as a complete information *extensive-form game*. The subgame-perfect Nash equilibrium (SPNE) gives the (perhaps untruthful) strategy in which agents cannot improve their allocation by deviating unilaterally. The SPNE can be computed by *backward-induction* as follows. We start with the last agent A in the order P. For any allocation of items in the previous rounds, only one item remains, and A will get it. Then, we move to the second to the last agent A' in P. For any allocation of items in previous round, A' can predict the final allocation for any item she picks. Therefore, she can pick an item that maximizes her total utility in the final allocation. We then move on to the third to the last agent in P, etc. Since an agent can obtain the same total utility for picking different items, there might be multiple SPNE.

Example 2. *Suppose there are two agents and four items. Agent 1's ordinal preferences are* $O_1 = c_1 \succ c_2 \succ c_3 \succ c_4$ *and agent 2's ordinal preferences are* $O_2 = c_2 \succ c_3 \succ c_4 \succ c_1$. *Let* $P = A_1 \succ A_2 \succ A_2 \succ A_1$. *If all agents behave truthfully, then* A_1 *chooses* c_1 *in the first round,* A_2 *chooses* c_2 *and* c_3 *in the second and third rounds, respectively, and* A_1 *chooses* c_4 *in the last round. If the agents behave strategically, then* A_1 *can choose* c_2 *in the first round, and still get* c_1 *in the last round. The unique SPNE allocation in this game has* A_1 *getting* $\{c_1, c_2\}$ *and* A_2 *getting* $\{c_3, c_4\}$.

In the above example, even though there are multiple SPNE, the final allocation is unique regardless of the utility functions. We will see later that this is not a coincidence. When there are two agents, the SPNE allocation is always unique (and indeed can be computed in linear time). The next example shows that with three or more agents, there can be multiple SPNE allocations.

Example 3. *Suppose there are four items and three agents with Borda utilities. The ordinal preferences of the agents are as follows.* $A_1 : c_1 \succ c_2 \succ c_3 \succ c_4$, $A_2 : c_3 \succ c_4 \succ \ldots$, *and*

$A_3 : c_1 \succ c_2 \succ l \ldots$. *Let* $P = A_1 \succ A_2 \succ A_3 \succ A_1$. *There are two SPNE allocations: (1) if* A_1 *picks* c_1 *in the first round, then in the SPNE* A_1 *gets* $\{c_1, c_4\}$*,* A_2 *gets* c_3*, and* A_3 *gets* c_2*; (2) if* A_1 *picks* c_3 *in the first round, then in the SPNE* A_1 *gets* $\{c_2, c_3\}$*,* A_2 *gets* c_4*, and* A_3 *gets* c_1*.*

5.1 Computing SPNE for Two Agents

With two agents and m items, computing the subgame-perfect Nash equilibrium by backward induction takes $\Omega(m!)$ time. This will be prohibitive when we have many items. The SPNE can, however, be computed in just $O(m)$ time by means of the following result. Let u_1, u_2 be the utility functions of the two agents, O_1, O_2 be their ordinal preferences, and P be the policy. We let $\text{Seq}(O_1, O_2, P)$ denote the truthful sequential allocation. We use $\text{SPNE}(u_1, u_2, P)$ to denote the subgame-perfect Nash equilibrium allocation. For any total strict order O, let $rev(O)$ denote the reversed order. Then, we can show that the SPNE allocation is unique, and can be computed from the truthful sequential allocation for the reversed preference orderings and policy.

Theorem 1. *When there are two agents, the SPNE allocation is unique. Moreover,*
$\textit{SPNE}(u_1, u_2, P) = \textit{Seq}(rev(O_2), rev(O_1), rev(P))$

Proof: (Sketch) W.l.o.g. suppose agent 1 has the last pick in policy P (and thus the first pick in policy $rev(P)$). Then, agent 1 knows that the item that is ranked last in O_2 is "safe", as agent 2 has no incentive to pick it in earlier rounds. Therefore, agent 1 can safely pick this item in her last round, and leave opportunities in previous rounds in P to pick more popular items. The formal proof is much more involved and is proved by induction on the number of items m. ♣

Example 4. *Suppose there are two agents and four items. The agents' preferences and the policy are the same as in Example 2. We have* $rev(P) = P$*. In* $\textit{Seq}(rev(O_2), rev(O_1), rev(P))$*,* A_1 *picks* c_1 *in the first round,* A_2 *picks* c_3 *and* c_4 *in the second round and third round respectively, and* A_1 *picks* c_2 *in the last round. This outcome is the same as the SPNE allocation in Example 2.*

5.2 Computing SPNE for more than Two Agents

When the number of agents n is comparable to the number of items m (more precisely, when $n = O(m)$), we prove that computing the SPNE is intractable. Consider the decision problem SUBGAMEPERFECT, where we are given the utility functions of n agents over m items, a particular agent A, a policy P, and a threshold T, and we are asked whether the utility of A is larger than T in any SPNE.

Theorem 2. SUBGAMEPERFECT *is PSPACE-complete for Borda scoring of utilities.*

Proof: Backward induction shows that it is in PSPACE. To show hardness, we give a reduction from QSAT, which is a standard PSPACE-complete problem. In a QSAT instance, We are given a quantified formula $\exists x_1 \forall x_2 \exists x_3 \cdots \forall x_q \,.\, \varphi$ where q is even and we are ask whether the formula is true. Let $\varphi = C^1 \wedge \cdots \wedge C^t$, where C^j is a 3-clause, $l_j^1 \vee l_j^2 \vee l_j^3$. We construct a SUBGAMEPERFECT instance where there is a unique SPNE with a utility to the first player larger than a threshold if and only if the formula is true.

In the SUBGAMEPERFECT instance, there are q agents who represent the binary variables. Each of these agents choosing one out of two items represents a valuation of the variable. The agents that correspond to $\exists$ quantifiers (that is, agents 1, 3,. . ., $q-1$) obtain higher utility if φ is true under the current valuation, and the agents that correspond to $\forall$ quantifiers (that is, agents 2, 4,. . ., q) obtain higher utility if φ is false under the current valuation. There are also some other agents that are used to encode the QSAT instance, which we will specify later.

Let a be an item, and k, p be natural numbers. We define an ordering $O_p^k(a)$ that will be used as part of the policy P as follows. It introduces $2k+1$ new agents $A_p^1, \ldots, A_p^{2k+1}$ and $5k+1$ new items $\{a_p, b_p^1, \ldots, b_p^k, \quad c_p^1, \ldots, c_p^k, d_p^1, \ldots, d_p^k, e_p^1, \ldots, e_p^k, f_p^1, \ldots, f_p^k\}$. The preferences of the new agents are as follows:

Agent	Preferences
A_p^1	$b_p^1 \succ c_p^1 \succ d_p^1 \succ e_p^1 \succ$ Others
$\vdots$	$\vdots$
A_p^k	$b_p^k \succ c_p^k \succ d_p^k \succ e_p^k \succ$ Others
A_p^{k+1}	$c_p^1 \succ f_p^1 \succ$ Others
$\vdots$	$\vdots$
A_p^{2k}	$c_p^k \succ f_p^k \succ$ Others
A_p^{2k+1}	$a \succ b_p^k \succ \cdots \succ b_p^1 \succ a_p \succ$ Others

Let the order over agents be $A_p^1 \succ \cdots \succ A_p^{2k+1} \succ A_p^1 \succ \cdots \succ A_p^2$. In $O_q^k(a)$, a is the item that we want to "duplicate", k is the number of duplicates, and q is merely an index. We can prove by induction that if a has not been chosen (in previous rounds), then after agents have chosen items according to $O_q^k(a)$, $\{f_p^1, \ldots, f_p^k\}$ will be chosen and $\{d_p^1, \ldots, d_p^k\}$ will not be chosen; if a has been chosen, then $\{d_p^1, \ldots, d_p^k\}$ will be chosen rather than $\{f_p^1, \ldots, f_p^k\}$.

We now specify the sequential allocation instance by using the orderings $O_p^k(a)$. All agents introduced in $O_p^k(a)$ will not appear in other places in the policy P. For each $i \le q$, there are two items 0_i and 1_i that represent the two values of x_i, an agent A_i corresponding to the valuation and another agent B_i that is used to make sure that A_i chooses 0_i or 1_i in the $(q+2i-1)$th round. For each $i \le q$, D_i is an agent whose preferences are $d_i \succ$ Others, where d_i is a new item that creates a "gap" between items available to agent A_i. The first $(2t+4)q$ agents in P are the following: $D_1 \succ \cdots \succ D_q \succ A_1 \succ \cdots \succ A_q \succ O_1^t(0_1) \succ \cdots \succ O_q^t(0_q) \succ B_1 \succ \cdots \succ B_q$. The preferences of B_i are $0_i \succ 1_i \succ$ Others. The preferences of A_i will be defined after we have defined all items and have specified P. For notational convenience, for each $i \le q$ and each $j \le t$ we rename d_i^j to be 0_i^j, and rename f_i^j to be 1_i^j.

For each clause C^i, we have an agent denoted by C_i. Suppose v_{j_1}, v_{j_2}, and v_{j_3} correspond to the 3 valuations that satisfy C_i, then we let the preferences of C_i be $v_{j_1}^i \succ v_{j_2}^i \succ v_{j_3}^i \succ g \succ g_i' \succ$ Others, where g and g_i' are new items. g is used to detect whether a clause is not satisfied. For example, suppose $C^i = x_1 \vee \neg x_2 \vee x_3$, then the preferences of C_i are $1_1^i \succ 0_2^i \succ 1_3^i \succ g \succ g_i' \succ$ Others. The remaining agents in the P are: $C_1 \succ \cdots \succ C_t \succ O_{q+1}^q(g) \succ A_1 \succ \cdots \succ A_q$.

The agents and new items introduced in $O_{q+1}^q(g)$ impose "feedback" on A_1 through A_q, such that if g is allocated before $O_{q+1}^q(g)$ (which means that the formula is not satisfied under the valuation encoded in the first q rounds), then some items that are more valuable to the agents that correspond to the $\forall$ quantifiers are made available; if g is not allocated before $O_{q+1}^q(g)$, then some items that are more valuable to the agents that correspond to the $\exists$ quantifiers are made available. Finally, for each $i \le q$, we define the ordinal preferences of A_i as follows. If i is odd, then A_i's preferences are

$0_i \succ 1_i \succ d^i_{q+1} \succ d_i \succ f^i_{q+1} \succ \ldots$. If i is even, then A_i's preferences are $0_i \succ 1_i \succ f^i_{q+1} \succ d_i \succ d^i_{q+1} \succ \ldots$.

To summarize, in the sequential allocation instance, there are $3q+t+(2t+1)q+2q+1$ agents and $m = 3q+(5t+1)q+1+t+5q+1$ items, which are polynomial in the size of the formula ($\Omega(t+q)$). Table 4 summarizes the items introduced in the reduction. Final, the

for	items	Introduced in
$i \leq q$	d_i	D_i
$i \leq q$	$0_i, 1_i$	A_i
$i \leq q, j \leq t$	a_i b^j_i c^j_i d^j_i (a.k.a. 0^j_i) e^j_i f^j_i (a.k.a. 1^j_i)	$O^j_i(0_i)$
	g	C_1
$j \leq t$	g'_t	C_j
$j \leq q$	$a_{q+1}, b^j_{q+1}, c^j_{q+1}$, $d^j_{q+1}, e^j_{q+1}, f^j_{q+1}$	$O^q_{q+1}(g)$

Table 4. Items introduced in the reduction.

policy P ordering over agents is the following.

$$\begin{aligned} D_1 \succ \cdots \succ D_q \succ A_1 \succ \cdots \succ A_q \succ O^t_1(0_1) \succ \cdots \succ O^t_q(0_q) \\ \succ B_1 \succ \cdots \succ B_q \succ C_1 \succ \cdots \succ C_t \succ O^q_{q+1}(g) \\ \succ A_1 \succ \cdots \succ A_q \end{aligned}$$

If we must allocate all items then we can add some dummy agents to the end of the ordering.

We note that if an agent only appears once in the ordering, then it is her strictly dominant strategy to pick her most preferred available item. In any SPNE, in the first q rounds $d_1, \ldots, d_q$ will be chosen. In the next q rounds, agent i must choose either 0_i or 1_i, otherwise 0_i will be chosen by agent A^{2t+1}_i introduced in $O^t_i(0_i)$ and 1_i will be chosen by B_i. Hence, the choices of agents A_i correspond to valuations of the variables, and these valuations are duplicated by $O^t_i(0_i)$ that will be used to satisfy clauses. (We note that if A_i chooses 0_i, then after $O^t_i(0_i)$, $\{0^1_i, \ldots, 0^t_i\}$ are still available, but $\{1^1_i, \ldots, 1^t_i\}$ are not available; and vice versa.) Then, a clause C^i is satisfied if and only if at least one of the top 3 items of agent C_i is available (otherwise C_i chooses g). Hence, after agent C_t, g is available if and only if all clauses are satisfied. Finally, if g is available after agent C_t, then the agents that correspond to the $\exists$ quantifiers can choose d_{q+1}'s to increase their total utility by $m-3$, but the agents that correspond to the $\forall$ quantifiers can only choose d_{q+1}'s to increase their utility by $m-5$; and vice versa. Hence, the agents that correspond to $\exists$ quantifiers will choose valuations to make F true, while the agents that correspond to $\forall$ quantifiers will choose valuations to make F false. It can be verified that there is a unique SPNE allocation, where agent A_1's utility is at least $2m-5$ (that is, she gets one of $\{0_1, 1_1\}$ and d^1_{q+1}) if and only if the formula F is true. ♣

6 Optimal Policies for Strategic Behavior

Suppose agents act strategically instead of truthfully. For example, suppose they pick items according to the subgame-perfect Nash equilibrium. The policies which maximize social welfare can now change. For a reversal symmetric scoring function like Borda, and a reversal symmetric policy like the simple alternating policy, it is easy to see that the situations where strategic behavior decreases social welfare will be exactly balanced by the symmetric situations where it increases social welfare. As a result, we did not observe any difference in the policies that optimizes social welfare for Borda scoring when agents behave strategically instead of truthfully. For example, brute force calculation with up to 8 items show that the expected sum of the utilities of the agents supposing Borda scoring is maximized by the same simple alternating policy whether agents pick either truthfully or strategically.

Strategic behavior can sometimes increase the social welfare of the agents. In other cases, it can decrease the social welfare of the agents or leave it unchanged. In fact, given the reversal symmetry of the optimal policy, and of the subgame perfect equilibrium, Borda scoring and the utilitarian criterium, we can prove that the cases when the utilitarian social welfare increases are exactly matched by cases where it decreases. With an egalitarian criterium, strategic behavior can improve social welfare slightly more often than it can decrease it. Averaged over all possible preference profiles, brute force calculations suggest that the expected sum of the utilities barely changes, whilst the expected minimum increases by less than 1%.

For scoring functions that are not symmetric, the optimal policy can change. For example, with lexicographical scores, the optimal policy for strategic behavior is different from that for truthful behavior. Table 5 summarizes results based on brute force calculation. When maximizing the expected minimum utility, the optimal policies for agents playing strategically are optimal policies for agents playing truthfully for 6 or fewer items. However, the optimal policy for strategic play with 7 items is 1221122 but for truthful play is 1221211. Similarly, for 8 items, the optimal policy for strategic play is 12212211 but for truthful play is 12212112. When maximizing the minimum utility, the optimal policies for strategic play are optimal policies for truthful play. When maximizing the expected sum of utilities and 4 or more items, the optimal policies for strategic play are not optimal alternating policies for truthful play.

n	`ExpMinUtil` egalitarian	`MinUtil` egalitarian	`ExpSumUtil` utilitarian
1	1	1	1
2	12	12	12
3	122	122	121
4	1221	1222	1212, *1221*
5	12122	12222	*12122*
6	122121	122222	*122112*
7	*1221122*	1222222	*1212122*
8	*12212211*	12222222	*12211221*

Table 5. Optimal policies when we assign utilities using lexicographical scoring, and assume agents play strategically by computing the subgame-perfect Nash equilibrium. *Emphasis* is added to highlight when policies differ from the optimal truthful policies.

We conjecture that the optimal `ExpMinUtil` policy supposing strategic behavior has the alternating form: $(1221)^k21$ for $m = 4k+2$, $(1221)^k122$ for $m = 4k+3$ and $(1221)^k2211$ for $m = 4k+4$. We also conjecture that the optimal `ExpSumUtil` policy supposing strategic behavior has the alternating form: $(12)^k122$ for $m = 2k+3$, $1(2211)^k2$ for $m = 4k+2$, and $1(2211)^k221$ for $m = 4k+4$. Strategic play also carries a small cost. Averaged over all possible preference profiles, the utility decreases by 5% or less for both the

expected sum and minimum of utilities.

As in [1], we also considered quasi-indifferent scoring. With quasi-indifferent scoring, an item at position k in an agent's ordering is given score $a - k$ where $a \gg n$ and n is the number of items. In Table 6, we give the optimal policies for agents playing strategically when agents are quasi-indifferent between items. The optimal policy for agents playing strategically is also the optimal policy for agents playing truthfully except $n = 6$ and the egalitarian criterium of maximizing the expected minimum utility. When agents play strategically, the optimal policy in this case is 122121. However, when agents play truthfully, the optimal policy in this case is 121221.

n	ExpMinUtil egalitarian	MinUtil egalitarian	ExpSumUtil utilitarian
1	1	1	1
2	12	12	12
3	122	122	121
4	1221	1221	1212
5	11222	11222	12121
6	122121	121221, ...	121212
7	1112222	1112222	1212121
8	12122121	12211221, ...	12121212

Table 6. Optimal policies when we assign utilities using a quasi-indifferent scoring function, and assume agents play strategically by computing the subgame perfect equilibrium.

7 Disposal of Items

One inefficiency of the policies considered so far is that one agent may use one of their early choices to select an item that the other agent would happily give away. There is an inherent asymmetry in agents declaring items that they like most but not the items that they like least. To address this issue, we suppose agentscan select the item that they least like to give to the other agent. For instance, the policy $1\bar{1}21$ describes a protocol in which the first agent starts by picking their most preferred item, then picks their least preferred item to give to the second agent, the second agent then picks the most preferred of the two items that remain, and the first agent then gets the last remaining item. $\bar{1}$ means that agent 1 gives the item remaining that she likes least to agent 2.

n	ExpMinUtil egalitarian	ExpSumUtil utilitarian
1	1	1
2	12	12
3	122	121, $\bar{1}21$
4	1221, $1\bar{1}21$, $1\bar{2}22$, $\bar{1}211$	$1\bar{1}21$
5	12122, $1\bar{1}\bar{2}12$	$12\bar{2}12$, $\bar{1}2\bar{2}12$
6	$1\bar{2}\bar{1}121$, $\bar{1}21121$	$1\bar{1}2121$, $1\bar{1}2\bar{1}21$
7	$12\bar{1}\bar{1}212$	$1212\bar{2}12$, $1\bar{2}\bar{1}2\bar{2}12$, $\bar{1}212\bar{2}12$, $\bar{1}2\bar{1}2\bar{2}12$
8	$1\bar{2}\bar{1}\bar{1}2\bar{1}21$, $\bar{1}2112121$	$1\bar{1}212121$, ... $1\bar{1}2\bar{1}2\bar{1}21$

Table 7. Optimal policy for dividing n items with utility measured using Borda scoring assuming egalitarianism or utilitarianism and full independence between the two agents. Note that when computing the optimal policy, we consider all possible policies including those in which agents only pick items, and those in which agents only give items away.

In Table 7, we give the optimal policies assuming strategic behavior, and Borda scoring of utilities when agents can dispose of items as well as pick them. We again put policies into a canonical form in which agent 1 makes the first move. There is a symmetric policy in which we swap agent 1 with agent 2 throughout. We also ignore policies which result in the same division of items. For instance, a policy containing the moves $\bar{1}1$ is equivalent one containing $1\bar{1}$. Our canonical form has agents picking items before giving give them away. For example, a policy that ends with the moves $\bar{2}1$ gives the last two items to the first agent so is equivalent to one that ends with the moves 11. Our canonical form describes a policy by the lexicographically least equivalent policy supposing that 1 and 2 are ordered before $\bar{1}$ and $\bar{2}$.

We make some observations about the results. First, we can often increase social welfare by having agents declare items that they dislike. There are a few optimal policies in which agents only pick items that they like (e.g. for $n = 5$, one of the optimal egalitarian policies is 12122). However, in most cases, the optimal policy has agents declaring both items that they like and dislike. Second, when dividing 4 items between two agents, there is a policy, $1\bar{1}21$ that is optimal for both the egalitarian and utilitarian measures of social welfare. Third, unlike protocols in which agents pick just items that they like, there are often several different protocols which maximize social welfare.

8 Conclusions

We have studied a simple sequential allocation procedure where agents get to choose items according to a policy, and agents have simple additive utilities over items given by Borda, lexicographical or quasi-indifferent scores. We have computed optimal policies assuming both truthful and strategic behavior of the agents for both egalitarian and utilitarian measure of social welfare. We have also proved that with two agents, the subgame perfect Nash equilibrium is polynomial to compute by simply reversing the agents' preferences and the policy. On the other hand, with more than two agents, we proved that computing the subgame perfect Nash equilibrium is PSPACE-hard. There are many directions for future work. One direction would be to prove the conjectures about the optimal policies for maximizing social welfare assuming truthful or strategic behavior and Borda or lexicographical scoring. Another direction would be to determine if we can compute the subgame-perfect Nash equilibrium in polynomial time for a fixed number agents k where $k > 2$. More generally, when we want to allocate multiple indivisible goods, how can we design simple, elicitation free mechanisms that balance efficiency and strategy-proofness?

REFERENCES

[1] S. Bouveret and J. Lang. A general elicitation-free protocol for allocating indivisible goods. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 73–78. IJCAI/AAAI, 2011.

[2] S. Brams, D. Kilgour, and C. Klamler. The undercut procedure: an algorithm for the envy-free division of indivisible items. *Social Choice and Welfare*, pages 1–17, 2011. To appear.

[3] E. Budish and E. Cantillon. The Multi-Unit Assignment Problem: Theory and Evidence from Course Allocation at Harvard. *American Economic Review*, page (forthcoming), 2012.

[4] S.D. Kominers, M. Ruberry, and J. Ullman. Course allocation by proxy auction. In A. Saberi, editor, *6th International Workshop on Internet and Network Economics (WINE 2010)*, volume 6484 of *Lecture Notes in Computer Science*, pages 551–558. Springer, 2010.

[5] L.-G. Svensson. Strategy-proof allocation of indivisible goods. *Social Choice and Welfare*, 16(4):557–567, 1999.

Explanation of the robust additive preference model by even swap sequences

Christophe Labreuche[1] and **Nicolas Maudet**[2] and **Vincent Mousseau**[3] and **Wassila Ouerdane**[4]

Abstract. The even swap method is an interesting approach for identifying the best alternative among several options [5]. This constructive method is intuitively attracting: only two attributes are involved in even swaps, and utilities are never explicitely mentioned to the Decision Maker (DM). The aim of this paper is to investigate whether this approach can be generalized to robust preference relations and used to generate convincing explanations.

1 Introduction

The problem of constructing or providing convincing explanations to a Decision Maker (DM) in order to justify recommended decisions is a central concern for decision-aiding tools (see for instance [1, 8, 11]). This issue raises many questions. What is an explanation? How to construct an explanation? What is the information, beyond the utility functions, that is useful or necessary to get to construct a "good" explanation? Roughly speaking, the aim is to increase the user's acceptance of the recommended choice, by providing supporting evidence that this choice is justified [6].

One of the difficulties of this question lies on the fact that the relevant concept of an explanation may be different, depending on the decision problem at hand and on the targeted audience. Depending on the situations, explanations may be required to be precise (like a proof), or instead to be only convincing arguments. Also, the information that may be put forward to generate an explanation may greatly vary: a convincing explanation for the decision analyst may be impossible to understand for the DM, simply because their level of understanding of the problem differ. This problem is especially difficult in the context of multi-attribute models [9, 10], where different criteria are at stake, where the DM is not necessarily able to fully assess how important are criteria or to understand the way criteria interact. In such models, explaining the result is certainly not an easy task.

In this paper we shall concentrate on the basic additive utility model. This well-known (quantitative) model assumes independence among criteria, although of course different criteria may have different weights. In other words, no synergy (either positive or negative) occurs between the different criteria. In this model, the basic approach is to construct, by elicitation techniques, a so-called *utility function* which hopefully captures the DM preferences. A natural way to generate an explanation would thus be to use this constructed function and to justify the decision by exploiting this function. Unfortunately, this constructed function is often not very meaningful to the DM.

Within the additive model, an alternative interesting approach for identifying the best decision is based on so-called *even swaps* [5]. This is basically an elimination process based on trade-offs between *pairs* of attributes (hence the name *even swaps*). Broadly speaking, in such a swap, the DM changes the consequence (or score) of an alternative on one attribute, and compensates this change with one on another attribute, so that the new alternative is equally preferred in the end. What is the point of making such swaps? Suppose you want to compare two options but that none dominates (in the Pareto sense) the other one. By replacing one option with a different but equally preferred one, the hope is that dominance will occur. The process is thus repeated until dominance can be shown to hold, allowing to progressively eliminate options. An intuitive interpretation of this method is thus to see it as a scattered exploration of the iso-preference curve (the curve where lies, even virtually, the alternatives equally preferred) of the DM. This constructive method is quite intuitive as only two attributes are involved in even swaps, and utilities are never explicitly mentioned to the DM. The idea is then that it may constitute a good starting point to justify a recommendation without refereeing explicitely to the utility function or the model used to get the solution. However this approach suffers from a limitation: by requiring each new generated option to be equally preferred to the initial one, this makes the technique poorly adapted to the context of incomplete preferences where such equivalence virtually never hold.

When utility functions are only partially known, a conservative approach consists in relying on a robust (or necessary) preference relation. In words, the relation holds if *any* possible completion of the available preferential information yields the preferential statement. The aim of this paper is to investigate whether this approach can be generalized to robust preference relations and used to generate convincing explanations. In fact, the sequence of swaps obtained at the end of the process can be seen as the reasoning steps allowing to highlight why an alternative is the best choice.

The remainder of the paper is as follows. In the next section, we provide the necessary background notions and concepts that we shall use for formulating explanations. In Section 3 we describe what is an explanation based on sequence of preference-swaps and we address the problem of its length in Section 4. In general, we argue that the simplicity of an explanation is not directly captured by its length. However, we focus on a specific case where such a simplified view is possible, and provide first results. Section 5 discusses related works.

[1] Thales Research & Technology, 91767 Palaiseau Cedex, France, email: christophe.labreuche@thalesgroup.com
[2] LIP6, Université Paris-6, 75006 Paris Cedex 06, France, email: nicolas.maudet@lip6.fr
[3] LGI, Ecole Centrale de Paris, Chatenay Malabry, France, email: vincent.Mousseau@ecp.fr
[4] LGI, Ecole Centrale de Paris, Chatenay Malabry, France, email: wassila.ouerdane@ecp.fr

2 Background and basic definitions

We consider a finite set $N = \{1, \ldots, n\}$ of criteria. Each criterion $i \in N$ is described by an attribute X_i. We assume that all attributes are numerical. For discrete attributes, X_i represents integers. For continuous attributes, X_i is an interval (possibly infinite). Alternatives are considered as elements of the Cartesian product of the attributes: $X = X_1 \times \cdots \times X_n$.

2.1 Comparison of two alternatives with even swap sequences

We ground our work on the even swaps method [5] which relies on an additive utility function to compare multi-attribute alternatives $x = (x_1, x_2, \cdots, x_n)$ and $y = (y_1, y_2, \cdots, y_n)$:

$$x \succsim y \quad \Leftrightarrow \quad \sum_{i \in N} u_i(x_i) \geq \sum_{i \in N} u_i(y_i)$$

So as to choose the best alternative, this method does not require to fully elicit the marginal utility functions, but only a limited number of trade-offs between pairs of attributes (swaps). In other words, the DM does not have to explicitly define the preferences over the attributes in general or to make any assumption about the form of the utility function. More precisely, the DM changes the consequence (or score) of an alternative on one attribute, and compensates this change with a preferentially equal change on another attribute. This creates a new fictitious alternative, that is indifferent to the previous one, with revised consequences. We use this alternative to try to eliminate the other ones. The aim of this process is to carry out even swaps that make either alternatives dominated or attributes irrelevant. To get an intuitive understanding of the process, consider the following example, largely inspired from the original example provided in [5].

Example 1. *You need to rent an office for your business and you have the choice between four alternatives $\{x, y, t, z\}$. Such options are evaluated, as it is depicted in the Table 1, on four criteria $\{$commute (min), office services (A$\succ$B$\succ$C), size (m^2), cost (€)$\}$. Of course you want to minimize the cost and commute time, while you seek to maximize the quality of service and the size. The problem is to identify the best option.*

Table 1. Evaluation of available offices

	$Commute$ (min)	$Service$	$Size$ (m^2)	$Cost$ (€)
x	25	B	700	1700
y	20	C	500	1500
z	25	A	950	1900
t	30	C	700	1750
y'	25	C	550	1500
y''	25	B	500	1750

First, we can observe that x dominates t, so t can be removed from the list of considered offices. As no more dominance exists among the remaining alternatives, the method proceeds by constructing a first trade-off (a swap), starting for instance with the office y, by asking the following question: *"What increase δ in $Size$ would exactly compensate a loss of 5 min on $Commute$?"*

This defines a new alternative $y' = (25, C, 500 + \delta, 1500)$ that is considered by the DM indifferent to y. Suppose, for instance that $\delta = 50$, then y can be substituted by $y' = (25, C, 550, 1500)$ in the analysis. As dominance cannot be applied, new trade-offs are assessed to neutralize the criterion $Service$ using $Cost$ as reference. This can be done in two ways as follows:

- what maximal increase in $Cost$ would you be prepared to pay to go from C to B on $Service$ for y' ? if the answer is 250 €, so y' is indifferent to $y'' = (25, B, 550, 1750)$
- what minimal decrease in $Cost$ would you ask if we go from A to B on service for z? if the answer is 100 €, so z is indifferent to $z' = (25, B, 950, 1800)$

Obtaining two new fictitious alternatives we can again check the dominance among the set of alternatives. We can observe that y'' is dominated by x, therefore y can be dropped. We continue the process by alternating phases of dominance and construction of trade-offs until obtaining the best alternative.

Originally, this method was designed to select the best alternative, by eliciting progressively the necessary swaps. What we can observe from this small example is that following the reasoning steps of the even swaps process we can deduce an intuitive and simple manner to explain the result to the decision maker. In fact, such an explanation will involve statements like "*an increase of δ_i on criterion i is compensated by a decrease of δ_j on criterion j*", together with dominance analysis, rather than utility computation. That is, we can simply rely on the sequence of even swaps used to identify the best alternative rather than discuss the parameters and values of the multi-attributes models.

For instance, in our example the statement $x \succ y$ can be explained in the following way: "*an increase in Size from 500m^2 to 550m^2 compensates a degradation from 20mn to 25mn in Commute, therefore office y=(20,C,500,1500) is indifferent to office y'=(25,C,550,1500). Moreover an improvement from C to B on Service is compensated by an increase in Cost from 1500€ to 1750€, therefore office y'=(25,C,550,1500) is indifferent to office y''=(25,B,550,1750)". Now observe that x is at least as good as y'' on all attributes, then x is preferred to y''. As y'' is indifferent to y, x is preferred to y.*"

Note that even if the utility functions u_i are known precisely and have been elicited with a technique different from the even swaps process, then it is possible from these utility functions to construct a sequence of even swaps that allows to show why x is preferred to y (assuming $x \succsim y$ of course). This sequence can be used as an explanation of why $x \succsim y$.

In a sense, the even swaps method already deals with some sort of incomplete preferences, as it does not require the full knowledge of the value function. The trick of the method is precisely to explore certain alternatives which stand on the same isopreference curve of the DM, until dominance occurs. However, at each step, the DM is required to answer *equivalence queries*, which may be difficult in practice. Consider again the question *"What increase in $Size$ would exactly compensate a loss of 5 min on $Commute$?"*. To such a question, the DM may be more comfortable to reply: *"I don't know, but 100 additional square meters would certainly compensate this additional commute time"*. Or, on the other hand, *"20 square meters are certainly not enough to compensate"*. By doing so however, the DM does not allow a proper even swap to occur. Instead of creating a fictitious alternative with the same utility, such statements generates a mere inequality constraint in the preferential information available. For instance, the last statement would create a new alternative $y''' = (25, C, 500 + 20, 1500)$, and it would be known that $y \succ y'''$.

In the remainder of this paper, we investigate whether the swap principle can be extended in order to allow such a wider range of preference statements.

2.2 Robust relation with the additive utility model

In what follows, we assume the decision-maker (DM) provides us some Preferential Information (PI) denoted by $\mathcal{P}$. We may consider the following types of PI:

- The most classical one is a comparison of two alternatives x and y in X, which can take different forms $x \trianglerighteq y$ (x is at least as good as y), $x \rhd y$ (x is strictly preferred to y) or $x \equiv y$ (x is indifferent to y)
- When X_i is an interval, one may also express the existence of saturation threshold. Value s_i^+ (resp. s_i^-) is an upper (resp. lower) saturation threshold on attribute i if for all $x_{-i} \in X_{-i}$ and all $x_i \in X_i$ with $x_i \geq s_i^+$ (resp. $x_i \leq s_i^-$), $(x_i, x_{-i}) \equiv (s_i^+, x_{-i})$ (resp. $(x_i, x_{-i}) \equiv (s_i^-, x_{-i})$).
- When X_i is an interval, the DM may also express that the preference over attribute X_i is concave or convex.

Given two alternatives $z, z' \in X$, we need to determine whether z is necessarily preferred (resp. similar) to z' given the previous PI [3, 4].

For each $i \in N$, we define $\widehat{V}_i$ as the set of values on attributes X_i appearing in the PI (i.e. the union of $\{x_i, y_i\}$ for all $[x \trianglerighteq y]$, $[x \rhd y]$ or $[x \equiv y]$ in the PI $\mathcal{P}$, of the thresholds s_i^+, s_i^- of the PI). Moreover, we define $V_i = \widehat{V}_i \cup \{z_i, z_i'\}$. The elements of V_i are denoted by $v_i^1 < v_i^2 < \cdots < v_i^{p_i}$, where $p_i = |V_i|$. The unknown variables of the model are the utility $u_i(v_i^1), u_i(v_i^2), \ldots, u_i(v_i^{p_i})$ of these points.

Throughout this paper, we will assume that the utility functions are non-decreasing, so that

$$\forall i \in N \quad u_i(v_i^1) \leq u_i(v_i^2) \leq \cdots \leq u_i(v_i^{p_i}). \tag{1}$$

Concerning the PI on the comparison of two alternatives, we have the following constraints:

$$\text{If } [x \trianglerighteq y] \in \mathcal{P}, \text{ then} \quad \sum_{i \in N} u_i(x_i) \geq \sum_{i \in N} u_i(y_i) \tag{2}$$

$$\text{If } [x \rhd y] \in \mathcal{P}, \text{ then} \quad \sum_{i \in N} u_i(x_i) > \sum_{i \in N} u_i(y_i) \tag{3}$$

$$\text{If } [x \equiv y] \in \mathcal{P}, \text{ then} \quad \sum_{i \in N} u_i(x_i) = \sum_{i \in N} u_i(y_i) \tag{4}$$

Now if the DM expresses an upper-saturation level on attribute i, the following constraint is added:

$$\forall v_i \in V_i \text{ with } v_i > s_i^+ \quad u_i(v_i) = u_i(s_i^+), \tag{5}$$

and if the DM expresses a lower-saturation level on attribute i, the following constraint is added:

$$\forall v_i \in V_i \text{ with } v_i < s_i^- \quad u_i(v_i) = u_i(s_i^-). \tag{6}$$

Finally, if the DM expresses that the utility function on criterion i is concave, then the following constraint is added:

$$\forall j \in \{3, \ldots, p_i\}$$

$$u_i(v_i^j) \leq u_i(v_i^{j-2}) + (u_i(v_i^{j-1}) - u_i(v_i^{j-2})) \frac{v_i^j - v_i^{j-2}}{v_i^{j-1} - v_i^{j-2}} \tag{7}$$

and if the DM expresses that the utility function on criterion i is convex, then the following constraint is added:

$$\forall j \in \{3, \ldots, p_i\}$$

$$u_i(v_i^j) \geq u_i(v_i^{j-2}) + (u_i(v_i^{j-1}) - u_i(v_i^{j-2})) \frac{v_i^j - v_i^{j-2}}{v_i^{j-1} - v_i^{j-2}} \tag{8}$$

Example 2. *(1, ctd.) For attribute X_1, we have $\widehat{V}_1 = \{20, 25, 30\}$. For attribute X_3, we have $\widehat{V}_3 = \{500, 550, 700, 950\}$. We have for instance $[y \rhd y''']$. Finally, the DM could express an upper-saturation level by stating: "Frankly, I don't need more than 100 square meters."*

This provides a set of constraints resulting from the DM statements. Typically, a number of utilities will be compatible with these constraints. In order to compare the alternatives, we compute :

$$\underline{M} := \min \sum_{i \in N} (u_i(z_i) - u_i(z_i'))$$
$$\text{under (1) – (8)}$$

and

$$\overline{M} := \max \sum_{i \in N} (u_i(z_i) - u_i(z_i'))$$
$$\text{under (1) – (8)}$$

Definition 1 (see [3, 4])**.** *We say that z is* necessarily at least as good as z' *(noted $z \succsim_N z'$) if $\underline{M} \geq 0$. Likewise, z' is* necessarily at least as good as z *(noted $z' \succsim_N z$) if $\overline{M} \leq 0$.*

We say that z is necessarily preferred *to z' (noted $z \succ_N z'$) if $\underline{M} > 0$. Likewise, z' is* necessarily preferred *to z (noted $z' \succ_N z$) if $\overline{M} < 0$, and z is* necessarily similar *to z' (noted $z \sim_N z'$) if $\underline{M} = \overline{M} = 0$.*

Note that it is very unlikely that the necessarily similar relation holds when the PI is incomplete.

2.3 Objective of this paper

We denote by $\geq_{\text{Pareto}}$ the Pareto ordering on X. In Section 2.1, as we have seen, when the utilities are completely fixed, showing that an option z is at least as good as z' (denoted by $z \succsim z'$) consists in exhibiting a sequence $z[1], z[2], \ldots, z[q]$ in X with $z[1] = z$ such that

$$z[1] \sim z[2] \sim \ldots \sim z[q] \geq_{\text{Pareto}} z' \tag{9}$$

(where $\sim$ means indifference) and $z[l] \sim z[l+1]$ corresponds to an even-swap.

The generalization of this approach to robust preference relation is not simple. The main reason is that the robust indifference relation $\sim_N$ almost never occur in practice. To circumvent this issue, we propose to generalize (9) in two different ways.

First of all, the equivalence relation used in even-swaps needs to be relaxed into a preference relation.

Definition 2. *We say that $z[l] \succsim_N z[l+1]$ is a* preference-swap *if there exists $i, j \in N$ such that $z[l]_i > z[l+1]_i$, $z[l]_j < z[l+1]_j$ and $z[l]_k > z[l+1]_k$ for all $k \in N \setminus \{i, j\}$.*

Secondly, one can generalize even-swap to trade-offs among coalition of more than two criteria.

Definition 3. *We say that $z[l] \succsim_N z[l+1]$ is a* preference-swap of order p *(with $p \in \{2, \ldots, n\}$ if there exists $A \subset N$ with $|A| = n-p$, $\forall i \in A$, $x_i = y_i$ and $\forall i \in N \setminus A$, $x_i \neq y_i$.*

Explanations in our context will thus be sequences generalizing (9) and in particular consisting of preference swaps of order 2 or higher. This is described more formally in the next section.

3 Explanations based on preference-swaps

We introduce the following sets:

- Δ_0 is the set of pairs (x, y) in $X \times X$ such that $[x \trianglerighteq y] \in \mathcal{P}$. In other terms, it is the set of comparative preferential information given by the decision maker.
- Δ_1 is the set of pairs (x, y) in $X \times X$ such that there exists $(x', y') \in \Delta_0$ with $x \geq_{\text{Pareto}} x'$ and $y' \geq_{\text{Pareto}} y$.
- For $p \in \{2, \ldots, n\}$, Δ_p is the set of pairs of alternatives (x, y) in $X \times X$ such that $x \succsim_N y$ is a preference-swap of order p.

$\Delta_0, \Delta_1, \ldots, \Delta_n$ are in increasing complexity to understand them.

Let $\Delta := \Delta_0 \cup \Delta_1 \cup \ldots \cup \Delta_n$. Clearly Δ is the set of pairs satisfying the binary relation $\succsim_N$.

Definition 4. *An* explanation *of $z \succsim_N z'$ is a sequence $z[1], z[2], \ldots, z[q]$ in X with $z[1] = z$ and $z[q] = z'$ such that $(z[k], z[k+1]) \in \Delta$ for all $k \in \{1, \ldots, q-1\}$. Let Ex denote the set of explanations.*

In order to define orderings over explanations, we introduce the concept of complexity of an explanation.

Definition 5. *For $(z[1], \ldots, z[q]) \in Ex$, The* complexity *of $(z[1], \ldots, z[q]) \in Ex$ is*

$$comp(z[1], \ldots, z[q]) := \Big(C_{(z[1],\ldots,z[q])}(0), C_{(z[1],\ldots,z[q])}(1), \ldots, C_{(z[1],\ldots,z[q])}(n)\Big)$$

where

$$C_{(z[1],\ldots,z[q])}(k) = \Big|\{j \in \{1, \ldots, q-1\},\ (z[j], z[j+1]) \in \Delta_k\}\Big|.$$

We now define several possible orderings over explanations (see Definitions 6 and 7).

Definition 6. *For $(z[1], \ldots, z[q]), (t[1], \ldots, t[q']) \in Ex$, $(z[1], \ldots, z[q]) \rhd_{Ex} (t[1], \ldots, t[q'])$ iff $comp(z[1], \ldots, z[q]) \succ_{\text{lex}} comp(t[1], \ldots, t[q'])$, where $\succ_{\text{lex}}$ is the lexicographic ordering. $(a_0, \ldots, a_n) \succ_{\text{lex}} (b_0, \ldots, b_n)$ if there exists $i \in \{0, \ldots, n\}$ such that $a_i > b_i$ and $a_j = b_j$ for all $j \in \{0, \ldots, n\}$ with $j > i$.*

Definition 7. *For $(z[1], \ldots, z[q]), (t[1], \ldots, t[q']) \in Ex$,*

$$\begin{aligned}
&(z[1], \ldots, z[q]) \rhd'_{Ex} (t[1], \ldots, t[q']) \quad \text{iff} \\
&\Big(C_{(z[1],\ldots,z[q])}(0) \cup C_{(z[1],\ldots,z[q])}(1) \cup C_{(z[1],\ldots,z[q])}(2), \\
&\quad C_{(z[1],\ldots,z[q])}(3), \ldots, C_{(z[1],\ldots,z[q])}(n)\Big) \\
&\succ_{\text{lex}} \Big(C_{(z'[1],\ldots,z'[q])}(0) \cup C_{(z'[1],\ldots,z'[q])}(1) \cup C_{(z'[1],\ldots,z'[q])}(2), \\
&\quad C_{(z'[1],\ldots,z'[q])}(3), \ldots, C_{(z'[1],\ldots,z'[q])}(n)\Big).
\end{aligned}$$

According to Definition 6, Δ_0 is the less complex elements of Δ, Δ_1 are the second less complex elements, ..., and Δ_n are the most complex elements. On the other hand, in Definition 7, the three sets Δ_0, Δ_1 and Δ_2 are of the same complexity and are combined.

We then look for a minimal explanation in the sense of $\rhd_{Ex}$ or $\rhd'_{Ex}$.

Example 3. *Consider the following PI on a set of 3 attributes*

$$(10, 100, 1000) \trianglerighteq (20, 80, 900) \quad (10)$$
$$(20, 70, 900) \trianglerighteq (15, 100, 1000) \quad (11)$$
$$(0, 85, 700) \trianglerighteq (30, 90, 500) \quad (12)$$
$$(30, 80, 500) \trianglerighteq (0, 85, 600) \quad (13)$$

We wish to compare $z = (10, 70, 700)$ with $z' = (15, 90, 600)$. From (10) and (11) we get

$$(10, 70, 900) \succsim_N (15, 80, 900)$$

and thus from the independence property of the model:

$$(10, 70, 700) \succsim_N (15, 80, 700). \quad (14)$$

From (12) and (13) we get

$$(30, 80, 700) \succsim_N (30, 90, 600)$$

and thus from the independence property of the model:

$$(15, 80, 700) \succsim_N (15, 90, 600). \quad (15)$$

From (14) and (15), we get the sequence

$$z = (10, 70, 700) \succsim_N (15, 80, 700) \succsim_N (15, 90, 600) = z'.$$

Hence

$$comp((10, 70, 700), (15, 80, 700), (15, 90, 600)) = (0, 0, 2, 0)$$
$$comp((10, 70, 700), (15, 90, 600)) = (0, 0, 0, 1)$$

and

$$comp((10, 70, 700), (15, 80, 700), (15, 90, 600)) \succ_{\text{lex}} comp((10, 70, 700), (15, 90, 600)).$$

The explanation $((10, 70, 700), (15, 80, 700), (15, 90, 600))$ is simpler than $((10, 70, 700), (15, 90, 600))$ in the sense of $\rhd_{Ex}$ or $\rhd'_{Ex}$.

This shows that the simplicity of an explanation is not directly captured by the length of the sequence. Short sequences involving preference-swaps of high order may not be desirable. However, if we restrict our attention to preference-swaps of order 2, the length of the sequence becomes very important to consider.

4 On the length of preference-swap sequences

We consider an explanation of $z \succsim_N z'$, that is a sequence $z[1], z[2], \ldots, z[q]$ (see Definition 4).

Definition 8. *The* length *of an explanation $(z[1], \ldots, z[q]) \in Ex$ is its number of elements, that is q.*

Furthermore, we assume that $(z[k], z[k+1]) \in \Delta_1 \cup \Delta_2$ (Pareto ordering and preference-swap of order 2) for all $k \in \{1, \ldots, q-1\}$.

In the case of sequences of even-swaps, it is easy to see that the length of the sequence is at most n. Let us show in an example that this is not the case with sequences of preference-swaps.

Example 4. *Let us consider four criteria and the following PI:*

$$(1,0,\cdot,\cdot) \trianglerighteq (0,1,\cdot,\cdot) \quad (16)$$
$$(0,\cdot,1,\cdot) \trianglerighteq (1,\cdot,0,\cdot) \quad (17)$$
$$(\cdot,1,\cdot,0) \trianglerighteq (\cdot,0,\cdot,1) \quad (18)$$

where '$\cdot$' means that the value on this attribute does not matter provided that the alternatives on the left hand side and on the right hand side have the same value.

Consider now two alternatives $(1,0,1,0)$ *and* $(0,1,0,1)$*. It can be readily seen that* $(1,0,1,0) \succsim_N (0,1,0,1)$*. One can obtain the following sequence of only 5 comparisons from the PI* (Δ_0)*:*

$$(1,0,1,0) \underbrace{\succsim_N}_{\text{from (16)}} (0,1,1,0) \underbrace{\succsim_N}_{\text{from (17)}} (1,1,0,0)$$
$$\underbrace{\succsim_N}_{\text{from (18)}} (1,0,0,1) \underbrace{\succsim_N}_{\text{from (16)}} (0,1,0,1)$$

We restrict ourself in the rest of this paper to preference swaps of order 2. Can we get an upper bound on the length q?

4.1 Unboundedness of the length of the sequence

We start with a negative result. If we make no assumption on the values of the attributes taken by the alternatives in the PI, the length q is unbounded. This is shown by the following lemma when $n \geq 3$.

Lemma 1. *Consider* $n \geq 3$*. Let* $z, z' \in \mathbb{R}^N$ *where* $z_i \neq z_i$ *for at least three attributes* i_1, i_2, i_3 *with* $z_{i_1} < z'_{i_1}$, $z_{i_2} > z'_{i_2}$ *and* $z_{i_3} > z'_{i_3}$*. Assume that* $[\min(z_i, z'_i), \max(z_i, z'_i)] \subseteq X_i$ *for all* $i \in \{i_1, i_2, i_3\}$*. Then for every* $k \in \mathbb{N}^*$*, there exists some PI such that* $z \succsim_N z'$ *and the minimal length of the explanation in* $\Delta_0 \cup \Delta_1 \cup \Delta_2$ *is at least* k*.*

Note that we can also add Δ_1 on top of Δ_2 in the previous lemma.

Proof : Let $n \geq 3$, $k \in \mathbb{N}^*$ and $p = \lfloor \frac{k}{2} \rfloor$. Without loss of generality, take $i_1 = 1$, $i_2 = 2$, $i_3 = 3$, $z_1 = 0$, $z_2 = 0$, $z_3 = 0$, $z'_1 = 1$, $z'_2 = -1$ and $z'_3 = -1$. Assume that $X_1 \supseteq [0,1]$, $X_2 \supseteq [-1,0]$ and $X_3 \supseteq [-1,0]$. Consider the following PI:

$$\forall j \in \{0,\dots,p-1\} \quad \Big(\frac{2j}{2p}, -\frac{j}{p}, -\frac{j}{p}, z_{-123}\Big) \trianglerighteq \Big(\frac{2j+1}{2p}, -\frac{j+1}{p}, -\frac{j}{p}, z_{-123}\Big)$$
$$\forall j \in \{0,\dots,p-1\} \quad \Big(\frac{2j+1}{2p}, -\frac{j+1}{p}, -\frac{j}{p}, z_{-123}\Big) \trianglerighteq \Big(\frac{2j+2}{2p}, -\frac{j+1}{p}, -\frac{j+1}{p}, z_{-123}\Big)$$
$$\forall i \in \{4,\dots,n\} \quad (z'_1, \dots z'_{i-1}, z_i, z_{i+1}, \dots, z_n) \equiv (z'_1, \dots z'_{i-1}, z'_i, z_{i+1}, \dots, z_n)$$

With this PI, we clearly obtain $z \succsim_N z'$ and the sequence

$$\begin{aligned} z =& (0,0,0,z_{-123}) \succsim_N \Big(\frac{1}{2p}, -\frac{1}{p}, 0, z_{-123}\Big) \\ &\succsim_N \Big(\frac{2}{2p}, -\frac{1}{p}, -\frac{1}{p}, z_{-123}\Big) \succsim_N \cdots \\ &\succsim_N \Big(\frac{2p-2}{2p}, -\frac{p-1}{p}, -\frac{p-1}{p}, z_{-123}\Big) \\ &\succsim_N \Big(\frac{2p-1}{2p}, -1, -\frac{p-1}{p}, z_{-123}\Big) \succsim_N (1,-1,-1,z_{-123}) \\ &\equiv_N (z'_1, \dots, z'_4, z_5 \dots, z_n) \equiv_N \cdots \equiv_N z'. \end{aligned}$$

This sequence is of length $(2\,p+1) + (n-3) \geq k + (n-3)$.

We obtain the following constraints from the PI

$$\forall j \in \{0,\dots,p-1\} \quad u_1\Big(\frac{2j}{2p}\Big) + u_2\Big(-\frac{j}{p}\Big) \geq u_1\Big(\frac{2j+1}{2p}\Big) + u_2\Big(-\frac{j+1}{p}\Big)$$
$$\forall j \in \{0,\dots,p-1\} \quad u_1\Big(\frac{2j+1}{2p}\Big) + u_3\Big(-\frac{j}{p}\Big) \geq u_1\Big(\frac{2j+2}{2p}\Big) + u_3\Big(-\frac{j+1}{p}\Big)$$
$$\forall j \in \{0,\dots,p-1\} \quad u_1\Big(\frac{2j}{2p}\Big) \leq u_1\Big(\frac{2j+1}{2p}\Big) \leq u_1\Big(\frac{2j+2}{2p}\Big)$$
$$\forall j \in \{0,\dots,p-1\} \quad u_2\Big(-\frac{j}{p}\Big) \geq u_2\Big(-\frac{j+1}{p}\Big) \text{ and } u_3\Big(-\frac{j}{p}\Big) \geq u_3\Big(-\frac{j+1}{p}\Big)$$
$$\forall i \in \{4,\dots,n\} \quad u_i(z_i) = u_i(z'_i)$$

As the alternatives appearing in the PI use different values on the attributes, the necessary relation is composed of $\trianglerighteq$ with the Pareto ordering. From this, one cannot skip any comparison in the sequence. Hence there is no explanation sequence of preference-swap of order 2 strictly shorter than $2\,p+1$. Note that the $n-3$ comparisons for the explanation of $(1,-1,-1,z_{-123}) \equiv_N z'$ can be done with only one Pareto comparison (depending on the values of z and z'). ■

The only hope to have an upper bound on the length q is thus to restrict the number of values that the PI can take on each attribute.

4.2 Some solution to bound the length of sequences

We take *binary alternatives* as an extreme case. Assume now that there exists two values on each attribute (denoted by 0 and 1) such that the alternatives appearing in the PI belong to $\{0,1\}^N \subseteq X$.

Lemma 2. *Let* $w_i := u_i(1) - u_i(0)$*. For every* $A, B \subseteq N$*, we have*

$$(1_A, 0_{-A}) \succsim_N (1_B, 0_{-B}) \iff \sum_{i \in A \setminus B} w_i \geq \sum_{i \in B \setminus A} w_i \text{ for all } w \text{ compatible with the PI.}$$

Proof :

$$\begin{aligned} &(1_A, 0_{-A}) \succsim_N (1_B, 0_{-B}) \\ \iff &\sum_{i \in A} u_i(1) + \sum_{i \in N \setminus A} u_i(0) \geq \sum_{i \in B} u_i(1) + \sum_{i \in N \setminus B} u_i(0) \\ \iff &\sum_{i \in A \setminus B} u_i(1) + \sum_{i \in B \setminus A} u_i(0) \geq \sum_{i \in B \setminus A} u_i(1) + \sum_{i \in A \setminus B} u_i(0) \\ \iff &\sum_{i \in A \setminus B} w_i \geq \sum_{i \in B \setminus A} w_i \end{aligned}$$

■

Let $\mathcal{W}$ be the set of $w \in \mathbb{R}_+^N$ s.t. $\sum_{i\in A\setminus B} w_i \geq \sum_{i\in B\setminus A} w_i$ for all PI $(1_A, 0_{-A}) \trianglerighteq (1_B, 0_{-B})$.

Lemma 3. *For $x, y \in \{0,1\}^N$. If $(x,y) \in \Delta_2$, there exist i and j are such that $x_i = 1$, $y_i = 0$, $x_j = 0$, $y_j = 1$ and $x_k = y_k$ for all $k \in N \setminus \{i,j\}$. Then we have*

$$(x,y) \in \Delta_2 \iff w_i \geq w_j \text{ for all } w \in \mathcal{W}.$$

Proof : Clear from Lemma 2. ■

From Lemma 3, it is apparent that the length of sequences using only terms in Δ_2 is bounded. If the length of the sequence is large enough, one necessarily finds relations of the form $w_i \geq w_j$ and $w_j \geq w_k$ in the sequence. Clearly, these two relations can be replaced by the comparison $w_i \geq w_k$, by transitivity. We believe that this allows to keep the length of sequences under a given value (not provided in this paper). Let us illustrate this intuition on an example.

Example 5 (Example 4, ctd.)**.** *By Lemma 3, (16) is equivalent to*

$$w_1 \geq w_2, \tag{19}$$

(17) is equivalent to

$$w_3 \geq w_1, \tag{20}$$

and (18) is equivalent to

$$w_2 \geq w_4. \tag{21}$$

Consider now the two alternatives $(1,0,1,0)$ and $(0,1,0,1)$ (note that $(1,0,1,0) \succsim_N (0,1,0,1)$). The explanation in Example 4 may seem simple for the user since it is based only on PI. However, it uses the first example (16) twice, giving the feeling that it is circular.

Actually, another explanation can be constructed. First of all, let us note that adding (19) and (20), we obtain

$$w_3 \geq w_2, \tag{22}$$

and adding (19) and (21), we get

$$w_1 \geq w_4. \tag{23}$$

Hence the following explanation is reached:

$$(1,0,1,0) \underbrace{\succsim_N}_{\text{from (22)}} (0,0,1,1) \underbrace{\succsim_N}_{\text{from (23)}} (0,1,0,1).$$

The length of this sequence is only 3 and it is composed of comparisons in Δ_2. This explanation is more direct than the previous one, and seams better for the user. It is better than the explanation of Example 5 in the sense of $\rhd'_{Ex}$.

In Example 5, the reduction of the length of the explanation is based on the trick described just before this example.

5 Related works

The idea of even swap can be found in negotiation in multi-agent systems [2]. There is a difference between the *concession* and *trade-off*. In a concession, the agent give up on something and it is ready to accept an offer which overall utility is smaller than a previous offer. By contrast, in a trade-off analysis, the agent explores the set of options that yield the same overall utility (a level-set), and one looks at balancing utilities among the criteria (while remaining on the same level curve) so that another agent will be better satisfied. This implies decreasing the expectation on a criterion while increasing the expectation on another criterion. For instance, a customer may be ready to pay more if the item is delivered faster. The main idea of [2] is to instantiate the idea of trade-off: the proposer at an iteration of the protocol shall propose, among all options that yield a given satisfaction to it, the option that is *best* for the other(s) agent(s). The main idea of the paper is to represent the preferences of an agent by a *similarity measure* to the last proposal made by this agent.

Another potentially fruitful connection to explore is with planning problems, where the objective is to find how to sequentially apply different operators so as to attain an objective state from an initial state. Preference-swaps can be seen as operators, and the objective state to reach is an alternative exhibiting the desired dominance.

6 Conclusion

This paper investigates the problem of providing minimal explanation by relying on an extension of the even swaps. A first contribution of the paper is to set up the framework allowing to generalize such an approach to more general preference statements, and to be used to generate convincing explanation (on the basis on so-called preference swaps). Another natural extension is to trade-off involving more than two criteria, although this may quickly be difficult to handle for the DM. The first result put forward in this paper is negative: it states that in the absence of any restriction on the size of the domain considered for the value of attributes (in particular when such a domain is an interval over the reals), the sequence of preference-swaps may not be bounded. This challenges the practical use of this technique in this case. It is thus natural to consider restricted domains: we show that in binary domains positive results (bounded sequence) can hold, and sketch possible solutions to reduce the length of explanations.

REFERENCES

[1] G. Carenini and J.D. Moore, 'Generating and evaluating evaluative arguments', *AIJ*, **170**, 925–952, (2006).

[2] P. Faratin, C. Sierra, and N.R. Jennings, 'Using similarity criteria to make issue trade-offs in automated negotiations', *AIJ*, **142**, 205–237, (2002).

[3] S. Greco, B. Matarazzo, and R. Słowinski, 'Ordinal regression revisited: Multiple criteria ranking with a set of additive value functions', *European Journal of Operational Research*, **191**, 416–436, (2008).

[4] S. Greco, R. Słowinski, J. Figueira, and V. Mousseau, 'Robust ordinal regression', in *Trends in Multiple Criteria Decision Analysis*, 241–284, Springer Verlag, (2010).

[5] J. Hammond, R. Keeney, and H. Raiffa, 'Even Swaps: a rational method for making trade-offs', *Harvard Business Review*, 137–149, (1998).

[6] J. L. Herlocker, J. A. Konstan, and J. Riedl, 'Explaining collaborative filtering recommendations', in *CSCW*, pp. 241–250, (2000).

[7] D.A. Klein, *Decision analytic intelligent systems: automated explanation and knowledge acquisition*, Lawrence Erlbaum Associates, 1994.

[8] Ch. Labreuche, 'A general framework for explaining the results of a multi-attribute preference model', *AIJ*, **175**, 1410–1448, (2011).

[9] Ch. Labreuche, N. Maudet, and W. Ouerdane, 'Justifying dominating options when preferences are incomplete', in *Proceedings of ECAI-12*, Montpellier, FR, (2012). To appear.

[10] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, 'MoviExplain: a recommender system with explanations', in *Proceedings of the third ACM conference on Recommender systems (RecSys'09)*, pp. 317–320, New York, NY, USA, (2009). ACM.

Resistance to bribery when aggregating soft constraints, and exploitation of bribery cost schemes in preference compilation and optimization

Alberto Maran[1] and **Maria Silvia Pini**[2] and **Francesca Rossi**[3] and **Kristen Brent Venable**[4]

Abstract. We consider a multi-agent scenario, where the preferences of several agents are modelled via soft constraint problems and need to be aggregated to compute a single "socially optimal" solution. We study the resistance of various ways to compute such a solution to attempts to influence the result, such as those based on the notion of bribery. In doing this, we link the cost to bribe an agent to the effort needed by the agent to make a certain solution optimal, by only changing preferences associated to parts of the solution. This leads to the definition of four notions of distance from optimality of a solution in a soft constraint problem. The notions differ on the amount of information considered when evaluating the effort. We then show how to pass from such distance notions to suitable linearizations of the solution preference ordering, which can be exploited in the context of computing sets of k best solutions. We also show how the considered distances can be used in preference compilation tasks, such as when encoding elicited solution preferences in the constraint structure.

1 Introduction

Often agents need to cooperate to take a collective decision. By doing this, the decision can be better than what they would have chosen, had they reasoned in isolation. Examples are collections of experts that have suggestions on what to do, which are then aggregated to obtain a single suggestion. Such experts could be, for example, classifiers in machine learning tasks, or web page rankers in web search. We model such scenarios via a collection of agents that express their preferences over a common set of solutions to a problem. We assume that such preferences are described by soft constraints [14], more precisely either fuzzy and weighted constraints. Agents' preferences are aggregated to compute a single "socially optimal" solution. To model this process, we consider some voting rules [1]. Although voting rules have been defined and studied in the context of political elections, they do exactly what we want: aggregating individual's preferences into a single collective "winner".

We study the resistance of this setting, considering different voting rules, to external or internal attempts to influence the result. This happens often in political elections, but it could occur also in other scenarios. For example, when voting to choose a date for a meeting [10], if one participant sees how the others have voted (and thus can compute the result by considering these votes and her true vote), she could vote in a strategic way (that is, differently to what her true vote would say) in order to get a better result for her. This example is an instance of the so-called manipulation, where one or more agents may misreport their votes in order to get a better solution. Another kind of attempt may come from an external agent, usually called the "briber", who has a preferred solution, and tries to get that solution as the result of the voting process, by paying some agents to vote in a certain way, and by doing this while staying within its budget. In defining bribing scenarios, it is thus necessary to decide what the briber can ask an agent to do (for example, just making a certain candidate optimal, or changing more of its preference ordering) and how costly it is for the briber to submit a certain request. The cost usually represents the effort the agent has to make to satisfy the briber's request.

Classical results on voting theory tell us that every voting rule can be influenced by such attempts [1]. However, for some voting rules, it may be computationally difficult for the manipulators, or the briber, to understand how to design the attempt. Such rules are then said to be resistant to these attempts [2, 9].

In this paper we study whether our soft constraint aggregation scenarios are resistant to bribery. Resistance to manipulation has been studied already, for example in [6]. We consider two main approaches to aggregate the preferences: a sequential one, where agents vote on each variable at a time, and a one-step approach, where agents vote just once on entire solutions. We then define five cost schemes to compute the cost of satisfying a briber's request. We find out that the one-step approach (which uses the Plurality voting rule) is not resistant to bribery. On the other hand, the sequential approaches (which are based on voting rules such as Plurality, Approval, and Borda), are all resistant to bribery for five out of five cost schemes. This is very interesting, since the sequential approaches are also better in terms of complexity of determining the collective solution. The cost schemes used in the bribery setting can be seen as a measure of the effort for an agent to respond to a briber's request. If the request is related to making a certain solution, say A, optimal (which means voting for it, if we use Plurality), then the cost can be considered a measure of how much the agent needs to change in its soft constraint problem in order to make A optimal. By following this line of reasoning, we exploit some of the cost schemes used for bribery to define four notions of distance from optimality of a solution in a soft constraint problem. We then show how to pass from such distance notions to corresponding linearizations of the solution preference ordering, which can be exploited in the context of com-

[1] Department of Mathematics, University of Padova, Italy, email: amaran@studenti.math.unipd.it
[2] Department of Information Engineering, University of Padova, Italy, email: pini@dei.unipd.it
[3] Department of Mathematics, University of Padova, Italy, email: frossi@math.unipd.it
[4] Department of Mathematics, University of Padova, Italy, email: kvenable@math.unipd.it

puting sets of k best solutions. The computational complexity results obtained for the bribery problem can then be useful to determine how expensive it is to compute the top k solutions.

We also show how these distances can be used in preference compilation tasks, such as when encoding optimal solutions in the constraint structure. Making a solutions optimal according to a certain distance notion has the same computational complexity as determining the bribery cost with the corresponding cost scheme.

In the following, the formal proofs of some results have been omitted for lack of space.

2 Background

Soft constraints. A soft constraint [14] involves a set of variables and associates a value from a (partially ordered) set to each instantiation of its variables. Such a value is taken from a c-semiring, which is defined by $\langle A, +, \times, 0, 1\rangle$, where A is the set of preference values, $+$ induces an ordering over A (where $a \leq b$ iff $a + b = b$), $\times$ is used to combine preference values, and 0 and 1 are respectively the worst and best element. 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, C is a set of soft constraints (each one involving a subset of V), A is the set of preference values.

An instance of the SCSP framework is obtained by choosing a specific c-semiring. For instance, a classical CSP [14] is just an SCSP where the c-semiring is $S_{CSP} = \langle\{false, true\}, \vee, \wedge, false, true\rangle$. By choosing $S_{FCSP} = \langle[0, 1], max, min, 0, 1\rangle$ instead it means that preferences are in [0,1] and we want to maximize the minimum preference. This is the setting of fuzzy CSPs (FCSPs) [14], that we will use in the examples of this paper. In the paper we will also consider the setting of weighted CSPs (WCSPs), where the c-semiring is $S_{WCSP} = \langle R^+, min, +, +\infty, 0\rangle$, which means that preferences are interpreted as costs from 0 to $+\infty$, and that we want to minimize the sum of the costs.

Figure 1 shows the constraint graph of an FCSP where $V = \{x, y, z\}$, $D = \{a, b\}$ and $C = \{c_x, c_y, c_z, c_{xy}, c_{yz}\}$. Each node models a variable and each arc models a binary constraint, while unary constraints define variables' domains. For example, c_y associates preference 0.4 to $y = a$ and 0.7 to $y = b$. Default constraints such as c_x and c_z will often be omitted in the following examples.

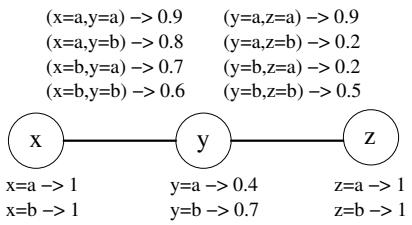


Figure 1. A tree-shaped FCSP.

Solving an SCSP means finding some information about the ordering induced by the constraints over the set of all complete variable assignments. In the case of FCSPs and WSCSPs, such an ordering is a total order with ties. In the example above, the induced ordering has $(x = a, y = b, z = b)$ and $(x = b, y = b, z = b)$ at the top, with preference 0.5, $(x = a, y = a, z = a)$ and $(x = b, y = a, z = a)$ just below with 0.4, and all others tied at the bottom with preference 0.2. An optimal solution, say s, of an SCSP is then a complete assignment with an undominated preference (thus $(x = a, y = b, z = b)$ or $(x = b, y = b, z = b)$ in this example). Given a variable x, we write $s \downarrow x$ to denote the value of x in s.

Given an FCSP Q and a preference α, we will denote as $cut_\alpha(Q)$ the CSP obtained from Q allowing only tuples with preference greater than or equal to α. It is known that the set of solutions of Q with preference greater than or equal to α coincides with the set of solutions of $cut_\alpha(Q)$.

Finding an optimal solution is an NP-hard problem, unless certain restrictions are imposed, such as a tree-shaped constraint graph. Constraint propagation may help the search for an optimal solution. Given a variable ordering o, a FCSP is directional arc-consistent (DAC) if, for any two variables x and y linked by a fuzzy constraint, such that x precedes y in the ordering o, we have that, for each a in the domain of x, $f_x(a) = max_{b \in D(y)}(min(f_x(a), f_{xy}(a, b), f_y(b)))$, where f_x, f_y, and f_{xy} are the preference functions of c_x, c_y and c_{xy}. This definition can be generalized to any instance of the SCSP approach by replacing max with $+$ and min with $\times$. Therefore, for WCSPs it is sufficient to replace max with min and min with sum. DAC is enough to find the preference level of an optimal solution when the problem has a tree-shaped constraint graph and the variable ordering is compatible with the father-child relation of the tree [14]. In fact, such an optimum preference level is the best preference level in the domain of the root variable.

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 (or part of it) 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 collection of total orderings over the set of candidates, one for each voter. Given a profile, a *voting rule* maps it onto a single winning candidate (if necessary, ties are broken appropriately). In this paper, we will often use a terminology which is more familiar to multi-agent settings: we will sometimes call "agents" the voters, "solutions" the candidates, and "decision" or "best solution" the winning candidate. Some examples of widely used voting rules, that we will study in this paper, are:

- **Plurality**: each voter states a single preferred candidate, and the candidate who is preferred by the largest number of voters wins;
- **Borda**: given m candidates, each voter gives a ranking of all candidates, the i^{th} ranked candidate gets a score of $m - i$, and the candidate with the greatest sum of scores wins;
- **Approval**: given m candidates, each voter approves between 1 and $m - 1$ candidates, and the candidate with most votes of approval wins.

We know that every voting rule is manipulable [1]. However, if it is computationally difficult to influence the result by using a certain voting rule, we can say that the voting rule is *resistant* to such attempts. Thus the computational complexity of various attempts to influence the result of the voting process has been studied [2, 9, 5]. Besides manipulation, which refers to scenarios where there is a voter (or a group of voters) who can get a better result by lying on its preference ordering, another kind of attempt to influence the result is called *bribery*: there is an outside agent, called the briber, that wants to affect the result of the election by paying some voters to change their votes, while being subject to a limitation of its budget.

Sequential preference aggregation. Assume to have a set of agents, each one expressing its preferences over a common set of objects via an SCSP whose variable assignments correspond to the

objects. Since the objects are common to all agents, this means that all the SCSPs have the same set of variables and the same variable domains but they may have different soft constraints, as well as different preferences over the variable domains. In [7] this is the notion of *soft profile*, which is formally defined as a triple (V, D, P) where V is a set of variables (also called issues), D is a sequence of $|V|$ totally ordered finite domains, and P a sequence of m SCSPs over variables in V with domains in D. A *fuzzy profile* (resp., *weighted profile*) is a soft profile with fuzzy (resp., weighted) soft constraints. An example of a fuzzy profile where $V = \{x, y\}$, $D_x = D_y = \{a, b, c, d, e, f, g\}$), and P is a sequence of seven FCSPs, is shown in Fig. 2.

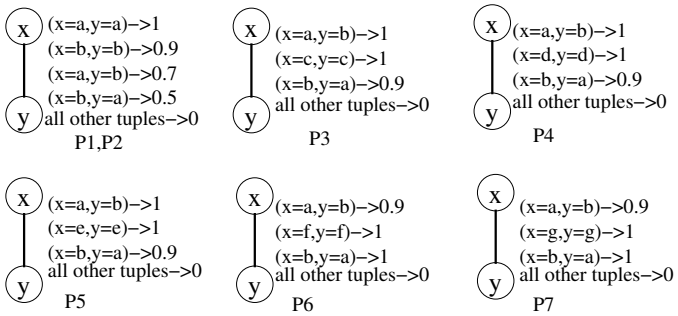


Figure 2. A fuzzy profile.

The idea proposed in [7, 6] to aggregate the preferences in a soft profile in order to compute the winning variable assignment is to sequentially vote on each variable via a voting rule, possibly using a different rule for each variable. Given a soft profile (V, D, P), assume $|V| = n$, and consider an ordering of such variables $O = \langle v_1, \ldots, v_n \rangle$, and a corresponding sequence of voting rules $R = \langle r_1, \ldots, r_n \rangle$ (that will be called "local rules"). The sequential procedure is a sequence of n steps, where at each step i,

- All agents are first asked for their preference ordering over the domain of variable v_i, yielding profile p_i over such a domain. To do this, the agents achieve DAC on their SCSP, considering the ordering O.
- Then, the voting rule r_i is applied to profile p_i, returning a winning assignment for variable v_i, say d_i. If there are ties, the first one following the given lexicographical order will be taken.
- Finally, the constraint $v_i = d_i$ is added to the preferences of each agent and DAC is applied to propagate its effect considering the reverse ordering of O.

After all n steps have been executed, the winning assignments are collected in the tuple $\langle v_1 = d_1, \ldots, v_n = d_n \rangle$, which is declared the winner of the election. This is denoted by $Seq_{O,R}(V, D, P)$. In the soft profile above, assume the variable ordering is $\langle x, y \rangle$ and $r_i =$ Approval for all $i = 1, \ldots, 6$. In step 1, agents apply DAC. This changes the preferences of the agents over x. For example, in P_1 and P_2, $x = a$ maintains preference 1, $x = b$ gets preferences 0.9, and all other domain values get preference 0, while in P_3, $x = a$ and $x = c$ maintain preference 1, $x = b$ gets preference 0.9, while all other values get preference 0. Then, Approval is applied on the profile over x where the sets of approved values are all the optimals: $\{a\}$ for the first two voters and respectively $\{c, a\}$, $\{d, a\}$, $\{e, a\}$, $\{f, b\}$, and $\{g, b\}$ for the others. Thus, $x = a$ is chosen and the constraint $x = a$ is added to all SCSPs, and its effect is propagated by achieving DAC on the domain of y. In step 2, achieving DAC does not modify any preference (since y is the last variable) and the set of approved values for y is $\{a, b\}$ for P_1 and P_2 and $\{b\}$ for the other agents. Thus the elected solution with the sequential procedure is $s = (x = a, y = b)$, which has preference 0.7 for P_1 and P_2, 1 for P_3, P_4, and P_5, and 0.9 for P_6 and P_7.

An alternative to this sequential procedure would be to generate the preference orderings for each voter from their FCSPs, and then to aggregate them in one step via a voting rule, for example Approval. In our example, $(x = a, y = b)$ gets 3 votes (that is, it is optimal for 3 agents), $(x = a, y = a)$ and $(x = b, y = a)$ each gets 2 votes, $(x = f, y = f)$, $(x = d, y = d)$, $(x = c, y = c)$, $(x = e, y = e)$, and $(x = g, y = g)$ each gets 1 vote, while all other solutions get no vote. Thus the winner is $(x = a, y = b)$.

3 The bribery problem

We consider scenarios where a collection of agents need to take a decision, by selecting it out of a common set of possible objects. Each agent has its own preferences over such objects, described via an SCSP, as described in Section 2, and charges the briber for changing his preferences according to a cost scheme. In this paper, by soft constraints we mean either fuzzy or weighted constraints. Also, we assume that all agents have tree-shaped SCSPs. Notice that the set of solutions of such constraint problems (that is, the set of decision among which to choose one) is in general exponentially large w.r.t. the size of the soft constraint problems. We also assume that the number of such solutions is exponentially large w.r.t. the number of agents. We now formally define the bribery problem of which we will study the computational complexity:

Definition 1 *Given a voting rule V and a cost scheme C, we denote by (V, C)-Bribery the problem of determining if it is possible to make a preferred candidate win, when voting rule V is used, by bribing agents according to cost scheme C and by spending less than a certain budget according to cost scheme C.*

3.1 Winner determination

It makes sense to consider only winner determination approaches which are polynomial to compute: if it is difficult to compute the winning decision, it is also difficult for a briber to compute how to bribe the agents (since he needs to know who the winner is without the bribery). We consider two main approaches: sequential and one-step. For the sequential approach, we employ the sequential voting procedure described in the previous section. We have an ordering O over the variables, and we are going to consider each variable in turn in such an ordering. At each step, each agent provides some information about the considered variable, say X, which depends on the voting rule we use:

- Sequential Plurality (SP): one best value for X;
- Sequential Approval (SA): all best values for X;
- Sequential Borda (SB): a total order (possibly with ties) over the values of X, along with the preference values for each domain element.

We then choose one value for the considered variable, as follow:

- SP and SA: the value voted by the highest number of agents;

- SB: the value with best score, where the score of a value is the sum of its preferences over all the agents; notice that "best" here means maximal in the case of fuzzy constraints, while it is the minimal in the case of weighted constraints.

Once a value is chosen for a variable, this value is broadcasted to all agents, who fix variable X to this value in their soft constraints and apply DAC in the reverse ordering w.r.t. O. We then continue with the next variable, and so on until all variables have been handled. The alternative to a sequential approach is a one-step approach, where each agent votes over decisions regarding all variables, not just one at a time. In this case, a possible voting rule to use is what we call One-step Plurality (OP), where each agent provides an optimal solution of his soft constraint problem, and then we select the solution which is provided by the highest number of agents.

For all the voting rules we consider (SP, SA, SB, and OP), it is computationally easy for an agent to vote. An approach like OP is however less satisfactory that the sequential approaches in terms of *ballot expressiveness*: since the number of solutions is exponentially large with respect to the number of agents, there is an exponential number of solutions which are not voted by any agent. However, if we want agents to be able to compute their vote in polynomial time, we need to set a bound to the number of solutions they can vote for, and this means that in all cases an exponentially large number of solutions will not be voted. So there is trade-off between ease of computing votes and ballot expressiveness.

We don't consider one step Approval since voting could require exponential time due to the fact that each agent may have an exponential size set of optimals.

3.2 Bribery actions and cost schemes

If we use Plurality to determine the winner, either in its sequential or one-step version, the most natural request a briber can have for an agent is to ask the agent to make a certain solution (or a certain value in the sequential case) optimal in his soft constraint problem. In order to do this, the agent can modify the preference values inside its variable domains and/or constraints. To define the cost of a briber's request, which is to make a certain solution A optimal, we consider the following approaches:

- C_{equal}: The cost is fixed (without loss of generality, we will assume it is 1), no matter how many changes are needed to make A optimal;
- C_{do}: The cost is the distance from the preference of A, denoted with $pref(A)$, to the optimal preference of the soft constraint problem of the agent, denoted with *opt*. If we are dealing with fuzzy numbers and we may prefer to have integer costs, the cost will be defined as $C_{do} = (opt - pref(A)) * l$, where l is the number of different preference values allowed. With weighted constraints, if costs are natural numbers, we may define $C_{do} = pref(A) - opt$, since *opt* is the smallest cost.
- C_{don}: The cost is determined by considering both the distance between the preference of A and the optimal preference, and the number of parts of A, say t, that correspond to the projections of A over the constraints, that must be modified in order to make A optimal. Thus, if we have n variables, with fuzzy constraints we may define $C_{don} = ((opt - pref(A)) * l * M) + t$, where M is a large integer and $1 \leq t \leq 2n-1$. If instead we consider weighted constraints, we define $C_{don} = ((pref(A) - opt) * M) + t$. In both cases, the role of M is to assure a higher bribery cost for a less preferred candidate: we want that the highest cost at a given preference level for A, that is, $d * M + 2n - 1$, where $d = (opt - pref(A)) * l$ and n is the number of variables, to be smaller than the lowest cost at the next preference level, that is, $(d+1)M + 1$. This yields $M > 2n - 2$.
- C_{dow}: The cost is computed by considering the same as in C_{don}, but each preference to be modified is associated with a cost proportional to the change required on that preference. If we denote by t_i any tuple of A with preference $\leq$ *opt*, then the cost will be $((opt - pref(A)) * l * M) + \sum_{t_i}(opt - pref(t_i)) * l$ for fuzzy constraints, where the role of M is similar to the one in C_{don}. For weighted constraints, we analogously define $C_{dow} = ((pref(A) - opt) * M) + \sum_{t_i}(pref(t_i) - opt)$. However, it is easy to see that $\sum_{t_i}(pref(t_i) - opt) = pref(A) - opt$, thus we have $C_{dow} = ((pref(A) - opt) * (M + 1))$.
- C_{donw}: The cost is the combination of C_{don} and C_{dow}. For fuzzy constraints: $C_{donw} = ((opt - pref(A)) * l * M) + t * M' + \sum_{t_i}(opt - pref(t_i)) * l$, where M' has a similar role than M w.r.t. the second a third component of the sum. For weighted constraints: $C_{donw} = ((pref(A) - opt) * M) + t$ (by simplifying as in C_{dow}).

In the names of the cost schemes C stands for cost, $_{do}$ stands for *distance* from the preference of A to the *optimal preference*, $_{n}$ stands for the *number of tuples* that must be changed to make A optimal, and $_{w}$ stands for *weighted cost*. In this paper, we consider only cost schemes that are obtained by some of these combinations. Other cost combinations can lead to interesting cost schemes but special care must be devoted to making sure a linearization of the solution ordering is induced.

4 Winner and cost determination are both computationally easy

As noted above, voting is easy. It is easy to check that also computing the winner is easy with any of the voting rules we consider and that it is easy also to compute the cost to respond to a briber's request, for all the cost schemes we have defined. Thus we have the following results:

Theorem 1 *Winner determination takes polynomial time for SP, SA, SB, and OP.*

Proof: For each variable, SP (resp., SA) requires a (resp., all) The fact that we are considering tree-shaped soft constraint problems ensures that voting, in all these cases, can be done in polynomial time by applying DAC. Winner determination is then polynomial as well, since it just requires a number of polynomial steps which equals the number of variables. For OP, computing an optimal solution is polynomial on tree-shaped soft constraint problems, so voting is polynomial. determining the winner requires just counting the number of votes for each of the voted candidates (which are in polynomial number), so it is polynomial as well. □

Theorem 2 *Given a tree-shaped fuzzy or weighted CSP and an outcome A, determining the cost to make A an optimal outcome takes polynomial time for C_{equal}, C_{do}, C_{don}, C_{dow}, and C_{donw} .*

Proof: We can check if A is already optimal in polynomial time by first computing the optimal preference *opt* and then checking if it coincides with the preference of A, denoted $pref(A)$. If so, the cost is 0. Otherwise, with C_{equal} the cost is always 1. To compute the

cost according to C_{do}, C_{don}, C_{dow}, and C_{donw}, we need to compute *opt*, the numbers of tuples of A with preference worse than *opt*, and the distance of their preferences from *opt*. All of these components can be computed in polynomial time with tree-shaped problems. □

5 Resistance to bribery

We show our complexity results about the resistance to bribery for sequential and one-step approach.

5.1 Voting sequentially

We now study the resistance to bribery of SP, SA, and SB.

Theorem 3 *(V, C)-Bribery is NP-complete (and also W[2]-complete with parameter being the budget) for $V \in \{SP, SA, SB\}$ and $C \in \{C_{equal}, C_{do}\}$.*

Proof: Membership in NP is easy to prove. To show completeness, we provide a polynomial reduction from the OPTIMAL LOBBYING (OL) problem [4]. In this problem, we are given an $m \times n$ $0/1$ matrix E and a $0/1$ vector $\vec{x}$ of length n where each column of E represents an issue and each row of E represents a voter. We say E is a binary approval matrix with 1 corresponding to a "yes" vote and $\vec{x}$ is the target group decision. We then ask if there a choice of k rows of the matrix E such that these rows can be edited so that the majority of votes in each column matches the target vector $\vec{x}$. This problem is shown to be $W[2]$-complete with parameter k. By giving a polynomial reduction from OL to our bribery problem, we show that our problem is NP-complete (actually $W[2]$-complete with parameter being the budget B). Given an instance $(E, \vec{x}, k)$ of OL, we construct an instance of (V-C_{do})-Bribery, where $V \in \{SP, SA, SB\}$, containing constraints with only independent binary variables. The number of variables, n, is equal to the number of columns in E. For each row of E, we create a voter with the preferences over the n variables as described in the row of E. In particular, for each variable the value indicated in the row will be associated with preference 1 while the other value will be associated with preference 0. Thus, each voter has a unique most preferred solution with preference 1 and all other complete assignments have preference 0. We set the preferred outcome $A = \vec{x}$. This means that according to C_{do}, all voters not voting for A have the same cost to be bribed, which is $(opt - pref(A)) * 2 = (1 - 0) * 2 = 2$. Finally, we set the budget $B = 2k$. With C_{equal}, the cost is always 1 if A is not already voted for. We note that since we have only two values for each variable, SP, SA and SB coincide with sequential majority, thus A wins the election if and only if there is a selection of k rows of E such that $\vec{x}$ becomes the winning agenda of the OL instance. Since both fuzzy and weighted CSPs generalize CSPs, the result holds also for such classes of soft constraints. □

Theorem 4 *(V,C)-Bribery is NP-complete (and also W[2]-complete) for $V \in \{SP, SA, SB\}$ and $C \in \{C_{don}, C_{dow}, C_{donw}\}$, if $M > n * m$, where n is the number of variables and m the number of voters.*

Proof: We use a reduction similar to the one described for Thm. 3 from the optimal lobbying problem. In particular the structure of the soft profile is the same. The only things that vary are the costs for each voter and the budget. With fuzzy constraints, assume that we have l different levels of preferences and let us denote with d_i the positive integer $(opt_i - pref(A)) * l$, were i varies over the voters. For C_{don}, the cost for voter i is $d_i * M + t_i$ where t_i is the number of tuples where the candidate voted by voter i differs from A. For C_{dow}, the cost is $d_i * M + \sum_{t \in Diff_i(A)}(opt_i - pref(t))$, where $Diff_i(A)$ is the set of tuples in the soft constraint problem of agent i which not belong to A. Let us define budget B to be $B = kl(M + n)$ for fuzzy constraints and $B = k(M + n)$ for weighted constraints. Since we have only binary variables, SP, SA and SB coincide with sequential majority. There is a bribery strategy that does not exceed B if and only if there is a way to change at most k rows to solve the OL problem. We note that requiring $M > n * m$ is of key importance for the connection between the budget B and the modifications of k rows. For C_{donw}, the cost is $d_i * M + t_i * M' + \sum_{t_i \in Diff_i(A)}(opt_i - pref(t_i))$. Here a similar constraint for M' would work for the reduction. For weighted constraints, a similar reasoning works as well. □

5.2 Voting with OP

We now show that OP is not resistant to bribery. To do this, we will need to compute n cheapest alternative candidates for each agent to vote for. We will thus start by studying the computational complexity of this task.

Theorem 5 *Given a tree-shaped fuzzy or weighted CSP, computing a set of k cheapest outcomes according to C_{do} and C_{equal} is in $\mathcal{P}$ when k is given in unary.*

Proof: The cost of an outcome according to C_{do} is an integer proportional to the distance between the preference of the outcome and the preference of an optimal outcome. In order to compute k cheapest solutions, we assume to have a linear order over the variables and the values in their domains. Such linear orders can be provided by the agent or can be chosen by the system. They do not need to be the same for all agents. For tree-shaped fuzzy CSPs, it has been shown in [3] that, given such linear orders and an outcome s, it is possible to compute, in polynomial time, the outcome following s in the induced lexicographic linearization of the preference ordering over the outcomes. The procedure that performs this is called Next. Thus, in order to compute k cheapest according to C_{do}, we compute the first optimal outcome according to the linearization and then we generate the set of k cheapest candidates by applying Next $k - 1$ times (each time on the outcome of the previous step). Similarly, computing the k best solutions of a weighted CSP can be done in polynomial time by using the procedure suggested in [13]. If we consider C_{equal}, an agent will not charge the briber for changing his vote to another optimal candidate and will charge a fixed cost to change his vote in favor of any other (non-optimal) candidate. Thus any of the above procedures can be used (although, if k exceeds the cardinality of the set of optimal solutions, the remaining ones could, in principle, be generated randomly in a much faster way). □

Theorem 6 *Given a tree-shaped weighted CSP, computing a set of k cheapest outcomes according to C_{dow} is in $\mathcal{P}$ when k is given in unary.*

Proof: This result follows immediately from the fact that, for weighted CSPs, C_{dow} is proportional to C_{do}. □

For the other cost schemes, we define a general algorithm, called $KCheapest$, that will work for C_{don}, as well as C_{dow} and C_{donw}, via small modifications. In what follows we assume a voter represents his preferences with a tree-shaped fuzzy CSP. The input to

$KCheapest$ is a tree-shaped fuzzy CSP P, an integer k, and a cost scheme C. The output is a set of k cheapest solutions of P according to C. $KCheapest$ performs the following steps:

1. **Find k optimal solutions of P, or all optimal solutions if they are less than k.** If the number of solutions found is k, we stop, otherwise let k' be the number of remaining solutions to be found.
2. **Look for the remaining top solutions within non-optimal solutions**. More in detail, until k' best solutions have been found or all solutions of P have been exhausted, consider each preference pl associated to some tuple in P in decreasing order and, for each tuple t of P with preference pl, perform the following:
   (a) Compute the new fuzzy CSP, P_t, obtained by fixing the tuple in the constraint (that is, by forbidding all other tuples in that constraint).
   (b) Compute a new soft CSP, say P_t^w, associated to P_t, defined as follows:
      i. the constraint topology of P_t^w and P_t coincide;
      ii. each tuple with a preference greater or equal than opt in P_t has weight 0 in P_t^w;
      iii. each tuple with a preference pt such that $pl \leq pt < opt$ in P_t has weight c in P_t^w defined as follows: $c = 1$ if $C = C_{do}$, $c = pt - opt$ if $C = C_{dow}$ and $c = (1, pt - opt)$ if $C = C_{donw}$;
      iv. each tuple with preference less than pl in P_t has weight $+\infty$ in P_t^w.

      Thus, P_t^w is a weighted CSP if $C = C_{don}$ or $C = C_{dow}$, while it is a SCSP defined on the Cartesian product of two weighted semirings if $C = C_{donw}$.
   (c) Compute the k' best solutions or all the solutions if they are less than k' of P_t^w.

   Take the k' top solutions (or all solutions if less than k') among the sets of best solutions computed for P_t^w, $\forall t$ such that $pref(p) = pl$.

Theorem 7 *Given a tree-shaped fuzzy CSP P, computing a set of k cheapest outcomes according to C_{don}, C_{dow}, and C_{donw} is in $\mathcal{P}$.*

Proof: (Sketch) For all cost schemes, optimal solutions are always cheaper than other solutions. Thus step 1 is correct. In step 2, the solutions of any P_t^w correspond to solutions of P with preference pl, and considering all such problems allows to cover all solutions of P with such a preference. The way weights are defined in P_t^w allows to order solutions with the same preference, respectively, in increasing order either w.r.t. the number of tuples that need to be changed in order to make the solution optimal ($c = 1$), or w.r.t. the weighted sum of the changes ($c = opt - pt$) needed to make the solutions optimal, or in lexicographic order with respect to these two criteria where the number of tuples to be changed comes first ($c = (1, opt - pt)$). These three ways of breaking ties among solutions with the same preference correspond, respectively, to C_{don}, C_{dow}, and C_{donw}. All remaining ties are assumed to be broken using a lexicographic ordering induced by linear orders over the variables and the values in the domains. In terms of computational complexity, step 1 is achieved by computing the optimal preference level opt, and obtaining the tree-shaped CSP corresponding to the opt-cut of P, denoted with P^{opt}. Then $KCheapest$ finds a set of k solutions of P^{opt}. This is done by exploiting a lexicographic order over the solutions, by finding the first optimal solution according to such an order, and by iteratively finding the following next best solutions, until either $k - 1$ steps have been performed, or the set of solutions has been exhausted. This can be done in polynomial time as shown in [3]. Also step 2 is polynomial, since computing the k best solutions on a weighted tree-shaped CSP can be done in polynomial time by using the procedure in [13]. □

Theorem 8 *(OP, C)-Bribery is in $\mathcal{P}$ for $C \in \{C_{equal}, C_{do}, C_{don}, C_{dow}, C_{donw}\}$ when agents vote with tree-shaped fuzzy CSPs and for $C \in \{C_{equal}, C_{do}, C_{dow}\}$ when agents vote with tree-shaped weighted CSPs.*

Proof: Main idea: Faliszewski [8] shows that bribery when voting with plurality in single variable elections with non-uniform cost schemes is in $\mathcal{P}$ through the use of flow networks. The algorithm requires the enumeration of all possible elements of the candidate set as part of the construction of the flow network. In our model, the number of candidates can be exponential in the size of the input, so we cannot use that construction directly. However, a similar technique works by considering only a polynomial number of candidates. However, such candidates need to be the (at most) $2n + 1$ cheapest candidates for each voter. This is why we need to be able to compute such candidates in polynomial time, via the methods described above. □

5.3 Summary of bribery results

Our complexity results about the resistance to bribery when aggregating the preferences of a collection of agents, if they are modelled via soft constraints, can be seen in Table 1. We can see that OP is not resistant to bribery, since it is computationally easy for the briber to compute who to bribe and what to ask for, and to check whether he can do it within its budget. On the other hand, the sequential approaches (SP, SA, and SB) are all resistant to bribery, if agents compute costs according to C_{equal}, C_{do}, C_{don}, C_{dow}, or C_{donw}. Thus, it is clear that sequential approaches should be preferred if resistance to bribery is an important feature. Notice that, when a problem is polynomial for soft constraints, it is also so for CSPs. Thus, OP is easy to bribe also when agents use CSPs.

	SP	SA	SB	OP
C_{equal}	NP-c	NP-c	NP-c	P
C_{do}	NP-c	NP-c	NP-c	P
C_{don}	NP-c*	NP-c*	NP-c*	P/?
C_{dow}	NP-c*	NP-c*	NP-c*	P
C_{donw}	NP-c*	NP-c*	NP-c*	P/?

Table 1. Our results. NP-c* stands for NP-complete with the restriction on M (and M' if present). When the complexity results for fuzzy constraints and weighted constraints are different, we write X/Y, where X is the complexity for fuzzy and Y is the complexity for weighted constraints.

6 Bribery cost schemes in preference optimization and compilation

There are many constraint reasoning scenarios in which it is useful to consider, given a solution, how far it is from being optimal. One natural way to evaluate this is to consider the difference between its preference and the preference of an optimal solution, or one may want to take into account also the effort that would be required in terms of changes needed in the soft constraints to make such a solution optimal. This is the same kind of reasoning that led us to the bribery cost

schemes defined in Section 3.2. Given a tree-shaped SCSP P, let us define for each solution s the following triple of values (p_s, t_s, w_s) where $p_s = opt(P) - pref(s)$, t_s is the minimum number of tuples of s that must be changed to make s optimal, and w_s is the sum of the amount of changes that must be performed on such tuples to make s optimal. Given such a triple, we can define the following notions of distance of a solution s from optimality: do is the distance is p_s, that is, the distance is computed only looking at the first component of the triple; don is the distance is determined by considering first p_s and then t_s; dow is the distance is determined by considering first p_s and then w_s; $donw$ is the distance is determined by first considering p_s, then t_s, and then w_s.

Often finding just one optimal solution may not be enough and it may be desirable to produce a set of top solutions. This occurs, for example, in web search or configuration problems, where we usually want more than one answer to our query. With soft constraints, usually the number of different preference values is much smaller than the number of solutions, thus many solutions end up having the same preference. When computing the k best solutions, it is thus necessary to employ a tie-breaking rule among solutions with the same preference value. Some of the notions of distance from optimality defined above provide a meaningful way to tie-break: in addition to the distance of the solution preference from the optimal preference, they consider also the "structural" distance of the solution from being optimal. For example, among the solutions with the same preference, don will put first the ones that require the minimum number of tuples to be changed. Besides this, dow weights each tuple to be modified with the amount by which it must be modified. A refinement of don is $donw$, which considers the amount of changes needed only in case of a tie on both the preference distance and the number of tuples to be modified.

From results in [3] and [13], we know that computing the k best solutions according to do is polynomial, for both fuzzy and weighted constraints. The algorithms defined above to compute the cheapest top solutions allow us to give the following result on the complexity of computing the k best solutions according to the new refinements of the solution preference ordering.

Theorem 9 *Computing k best solutions is in $\mathcal{P}$ according to distances don, dow and donw for tree-shaped fuzzy CSPs and according to dow for tree-shaped weighted CSPs.*

Proof: It follows from the complexity of the $KCheapest$ algorithm. □

In the process of modeling a real-life problem via soft constraints, it can be reasonable to allow users to require that a certain solution be made optimal (e.g., in preference compilation) in the current soft constraint problem. To measure the effort needed to achieve this, we can use the distance notions defined above. Theorem 2 allows us to state the following result:

Theorem 10 *Computing the effort needed to make a solution optimal is in P when using fuzzy or weighted tree-shaped constraint problems and any of the distance notions do, don, dow, and donw.*

Proof: It follows from Theorem 2. □

7 Conclusions and future work

We have studied resistance to bribery when aggregating preferences of several agents expressed via soft constraints. This has led to several results that are interesting and useful in themselves. However, they also have a wider applicability within typical CP tasks, such as computing the top k solutions and encoding solution preferences. With regard to this, we plan to study the link with robust CSP [12, 11]. We believe that this paper is just a first step in a very promising multi-disciplinary research line, which creates useful links between CP issues and voting theory.

REFERENCES

[1] K. J. Arrow and A. K. Sen amd K. Suzumura, *Handbook of Social Choice and Welfare.*, North-Holland, 2002.

[2] J.J. Bartholdi, C.A. Tovey, and M.A. Trick, 'The computational difficulty of manipulating an election', *Social Choice and Welfare*, **6**(3), 227–241, (1989).

[3] R. I. Brafman, F. Rossi, D. Salvagnin, K. B. Venable, and T. Walsh, 'Finding the next solution in constraint- and preference-based knowledge representation formalisms', in *Proc. KR 2010*, (2010).

[4] R. Christian, M. Fellows, F. Rosamond, and A. Slinko, 'On complexity of lobbying in multiple referenda', *Review of Economic Design*, **11**(3), 217–224, (2007).

[5] V. Conitzer, T. Sandholm, and J. Lang, 'When are elections with few candidates hard to manipulate?', *JACM*, **54**(3), 1–33, (2007).

[6] G. Dalla Pozza, M. S. Pini, F. Rossi, and K. B. Venable, 'Multi-agent soft constraint aggregation via sequential voting', in *IJCAI*, pp. 172–177, (2011).

[7] G. Dalla Pozza, F. Rossi, and K. B. Venable, 'Multi-agent soft constraint aggregation:a sequential approach', in *ICAART*, pp. 277–282, (2011).

[8] P. Faliszewski, 'Nonuniform bribery', in *Proc. AAMAS 2008*, pp. 1569–1572, (2008).

[9] P. Faliszewski, E. Hemaspaandra, and L.A. Hemaspaandra, 'How hard is bribery in elections?', *JAIR*, **35**, 485–532, (2009).

[10] T. Haynes and S. Sen, 'Satisfying user preferences while negotiating meetings', in *Proc. ICMAS'96*, (1996).

[11] E. Hebrard, B. Hnich, and T. Walsh, 'Robust solutions for constraint satisfaction and optimization', in *Proc. ECAI'04*, pp. 186–190, (2004).

[12] E. Hebrard, B. Hnich, and T. Walsh, 'Super solutions in constraint programming', in *Proc. CPAIOR'04*, pp. 157–172, (2004).

[13] E. Rollon, N. Flerova, and R. Dechter, 'Inference schemes for m best solutions for soft CSPs', in *Proc. SOFT'11*, (2011).

[14] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of Constraint Programming*, Elsevier, 2006.

Manipulating Two Stage Voting Rules

Nina Narodytska[1] and **Toby Walsh**[2]

Abstract. We study the computational complexity of computing a manipulation of a two stage voting rule. An example of a two stage voting rule is Black's procedure. The first stage of Black's procedure selects the Condorcet winner if they exist, otherwise the second stage selects the Borda winner. In general, we argue that there is no connection between the computational complexity of manipulating the two stages of such a voting rule and that of the whole. However, we also demonstrate that we can increase the complexity of even a very simple base rule by adding a stage to the front of the base rule. In particular, whilst Plurality is polynomial to manipulate, we show that the two stage rule that selects the Condorcet winner if they exist and otherwise computes the Plurality winner is NP-hard to manipulate with 3 or more candidates, weighted votes and a coalition of manipulators. In fact, with any scoring rule, computing a coalition manipulation of the two stage rule that selects the Condorcet winner if they exist and otherwise applies the scoring rule is NP-hard with 3 or more candidates and weighted votes. It follows that computing a coalition manipulation of Black's procedure is NP-hard with weighted votes. With unweighted votes, we prove that the complexity of manipulating Black's procedure is inherited from the Borda rule that it includes. More specifically, a single manipulator can compute a manipulation of Black's procedure in polynomial time, but computing a manipulation is NP-hard for two manipulators.

1 Introduction

There exist several voting procedures that work in stages. For example, Black's procedure is a two stage voting rule whose first stage elects the Condorcet winner, if one exists, and otherwise moves to a second stage which elects the Borda winner [12]. As a second example, the French presidential elections use a two stage runoff voting system. If there is a majority winner in the first stage, then this candidate is the overall winner, otherwise we go to the second stage where there is a runoff vote between the two candidates with the most votes in the first round. Such two stage voting rules can inherit a number of attractive axiomatic properties from their parts. For example, Black's procedure inherits Condorcet consistency from its first part, and properties like monotonicity, participation and the Condorcet loser property from its second part. Inheriting such properties from its parts might be considered an attractive feature of two stage voting rules. On the other hand, a less desirable property of one of the base rules can infect the overall two stage rule. For instance, it has been shown that, with single peaked votes, many types of control and manipulation problems are polynomial for Black's procedure [4]. This polynomiality is essentially inherited from the first stage of the rule which selects the Condorcet winner (which must exist with single peaked votes). Such vulnerability to manipulation and control might be considered an undesirable property for a two stage voting rule. This raises several interesting questions from the perspective of computational social choice. For example, with unrestricted votes as opposed to single peaked votes, are two stage voting rules more or less computationally difficult to manipulate than single stage voting rules? How does the computational complexity of manipulating a two stage voting rule depend on the computational complexity of manipulating the two rules that it composes? In this paper, we address such questions.

[1] NICTA and UNSW, Sydney, Australia. email: nina.narodytska@nicta.com.au
[2] NICTA and UNSW, Sydney, Australia. email: toby.walsh@nicta.com.au

2 Background

A *profile* is a sequence of n total orders over m candidates. A *voting rule* is a function mapping a profile onto a set of *winners* (strictly speaking this is a social choice correspondence). We consider some of the most common voting rules.

Scoring rules: Given a *scoring vector* $(w_1, \ldots, w_m)$ of weights, the ith candidate in a vote scores w_i, and the winner is the candidate with highest total score over all the votes. The **Plurality** rule has the weight vector $(1, 0, \ldots, 0)$, the **Veto** rule has the vector $(1, 1, \ldots, 1, 0)$, and the **Borda** rule has the vector $(m-1, m-2, \ldots, 0)$.

Cup: The winner is the result of a series of pairwise majority elections between candidates. Given the *agenda*, a binary tree in which the roots are labelled with candidates, we label the parent of two nodes by the winner of the pairwise majority election between the two children. The winner is the label of the root.

Black's procedure: This rule has two stages. We first determine if there is a *Condorcet winner*, a candidate that beats all others in pairwise majority comparions. If there is, this is the winner. Otherwise, we return the result of the *Borda* rule.

Single Transferable Vote (STV): This rule requires up to $m-1$ rounds. In each round, the candidate with the least number of voters ranking them first is eliminated until one of the remaining candidates has a majority.

Nanson's and Baldwin's rules: These are iterated versions of the Borda rule. In Nanson's rule, we compute the Borda scores and eliminate any candidate with less than half the mean score. We repeat until there is an unique winner. In Baldwin's rule, we compute the Borda scores and eliminate the candidate with the lowest score. We again repeat until there is an unique winner.

Coombs' rule: This is an iterated version of the Veto rule. We repeatedly eliminate the candidate with the most vetoes until we have one candidate with a majority.

We consider both unweighted and integer weighted votes. A weighted votes can simply be viewed as a block of identical unweighted votes.

3 Two stage voting rules

We consider a general class of two stage voting rules. Given voting rules X and Y, the rule XTHENY applies the voting Y to the profile constructed by eliminating all but the winning candidates from the voting rule X. Both X and Y can themselves be two stage voting rules giving us the possibility to construct multi-stage voting rules. For example, Black's procedure is $Condorcet$THEN$Borda$ where $Condorcet$ is the multi-winner rule that elects the Condorcet winner if it exists, and otherwise elects all candidates. As a second example, Plurality with Runoff is $TopTwo$THEN$Majority$ where $TopTwo$ is the multi-winner voting rule that elects the candidates with the two most plurality votes. There are many possible rules that we might choose to combine this way. $Condorcet$ is an attractive choice for the first rule as it guarantees that the resulting combination is Condorcet consistent. However, there are other interesting choices including:

CondorcetLoser: This is the rule that elects all candidates except, when it exists, the Condorcet loser.

CopelandSet: This is the rule that elects all candidates in the Copeland set. The Copeland score of a candidate is the number of candidates that it beats less the number of candidates that beats it. The Copeland set contains those candidates with the maximal Copeland score. When there is a Condorcet winner, this is the only candidate in the Copeland set.

SmithSet: This is the rule that elects all candidates in the Smith set. This is the smallest non-empty set of candidates such that every candidate in the set beats every candidate outside the set in pairwise elections. When there is a Condorcet winner, this is the only candidate in the Smith set. Voting rules like Nanson's and Kemeny are guaranteed to pick candidates from the Smith set. In the Related Work section, we argue that $SmithSet$ is used within the Alternative Smith rule [17].

SchwartzSet: This is the rule that elects all candidates in the Schwartz set. The Schwartz set is a subset of the Smith set and is the union of all the undominated sets. A set is undominated if every candidate inside the set is pairwise unbeaten by every candidate outside, and no non-empty proper subset satisfies this property. When there is a Condorcet winner, this is the only candidate in the Schwartz set.

We can also consider recursive definitions. We suppose any recursion terminates when either we have a single candidate left, or the set of candidates left does not reduce in size. For example, we can recursively define STV by $STV = PluralityLoser$THENSTV where $PluralityLoser$ is the rule that elects all candidates but the candidate with the fewest first place votes. As a second example, we can recursively define Baldwin's rule by $Baldwin = BordaLoser$THEN$Baldwin$ where $BordaLoser$ is the multi-winner rule that elects all candidates but the candidate with the lowest Borda score. Nanson's rule can be defined recursively in a similar way. As a third example, we can define Coombs' rule by $Coombs = Majority$THEN$(VetoLoser$THEN$Coombs)$ where $Majority$ elects the candidate with a majority of first place votes or, if there is no such candidate, elects all candidates, and $VetoLoser$ is the rule that elects all candidates but the candidate with the most last placed votes.

4 Axiomatic and algebraic properties

It is interesting to consider which axiomatic properties are inherited from the base rules being combined. For example, it is simple to see that we can inherit Condorcet consistency or the Condorcet loser properties.

Proposition 1. *For any voting rule X, the combinations $Condorcet$THENX, $CopelandSet$THENX, $SmithSet$THENX and $SchwartzSet$THENX are Condorcet consistent. Similarly, for any voting rule Y, the combination $CondorcerLoser$THENY satisfies the Condorcet loser property.*

With recursivley defined rules, we can give a similar result. We say that a multi-winner rule is Condorcet consistent if it includes the Condorcet winner in the set of winners, and satisfies the Condorcet loser property if the set of winners never includes the Condorcet loser.

Proposition 2. *Suppose Y is recursivley defined by $Y = X$THENY and X is Condorcet consistent. Then Y is also Condorcet consistent. Similarly, if X satisfies the Condorcet loser property then Y does also.*

Note that the Borda loser is never the Condorcet winner. Hence, the multi-winner rule $BordaLoser$ is Condorcet consistent. Thus, it follows from Proposition 2 that Baldwin's rule (which is recursively defined using $BordaLoser$) is also Condorcet consistent.

There are also axiomatic properties which can be lost by combining together voting rules. For example, the Borda loser rule which eliminates the lowest Borda scoring candidate is monotonic since increasing one's preference for a candidate can only prevent them from being the Borda loser. However, Baldwin's rule, which is the recursive version of the Borda loser rule, is not monotonic. It will therefore be interesting to identify conditions under which two stage voting rules are monotonic.

This combinator has a number of interesting algebraic properties. For example, the $Identity$ rule that returns all candidates is a left and right identity of the THEN combinator. Note that the THEN combinator is neither commutative nor associative. If a voting rule is recursively defined then it is idempotent (that is, XTHEN$X = X$). More complex algebraic idententies can be derived such as the following.

Proposition 3. *If X is idempotent then XTHEN$(X$THEN$Y) = X$THENY and $(Y$THEN$X)$THEN$X = Y$THENX.*

More specialized properties can also be derived such as the following.

Proposition 4. $SmithSet$THEN$Nanson = Nanson$.

Proposition 5. *If X is Condorcet consistent and only returns the Condorcet winner when they exist then $Condorcet$THEN$X = X$.*

5 Complexity of manipulation

One of the main contributions of this paper is to consider the impact of two stage voting rules on the computational complexity of computing a manipulation. As in previous studies (e.g. [2, 6]), we consider manipulation with unweighted votes and a small number of manipulators, and manipulation with weighted votes, a coalition of manipulators and a small number of candidates.

5.1 Weighted votes, general results

With weighted votes, we first argue that is no connection in general between the computational complexity of computing a manipulation of a two stage voting rule and the computational complexity of manipulating its parts.

Proposition 6. *There exist voting rules X and Y with the following properties for weighted votes:*

1. *computing coalition manipulations of X, Y and XTHENY are polynomial;*
2. *computing coalition manipulations of X and Y are polynomial but of XTHENY is NP-hard;*
3. *computing a coalition manipulation of X is polynomial and of Y is NP-hard, but of XTHENY is polynomial;*
4. *computing a coalition manipulation of X is polynomial, but of Y and XTHENY are NP-hard;*
5. *computing a coalition manipulation of X is NP-hard, but of Y and XTHENY are polynomial;*
6. *computing a coalition manipulation of X is NP-hard and of Y is polynomial, but of XTHENY is NP-hard;*
7. *computing coalition manipulations of X and Y are NP-hard but of XTHENY is polynomial;*
8. *computing coalition manipulations of X, Y and XTHENY are NP-hard.*

Proof: The NP-hardness results are derived from the NP-hardness of computing a coalition manipulation of STV with 3 or more candidates [7].

1. Consider $X = FirstRoundCup$ and $Y = Cup$. $FirstRoundCup$ is the multi-winner rule that runs one round of the Cup voting rule. Note that $FirstRoundCup$THENCup is the Cup rule itself, and both $FirstRoundCup$ and Cup are polynomial to manipulate by a coalition even with weighted votes [7].
2. Consider $X = TopTwo$ and $Y = Majority$ where $TopTwo$ elects the two candidates with the two highest plurality scores. On 3 candidates, $TopTwo$THEN$Majority$ is Plurality with runoff, which itself is equivalent STV which is NP-hard to manipulate by a coalition of weighted voters when we have 3 or more candidates [7].
3. Consider $X = Plurality$ and $Y = STV$. Note that XTHENY is again Plurality which is polynomial to manipulate by a coalition even with weighted votes [7].
4. Consider $X = Identity$ and $Y = STV$ where $Identity$ is the identity rule that elects all the candidates in the election. Note that XTHENY is also STV.
5. Consider $X = STV'$ which is the multi-winner voting rule that elects both the STV winner and the candidate with the smallest label, and Y elects the candidate with the smallest label. Note that XTHENY always elects the candidate with the smallest label. Such a rule is polynomial to manipulate by a coalition even with weighted votes.
6. Consider $X = STV$ and $Y = Identity$. Note that XTHENY is again STV.
7. Consider $X = STV''$ and $Y = STV'''$ where STV'' is the multi-winner rule that elects the STV winner as well as those candidates with the smallest and largest labels, and STV''' elects the plurality winner between the smallest and largest candidates if there are 3 or fewer candidates and otherwise elects the STV winner. Note that XTHENY elects the plurality winner between the smallest and largest candidates, and computing a coalition manipulation of such a rule is polynomial even with weighted votes.
8. Consider $X = Y = STV$. Note that XTHENY is also STV.

♡

5.2 Weighted votes, specific rules

With weighted votes, we already know that several multi-stage voting rules are NP-hard to manipulate including STV, Plurality with runoff, Baldwin's rule (all with 3 candidates), and Nanson's rule (with 4 candidates) [7, 15]. We first show that computing a manipulation of $Condorcet$THENX with weighted votes is NP-hard for any scoring rule X. This contrasts to scoring rules in general where computing a coalition manipulation is NP-hard for any rule that is not isomorphic to Plurality, but is polynomial for Plurality. The demonstrates that adding the test for a Condorcet winner to give $Condorcet$THENX increases the computational complexity of manipulation over that for the scoring rule X alone.

Proposition 7. *Deciding whether there exists a coalitional manipulation for $Condorcet$THEN$Plurality$ with weighted votes is NP-complete with 3 or more candidates.*

Proof: We reduce from the number partitioning problem with n numbers k_i, $i = 1, \ldots, n$, $\sum_{i=1}^{n} k_i = 2K$. We have n manipulators with the weight k_i each.

Consider a non-manipulator profile. Suppose the voters with the total weight $2K$ cast (a, b, p) and the voters with the total weight $2K$ cast (b, a, p). The candidate p is a Condorcet loser as it loses to both a and b. Moreover, as a and b are tied, there is no Condorcet winner.

Note that if all manipulators put p in the first position then p wins under plurality. However, the manipulators have to make sure that they also do not make a or b the Condorcet winner. Note that if a (b) gets a higher score than b (a) then a (b) is the Condorcet winner. Therefore, the only way to prevent one of them from becoming the Condorcet winner is to partition scores between a and b. Thus, manipulators with a total score of K have to vote (p, a, b) and the remaining manipulators have to vote (p, b, a). Therefore, there exists a manipulation iff there is a partition with the required sum K. ♡

Proposition 8. *With weighted votes and any scoring rule X that is not isomorphic to Plurality, computing a coalition manipulation of $Condorcet$THENX is NP-hard for 3 or more candidates.*

Proof: Without loss of generality, we consider a scoring rule which gives a score of α_1 for a candidate in 1st place in a vote, α_2 for 2nd place, and 0 for 3rd place. We adapt the reduction used in the proof of Theorem 6 in [8] for the NP-hardness of manipulating any scoring rule that is not isomorphic to Plurality voting. The reduction is from the number partitioning problem and constructs an election with a weight of $6\alpha_1 K - 2$ votes over the candidates a, b and p (who the manipulators wish to make win). Within these votes, the manipulators have a weight of $2(\alpha_1 + \alpha_2)K$ votes, and the rest are fixed. The number partition problem is to divide a set of integers summing to $2K$ into two equal sums. There is a manipulator of weight k_i for every integer k_i in the set being partitioned. We now add $6\alpha_1 K - 1$ triples of votes: (a, b, p), (b, p, a), (p, a, b). This has no impact on the differences in the scores between the candidates. However, it creates a Condorcet cycle so that there cannot be a Condorcet winner whatever the manipulators do with their votes. Hence, we must pass to the second round where the winner is decided by the scoring rule X. As in the proof of Theorem 6 in [8], there is a manipulation that makes p the winner of the scoring rule X iff there is a partition into two equal sums. Thus, computing a coalition manipulation of $Condorcet$THENX is NP-hard. ♡

It follows immediately that coalition manipulation of Black's procedure, which is $Condorcet$THEN$Borda$ is NP-hard with 3 or more candidates.

Corollary 1. *With weighted votes, coalition manipulation of Black's procedure is NP-hard with 3 or more candidates.*

5.3 Unweighted votes, general results

As with weighted votes, there is no connection in general between the computational complexity of computing a manipulation of a two stage voting rule with unweighted votes and the computational complexity of computing a manipulation of its parts.

Proposition 9. *There exist voting rules X and Y with the following properties:*

1. *computing manipulations of X, Y and XTHENY are polynomial;*
2. *computing manipulations of X and Y are polynomial but of XTHENY is NP-hard;*
3. *computing a manipulation of X is polynomial and of Y is NP-hard, but of XTHENY is polynomial;*
4. *computing a manipulation of X is polynomial, but of Y and XTHENY are NP-hard;*
5. *computing a manipulation of X is NP-hard, but of Y and XTHENY are polynomial;*
6. *computing a manipulation of X is NP-hard and of Y is polynomial, but of XTHENY is NP-hard;*
7. *computing manipulations of X and Y are NP-hard but of XTHENY is polynomial;*
8. *computing manipulations of X, Y and XTHENY are NP-hard.*

Proof: The NP-hardnes results are derived from the NP-hardness of manipulating STV with unweighted votes and a single manipulator [2].

1 Identical examples to the weighted case.

2 Consider the multiwinner voting rule X that eliminates the incumbent candiate, and the the rule Y that elects the plurality winner between the candidates that are preferred by at least one voter to the incumbent or, if there are no such candidates, the STV winner. Now X is polynomial to manipulate as it ignores the votes. Similarly, Y is polynomial to manipulate since the manipulators should always put the candidate that they wish to win in first place, and the incumbent anywhere else in their vote. If all other voters prefer the incumbent to any other candidate, then this vote will ensure that the manipulators' preferred candidate wins. On the other hand, if the other voters prefer one ore more candidates to the incumbent, then this is the best vote for ensuring the manipulators' preferred candidate is the plurality winner. Now XTHENY is NP-hard to manipulate. We adapt the reduction used in [2] to prove that STV is NP-hard to manipulate by a single manipulator. We simply introduce an additional candidate, the incumbent into the voting profile used in this proof.

3-8 Identical examples to the weighted case.

♡

5.4 Unweighted votes, specific rules

With unweighted votes, we already know that a number of specific multi-stage voting rules are NP-hard to manipulate including STV [2], Nanson's, Baldwin's [15] and Coombs rules [10] (all with a single manipulator). We can add to this list Black's procedure. Like Borda voting on which it is based, a single manipulator can compute a manipulation of Black's procedure in polynomial time, but coordinating two manipulators makes the problem NP-hard.

Proposition 10. *Manipulation of Black's procedure with unweighted votes and two manipulators is NP-hard.*

Proof: We adapt the reduction used in the proof of Theorem 3.1 in [3] for the NP-hardness of manipulating Borda voting. This reduction is from a special case of numerical matching with target sums. It constructs an election with 5 votes, 3 fixed votes and 2 votes of the manipulators over the candidates 1 to m. We now add 6 sets of cyclic votes: $(1, 2, \ldots, m-1, m)$, $(2, 3 \ldots, m, 1)$, $\ldots$, $(m-1, m, \ldots, m-3, m-2)$, $(m, 1, \ldots, m-2, m-1)$. This has no impact on the differences in the scores between the candidates. However, it creates a Condorcet cycle so that there cannot be a Condorcet winner whatever the manipulators do with their two votes. Hence, we must pass to the second round where the winner is decided by the Borda rule. As in the proof of Theorem 3.1 in [16], there is a manipulation that makes a chosen candidate the Borda winner iff there is a solution to the numerical matching problem with target sums. Thus, computing a manipulation of $Condorcet$THEN$Borda$, which is Black's procedure, is NP-hard. ♡

Proposition 11. *Deciding whether one manipulator can make a candidate win for Black's procedure with unweighted votes is polynomial.*

Proof: We consider several cases.

Suppose no Condorcet winner exists in E_{NM} but there are $a \neq p$ and $b \neq p$ such that $beat_{E_{NM}}(a, b) = beat_{E_{NM}}(b, a)$, where $beat_E(v_1, v_2)$ is the number of times v_1 beats v_2 in E. In this case, p loses regardless of the manipulator v as v gives an advantage of one vote to a or b in any case. Hence, one of a or b must be the Condorcet winner.

Suppose there is the Condorcet winner in E_{NM} and there is **no** $a \neq p$ and $b \neq p$ such that $beat_{E_{NM}}(a, b) = beat_{E_{NM}}(b, a)$. Then the manipulator casts his vote v with respect to the greedy rule. This vote does not create a Condorcet winner that is different from p, hence it is optimum for both the Condorcet criterion and Borda rule.

Suppose there is the Condorcet winner in E_{NM}, $a \neq p$. If there is no b such that $beat_{E_{NM}}(a, b) = beat_{E_{NM}}(b, a) + 1$ then a is the winner regardless of the manipulator vote v. Therefore, suppose there exists a set B such that $beat_{E_{NM}}(a, b) = beat_{E_{NM}}(b, a) + 1$, $b \in B$. If there exists b such that $score_{E_{NM}}(a) \geq score_{E_{NM}}(b)$ then a will be ranked below b in the manipulator vote v that is constructed based on the greedy algorithm (or we can swap a and b if their scores are equal). Therefore,we assume that $score_{E_{NM}}(a) < score_{E_{NM}}(b)$. Let b^* be the candidate with the minimum score $score_{E_{NM}}$, so that $b^* = argmin_{b \in B}(score_{E_{NM}}(b))$. The manipulator must rank a below b^* to prevent a from being the Condorcet winner. This is equivalent to assuming that $score_{E_{NM}}(a) = score_{E_{NM}}(b^*)$ and using the greedy algorithm to construct the vote v. If v is a valid manipulation then we are done. If it is not then there is no way to construct a manipulation. ♡

6 Multiple ballots

So far, we have assumed that voters vote only once. However, the THEN combinator is naturally sequential. We can therefore consider the case where voters are allowed to re-vote in each round. For example, in the French presidential elections, voters re-vote in the second stage. Such re-voting increases the potential for manipulation in two ways. First, as we illustrate here, there are elections which can only be manipulated when the manipulators vote differently in the two rounds. Of course, all those elections where manipulators can change

the result by strategically voting the same way in both rounds remain manipulable. Second, as we also argue in the next section, the first round of voting reveals voters' preferences, thereby enabling manipulations to take place that require such knowledge. Third, voters man vote strategically in the first round to give their preferred candidate an easier contest in the second round.

If voters re-vote between rounds, we add "with re-voting" to its name. Hence, plurality with runoff and re-voting is the two stage election rule used in French presidential elections in which, unless there a majority in the first round, plurality is used in the first round to select two candidates to go through to the runoff, and voters then re-vote in the second round to decide the winner of the runoff. The following example demonstrates that there exist elections where strategic voting with plurality with runoff is not possible, but is with plurality with runoff and re-voting.

Example 1. *Suppose we have 2 votes for* (a, b, p)*, 2 votes for* (b, a, p)*, 1 vote for* (b, p, a)*, 2 votes for* (p, a, b) *and 2 manipulators whose preferences are* (p, a, b)*. In addition, we suppose in the event of a tie in the first round between all 3 candidates, the manipulators' peferred candidate* p *and* a *go through to the runoff. Note that if the manipulators vote truthfully, then* p *and* b *have the most votes in the first round, and* b *wins the runoff by 5-4. To make* p *the winner, the manipulators need* a *and* p *to be in the runoff. This is possible if and only if one of the manipulators votes for* a *and the other votes for* p *in the first round. We then have a 3-way tie and, according to the tie-breaking rule,* a *and* p *go through to the runoff. If the manipulators do not re-vote in the runoff,* a *wins the runoff by 5-4. On the other hand, if the manipulators can re-vote in the runoff, both can vote for* p*, and* p *will beat* a *by 5-4.*

7 Revealed preferences

One of the strong assumptions made in much work on (the complexity of) manipulation is that the manipulators know the other voters' preferences [9]. There are many situations where this is unrealistic. When we have re-voting, it is reasonable to suppose voters' preferences have been (partially) revealed by the first round of voting. This introduces new opportunities for manipulation. Consider Black's procedure with re-voting and a manipulator who lacks any knowledge of the other voters' preferences, so votes truthfully in the first round. The following example demonstrates that this manipulator can vote strategically in the second round based on the votes revealed in the first round.

Example 2. *Suppose the first round reveals that there are 2 votes for* (a, b, p)*, 2 votes for* (b, p, a)*, 1 vote for* (p, a, b)*, and a single manipulator's truthful vote for* (p, b, a)*. There is no Condorcet winner so all candidates go through to the second round. Without re-voting,* b *has the highest Borda score in the second round and is the overall winner. On the other hand, suppose the manipulator changes their vote in the second round to* (p, a, b) *based on the preferences revealed in the first round. Then, assuming the other votes remain the same, the Borda scores of all candidates are equal. If such a 3-way tie is broken in favour of the manipulator, then the manipulator's preferred candidate* p *now wins.*

It is natural to consider more game theoretic behaviours in such two stage voting rules. Re-voting can be viewed as a finite repeated sequential game so we can use concepts like subgame perfect Nash equilibrium and backward induction to predict how agents will play strategically in each round. An interesting open question is the computational complexity of computing such strategic behaviour. This sort of strategic voting has already received some attention in the literature. For example, Bag, Sabourian and Winter prove that a class of voting rules which use repeated ballots and eliminate one candidate in each round are Condorcet consistent [1]. They illustrate this class with the *weakest link* rule in which the candidate with the fewest ballots in each round is eliminated.

It is also natural to consider iterated voting in multiple stage voting rules. After each round of voting, we might suppose that agents change their vote according to a best response strategy, starting perhaps from a truthful vote. We can also consider the situation where the full preferences of the agents are revealed in each round of voting, as well as the situation where only partially information is revealed like total Borda scores. However, unlike previous studies like [14], candidates are also eliminated in each round.

8 Related work

As noted earlier, a number of well known voting rules like Black's procedure and Plurality with runoff are instances of this voting schema. However, there exist many other related voting rules. For example, Contizer and Sandholm [5] studied the impact on the computational complexity of manipulation of adding an initial round of the Cup rule to a voting rule X. This initial round eliminates half the candidates and makes manipulation NP-hard to compute for several voting rule including plurality and Borda. Consider the multi-winner voting rule, $Bisect$ which runs an election between given pairs of candidates, and returns the winning half of the candidates. Then Conitzer and Sandholm's study can be viewed as of the two stage voting rule $Bisect$THENX. Elkind and Lipmaa [11] extended this idea to a general technique for combining two voting rules. The first voting rule is run for some number of rounds to eliminate some of the candidates, before the second voting rule is applied to the candidates that remain. They proved that many such combinations of voting rules are NP-hard to manipulate.

Beside STV, Nanson's, Baldwin's and Coombs rule, a number of other recursively defined rules have been put forwards in the literature. For example, Tideman proposed the Alternative Smith rule [17]. This is recursively defined as $SmithSet$THEN$(PluralityLoser$THEN$AlternativeSmith)$.
Other complex multi-stage rules have also been proposed. For example, [13] has proposed a complex rule that computes the Schwartz choice set, then iteratively applies Copeland's procedure to this set until a fixed point is reached. If several candidates remain at this point, the rule then selects the plurality winners. If there are several such winners, the rule then chooses among then according to the number of second place votes, and so on. If this still does not select a winner, a lottery is then used amongst the candidates that remain.

We recently proposed a combinator for taking the consensus of two (or more) voting rules. Given two voting rules X and Y, the combinator $X+Y$ computes the winners of X and Y and then recursively applies $X + Y$ to this set. If X and Y are majority consistent (that is, given an election with just two candidates, they both return the majority winner) then $X+Y$ is $(X$OR$Y)$THEN$Majority$ where XORY returns the union of the winners of X and Y.

9 Conclusions

We have considered voting rules which have multiple stages. For example, Black's procedure selects the Condorcet winner if they exist, otherwise in the second stage, Black's procedure selects the

Borda winner. We denoted this as $Condorcet$THEN$Borda$. We have studied the computational complexity of computing a manipulation for such two stage procedures. In general, we argued that there is no connection between the computational complexity of manipulating the two stages of such a voting rule and that of the whole. However, we also demonstrated cases where the complexity of a base rule increases by adding a stage in front of it. In particular, whilst Plurality is polynomial to manipulate with weighted votes, $Condorcet$THEN$Plurality$ is NP-hard with 3 or more candidates and a coalition of manipulators. In fact, with any scoring rule X, computing a coalition manipulation of $Condorcet$THENX is NP-hard with 3 or more candidates and weighted votes. It follows that computing a coalition manipulation of Black's procedure is NP-hard. With unweighted votes, the complexity of manipulating Black's procedure is inherited from the Borda rule that it includes. More specifically, a single manipulator can compute a manipulation of Black's procedure in polynomial time, but computing a manipulation is NP-hard for two manipulators. There are many directions for future work. For instance, it would also be interesting to consider the impact of such two stage voting on other types of control, on bribery and on issues like the computation of possible winners.

Acknowledgements

NICTA is funded by the Department of Broadband, Communications and the Digital Economy, and the Australian Research Council. The authors are also supported by the Asian Office of Aerospace Research and Development (AOARD).

REFERENCES

[1] P.K. Bag, H. Sabourian, and E. Winter. Multi-stage voting, sequential elimination and condorcet consistency. *Journal of Economic Theory*, 144(3):1278 – 1299, 2009.

[2] J.J. Bartholdi and J.B. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[3] N. Betzler, R. Niedermeier, and G.J. Woeginger. Unweighted coalitional manipulation under the Borda rule is NP-hard. In T. Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*. International Joint Conference on Artificial Intelligence, 2011.

[4] F. Brandt, M. Brill, E. Hemaspaandra, and L.A. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In M. Fox and D. Poole, editors, *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*. AAAI Press, 2010.

[5] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of 18th IJCAI*, pages 781–788. International Joint Conference on Artificial Intelligence, 2003.

[6] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate. *Journal of the Association for Computing Machinery*, 54, 2007.

[7] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate. *Journal of the Association for Computing Machinery*, 54, 2007.

[8] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate. *Journal of the Association for Computing Machinery*, 54, 2007.

[9] V. Conitzer, T. Walsh, and L. Xia. Dominating manipulations in voting wih partial information. In W. Burgard and D. Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*. AAAI Press, 2011.

[10] J. Davies, N. Narodytska, and T. Walsh. Eliminating the weakest link: Making manipulation intractable? In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012)*. AAAI Press, 2012.

[11] E. Elkind and H. Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Proceedings of the 16th Annual International Symposium on Algorithms and Computation (ISAAC'05)*, 2005.

[12] P.C. Fishburn. Condorcet social choice functions. *SIAM Journal on Applied Mathematics*, 33(3):469–489, 1977.

[13] C.G. Hoag and G.H. Hallett. *Proportional Representation*. Macmillan, 1926.

[14] O. Lev and J.S. Rosenschein. Convergence of iterative voting. In *The Eleventh International Joint Conference on utonomous Agents and Multiagent Systems (AAMAS 2012)*, Valencia, Spain, June 2012.

[15] N. Narodytska, T. Walsh, and L. Xia. Manipulation of Nanson's and Baldwin's rules. In W. Burgard and D. Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*. AAAI Press, 2011.

[16] B. Reilly. Social choice in the south seas: electoral innovation and the borda count in the pacific island countries. *International Political Review*, 23(4):355–372, 2002.

[17] N. Tideman. *Collective Decisions And Voting: The Potential for Public Choice*. Ashgate, 2006.

Magenda: Doodle with Preferences

Davide Navarro[1] and **Maria Silvia Pini**[2] and **Francesca Rossi**[3] and **Kristen Brent Venable**[4]

Abstract. A typical real-life problem is deciding when to plan a meeting or a social event that involves several people to guarantee that most of the people will participate to the event. Doodle is a well-known web-based tool that helps people to solve this problem in a simple way, that avoids many phone calls and email messages. However, Doodle allows the users only to express the acceptable time-slots for the event, but it does not allow them to express their preferences over the acceptable options. In this paper, we consider this problem. We show a simple web-based application, called Magenda, that allows the participants to reveal their preferences over their acceptable options. Such preferences are then aggregated by the system to find the final decision via a voting rule that is selected by the organizer of the event.

1 Introduction

Deciding when to plan a meeting or a social event that involves several people is a typical problem in many scenarios. The goal in this problem is to find a time slot for the event which is acceptable for all the possible participants. Among all the acceptable options, it is desirable to find the one which is the most preferred by the greater number of potential participants.

Doodle [3] is a well-known web-based tool that helps scheduling meetings and other appointments. It is simple, quick, and free and it avoids many phone calls and email messages, that would be necessary to plan the meeting without using the automated tool. However, Doodle allows the users only to express the acceptable time slots, but not the preferences over the acceptable options. Therefore, it could happen that Doodle selects a time slot that is acceptable by all the participants, but that is not the most preferred by any of them.

Preferences are a central concept of decision making [17]. Many real-life problems contain statements which can be expressed as preferences. Such preferences can be of many kinds. They may be qualitative (as in "I like A more than B") or quantitative (as in "I like A at level 10 and B at level 11"). The problem of representing preferences of agents has been deeply investigated in Artificial Intelligence (AI) community in recent past years [18, 24, 21, 16, 15, 13]. Preference representation is an important issue when we have to represent the desires of users or to reason about them for example in recommender systems and in multi-agent settings.

In a multiagent scenario, agents generally have different preferences, and it can be important to aggregate these preferences, i.e., to select a collectively desirable candidate from a set of candidates. Candidates could be, for example, potential presidents, time slots, joint plans, allocations of goods or resources, etc. A general method for aggregating preferences in multi-agent systems, in order to take a collective decision, is to run an election among the different options using a voting rule [12].

Due to the importance that preferences have in decision making and in multi-agent settings and due to the simplicity and the spread of Doodle, we have decided to develop an online application to schedule meetings that is very similar to Doodle, in terms of simplicity and quickness, but that takes into account the central role of the users preferences. Our web-based application, called Magenda, allows the participants to reveal their preferences over acceptable options, and takes into account these preferences to select the final decision. Such preferences are aggregated by our automated system via a specific voting rule that is selected by the organizer of the event.

This is not the first attempt to extend standard Doodle. Doodle itself now has a version with three qualitative scores (yes, no and ?) and other systems were developed along the same line: Whale [10], developed by Sylvain Bouveret and The Decider [2], developed by Ronen Brafman and students.

The task considered in the paper can be regarded as a group recommendation in the field of recommender systems, where different group members usually have different preferences and this disagreement among members must be resolved [14, 11, 19, 23].

The paper is organized as follows. Section 2 presents the basic notions regarding Doodle and voting systems. Section 3 describes Magenda. More precisely, it shows the kinds of preferences that are allowed by Magenda, it shows step by step how to use Magenda, and it describes the technologies that have been used to develop Magenda. Section 4 summarizes the result shown in the paper and it gives some hints for future work.

2 Basic notions

First, we will show how the web-based tool Doodle works. Then, we will give a brief description of the voting rules that can be used to aggregate the users preferences.

2.1 Doodle

Doodle is an online free tool that can be used when you need to setup a meeting or a social event with more than one person but you don't want to send email messages or making phone calls. Doodle eliminates the constant back and forth communication allowing your inbox to contain only relevant e-mail.

Doodle is very easy to use for any kind of user. Due to this simplicity, many users usually adopt this web-based tool to plan meetings that involve several people.

[1] Department of Mathematics, University of Padova, Italy, email: navarrodavide@gmail.com
[2] Department of Information Engineering, University of Padova, Italy, email: pini@dei.unipd.it
[3] Department of Mathematics, University of Padova, Italy, email: frossi@math.unipd.it
[4] Department of Mathematics, University of Padova, Italy, email: kvenable@math.unipd.it

To plan an event in Doodle, a user must perform the following steps:

- He must go to the following url

$$http://www.doodle.com$$

(see Figure 1(a)).
- He must create an event by specifying his name, the event title, its location, and its description (see Figure 1(b)).
- He must select a set of possible dates for the event, via a calendar performed by JQuery (see Figure 1(c)).
- For every selected date, he must specify some possible times for scheduling the event (see Figure 1(d)).
- He can create a Doodle account and use it to send the invitations to the participants by simply specifying the email addresses of the participants in a text-area (see Figure 1(e)). At this point every participant receives an email containing the link to a web page similar to the one shown in Figure 1(f) where it is possible to select the acceptable time slots.

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 (or part of it) 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 [12], given a set of candidates C, a *profile* is a collection of total orderings over the set of candidates, one for each voter. Given a profile, a *voting rule* maps it onto a single winning candidate (if necessary, ties are broken appropriately). In this paper, we will often use a terminology which is more familiar to multi-agent settings: we will sometimes call "users" the voters, "options" the candidates, and "decision" or "best solution" the winning candidate.

Some examples of widely used voting rules, that we will study in this paper, are:

- **Plurality**: each voter states a single preferred candidate, and the candidate who is preferred by the largest number of voters wins;
- **Scoring**: given m candidates, each voter gives a ranking of all candidates, each candidate gets a score and the candidate with the greatest sum of scores wins. **Borda** is a scoring rule such that the i^{th} ranked candidate gets a score of $m - i$.
- **Approval**: given m candidates, each voter approves between 1 and $m - 1$ candidates, and the candidate with most votes of approval wins.
- **Rated**: each voter can give a score to every candidate and he can give the same score to various candidates. The candidate with the highest total score wins.

Approval is a simple way to vote, where the users must only select the options that they accept. Scoring rules are more sophisticated ways of voting where the users must reveal a total order over all the possible options. Rated voting rules require extra work to the users since they must reveal a score for every option.

3 Magenda: Doodle with preferences

When an organizer wants to plan an event which involves several people, he can use Doodle [3], as shown in Section 2.1. However, in Doodle it may happen that an event is planned in a time slot that is acceptable for every participant but that is not the most preferred by any participant. Our web-based application Magenda extends Doodle to overcome this drawback. In particular, it allows the users to express different kinds of preferences over the acceptable time slots, and it allows the organizer to decide how to aggregate these preferences by selecting a specific voting rule.

3.1 Allowed preferences

The developed application allows the users to express their preferences over the possible time slots (that we will call 'options') in four different ways. More precisely, the kinds of preferences that can be expressed in Magenda correspond to the following voting systems:

- **Yes/No**: This is the voting system used by Doodle except that a user can avoid to express his vote if no option is acceptable for him. In such a case the system selects the option which is the more accepted by the users. This voting system corresponds to Approval.
- **Scoring**: In this voting system every user expresses a ranking over all the options, and then the system associates a score to them as follows: if there are m options, the most preferred option receives a score of $m-1$, the second one receives a score of $m-2$, and so on. Moreover, every unacceptable option receives 0 points. The score of an option is the sum of the received scores. The systems selects the option which is accepted by the greater number of users and among these it selects the one that has the highest score. In such a way, the organizer of the event knows, for every participant, his acceptable options and for each of them his level of preference. This voting system corresponds to Borda.
- **Sum-Points**: In this voting system every user specifies a distribution of 100 points over the options. The score of an option is the sum of the received scores. The systems selects the option which is accepted by the greater number of users and among these it selects the one that has the highest score. This voting system is a Rated voting method.
- **Minimum-Points**: In this voting system the users specify a distribution of 100 points over the options. The score of an option is the minimum of the received scores. The systems selects the option which is accepted by the greater number of users and among these it selects the one that has the highest score. This voting system is a Rated voting method.

3.2 How to use magenda

To plan an event in Magenda, a user must perform the following steps:

- He must go to the following url

$$http://stagemeeting.altervista.org$$

(see Figure 2(a)).
- If the user is a new user of Magenda, he has to perform the registration (see Figure 2(b)). In particular, he has to insert his personal data, a username and a password that will allow him to exploit the functionalities of Magenda. After this step, the user will receive an e-mail containing the link to the web page that will activate his accout.
- If the user forgets his password, he simply need to specify his email-address (see Figure 2(c)) and then the system will send him via e-mail the link to a new web-page where he can state a new password (see Figure 2(d)).

- To create an event, the user first needs to insert his username and password, and then he must specify the following things (see Figure 2(e)):
  - the title of the event;
  - the description of the event;
  - some possible dates and times (also called 'options') for the event;
  - the duration of the event;
  - the e-mail addresses of the people that must participate to the event;
  - the voting rule that the system has to use to compute the best option.

The selected voting system is the rule that will be used by the system to select the better schedule for planning the event according to the preferences expressed by the invited people. The available voting systems in Magenda are Yes/No, Scoring, Sum-Points, and Minimum Points. Such voting rules, that have been described in Section 3.1, are described in the Magenda's guide (see Figure 2(f)).

After the creation of the event, the system automatically sends an email to all the people that have been invited to the event (see Figure 2(g)). This email contains a link to a web site where they have to specify their preferences on the possible dates and times for scheduling the event. Depending on the voting system selected by the organizer, the invited people can express their preferences over the possible options in different ways. For example, if the selected voting rule is Sum-Points every participant needs to distribute 100 points over all the possible options (see Figure 2(h)).

The user who has organized the event can see, via his area named 'My events', how many people have expressed their preferences over it. If he thinks that this number is sufficient to decide when to plan the event, he can terminate the process. At this point, the system sends an e-mail to every participant that indicates the date and time computed for the event. Such a choice is the best one according to the voting system selected by the organizer and according to the preferences expressed by the invited people.

3.3 Technologies

We have developed Magenda on Ubuntu 11.04. Ubuntu is a free operative system that is a GNU/Linux distribution which is based on GNOME [5]. To implement Magenda we have used HTML5 and CC3. To test the application, we have used the following browsers that are compatible with HTML5 and CC3:

- Safari 5.0 [9],
- Google Chrome 7.0 [6],
- Firefox 3.6 [4],
- Opera 10.63 [8],
- Internet Explorer [7].

To develop Magenda, we have used the HTML editor Bluefish [1], since it supports HTML, Javascripts and PHP, that we have used to create our web-based application. To public online Magenda, we have used the free service Altervista, which is an Italian web platform, that allows the user to create immediately a free web site.

More details regarding the technologies that have been used to develop Magenda can be found in [22].

4 Conclusions and future work

We have shown a web-based application for scheduling meetings and social events among several people. Such an application has the same advantages of the well-known tool Doodle, that is, it is easy to use, it is quick, and it is free. Moreover, it allows the users to reveal preferences over the acceptable options and exploits such preferences to find the better schedule.

This task can be regarded as group recommendations in the field of recommender system. In group recommendations however, because different group members usually have different preferences, this disagreement among members must be resolved. This problem has been studied in [14, 11, 19, 23]. It will be interesting to discuss whether the related algorithms could achieve the goal and even perform better as it aimed at solving the disagreement challenge. We intend also to integrate in a suitable way Magenda with the calendars of the users. This can be useful to show to the invited people, if they desire it, only the options that are not already occupied in their calendar. Moreover, we plan to add to our tool other kinds of voting systems to aggregate user preferences, that are difficult to manipulate. An example could be Plurality with runoff, that requires the users to vote with Plurality first over all the possible options and then only on the two options that have obtained the greater number of votes. Moreover, we intend to include in the tool the possibility of using rules that are difficult to compute, such as for example the Kemeny rule [20]. It would be interesting also to study in a real-life scenario if users are willing to take the extra work to assign preference score to options expecially when there is a large number of choices.

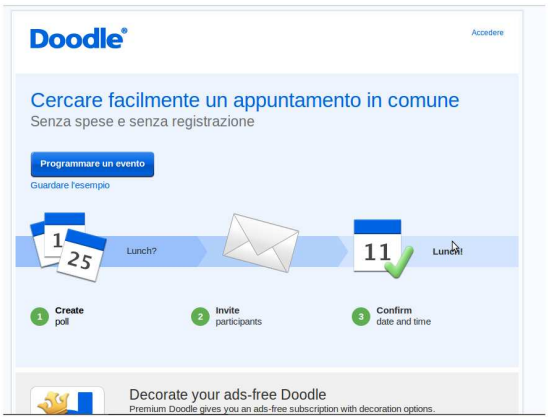
5 Acknowledgments

This work has been partially supported by the MIUR PRIN 20089M932N project 'Innovative and multi-disciplinary approaches for constraint and preference reasoning'.

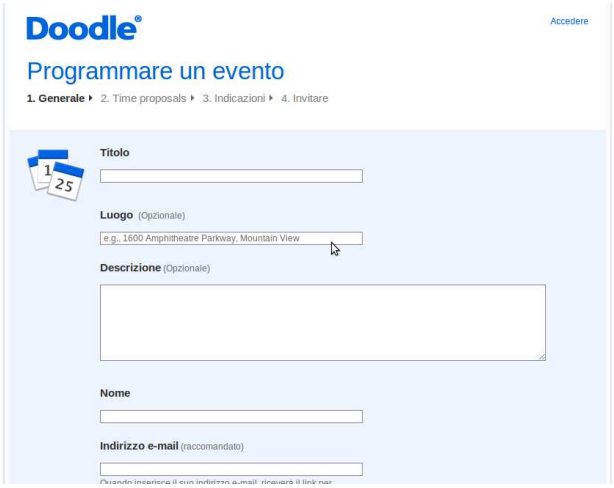
REFERENCES

[1] Bluefish, http://bluefish.openoffice.nl.
[2] The decider, http://apps/facebook.com/the_decider.
[3] Doodle, http://www.doodle.com.
[4] Firefox, http://www.mozilla-europe.org/it/firefox.
[5] Gnome, http://www.gnome.org.
[6] Google chrome, http://www.google.com/chrome.
[7] Internet explorer, http://windows.microsoft.com/it-it/internet-explorer/products/ie-9/home.
[8] Opera, http://www.opera.com.
[9] Safari, http://www.apple.com/it/safari.
[10] Whale, http://whale.noiraudes.net/en.
[11] S. Amer-Yahia, S. B. Roy, A. Chawla, G. Das, and C. Yu, 'Group recommendation: Semantics and efficiency', *PVLDB*, **2**(1), 754–765, (2009).
[12] K. J. Arrow and A. K. Sen amd K. Suzumura, *Handbook of Social Choice and Welfare.*, North-Holland, Elsevier, 2002.
[13] F. Bacchus and A. J. Grove, 'Utility independence in a qualitative decision theory', in *KR-96*, pp. 542–552, (1996).
[14] L. Baltrunas, T. Makcinskas, and F. Ricci, 'Group recommendations with rank aggregation and collaborative filtering', in *Proc. RecSys'10*, pp. 119–126. ACM, (2010).
[15] S. Benferhat, D. Dubois, and H. Prade, 'Towards a possibilistic logic handling of preferences', *Appl. Intell.*, **14**(3), 303–317, (2001).
[16] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole, 'Reasoning with conditional ceteris paribus preference statements', in *Proceedings of UAI-99*, eds., K. B. Laskey and H. Prade, pp. 71–80. Morgan Kaufmann, (1999).
[17] C. Domshlak, E. Hüllermeier, S. Kaci, and H. Prade, 'Preferences in ai: An overview', *Artif. Intell.*, **175**(7-8), 1037–1052, (2011).

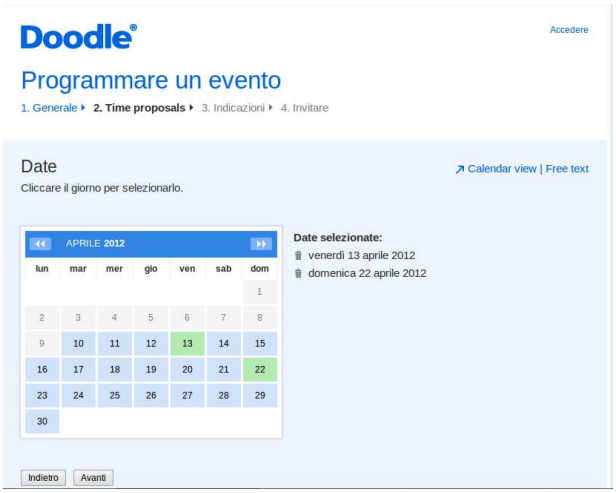
[18] J. Doyle, Y. Shoham, and M. P. Wellman, 'A logic of relative desire (preliminary report)', in *Proceedings of ISMIS-91*, volume 542 of *LNCS*, pp. 16–31. Springer, (1991).
[19] A. Jameson and B. Smyth, 'Recommendation to groups', in *The Adaptive Web*, volume 4321 of *LNCS*, pp. 596–627. Springer, (2007).
[20] J. G. Kemeny, 'Mathematics without numbers', *Daedalus*, **88**, 571–591, (1959).
[21] J. Lang, 'Conditional desires and utilities: an alternative logical approach to qualitative decision theory', in *Proceedings of ECAI-96*, pp. 318–322, (1996).
[22] D. Navarro, *Un'estensione di Doodle: Magenda*, Graduation Thesis, University of Padova, 2012.
[23] M. O'Connor, D. Cosley, J. A. Konstan, and J. Riedl, 'Polylens: A recommender system for groups of user', in *Proc. ECSCW'01*, pp. 199–218. Kluwer, (2001).
[24] M. P. Wellman and J. Doyle, 'Preferential semantics for goals', in *Proceedings of AAAI*, pp. 698–703, (1991).



(a)



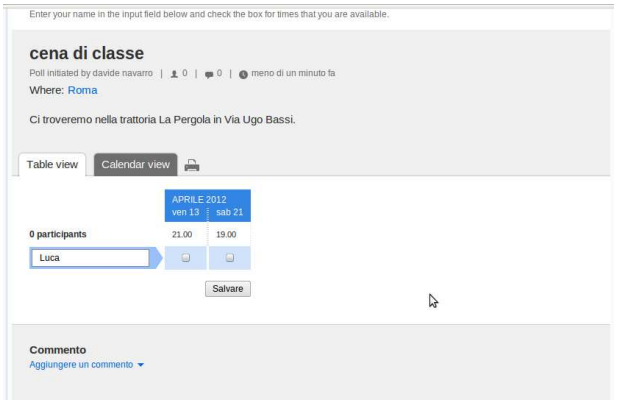
(b)



(c)

(d)

(e)



(f)

Figure 1. Doodle.

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

Figure 2. Magenda.

From Preferences over Objects to Preferences over Concepts

Sergei Obiedkov[1]

Abstract. We present an approach to preference modeling and learning preferences from data based on formal concept analysis. We consider techniques to derive preferences over attribute subsets from preferences over objects, including ceteris paribus preferences.

1 INTRODUCTION

If John chooses a strawberry over an apple, would he choose a raspberry over a pear? If so, is it because, for Peter, red berries taste better than tree fruit or are there other factors involved? More generally, given a number of alternatives each described by a set of elementary features together with a preference relation over these alternatives, we would like to derive preferences among feature sets that would explain, at least partially, the observed preferences over individual alternatives.

A move from a strawberry and a raspberry to red berries is a move from individual objects to concepts; thus, our aim is to generalize from preferences over objects to preferences over concepts. A concept can be understood *extensionally*, through objects it covers, or *intensionally*, through attributes that define it. This duality reflects itself in preferences over object sets and preferences over attribute sets, so that the latter can be defined in terms of the former. The first step is then to get from preferences over objects to preferences over object sets, and for this there are various options extensively studied in, e.g., preference logics [21]. We pick up two such options and see what consequences they have for preferences over attribute sets.

In terms of preference logics, attributes can be regarded as atomic propositions and attribute sets as atomic conjunctions. Thus, what we present here is a simple version of propositional preference logic where only preferences over conjunctions of atomic formulae can be expressed. This limitation allows us to make a link to formal concept analysis (FCA) [12], through which we develop techniques for learning preferences from empirical data. FCA provides a wide range of computational tools, and, despite their often unattractive theoretical complexity, they are successfully used in practical data analysis [8].

The paper is organized as follows. We start by describing the types of preferences discussed throughout the paper including two types of global preferences and *ceteris paribus* preferences. We proceed with FCA definitions and then consider each type of preferences separately providing FCA-based semantics, discussing inference, and describing methods for learning preferences from data. Finally, we show a way to take into account the conceptual structure of the data and thus reduce the learning bias down to a well-defined point.

[1] National Research University Higher School of Economics, Moscow, Russia, email: sergei.obj@gmail.com

2 PREFERENCES IN PREFERENCE LOGICS

In modal preference logics [17, 21], preference relations are modeled by accessibility relations on possible worlds, which correspond to alternatives being compared. The preference relation is often assumed to be a preorder, that is, reflexive and transitive. We stick to this assumption and denote this relation by $\leq$. It can be extended to sets of possible worlds in several ways, of which we consider two.

In von Wright's version of preference logic [22], a set Y is preferred to a set X (notation: $X \trianglelefteq_\forall Y$) if

$$\forall x \in X \forall y \in Y (x \leq y), \tag{1}$$

that is, every alternative in Y is preferred to every alternative in X. The induced relation $\trianglelefteq_\forall$ is not necessarily reflexive or irreflexive: reflexivity is violated by a set containing two incomparable alternatives, while $\{x\} \trianglelefteq_\forall \{x\}$ for every single-element set $\{x\}$. Since $X \trianglelefteq_\forall \varnothing$ and $\varnothing \trianglelefteq_\forall Y$ for all X and Y, transitivity is not preserved, either. However, this is the only way transitivity may fail; by disallowing the empty set, we obtain a transitive relation. Besides, the $\trianglelefteq_\forall$ relation can be easily transformed into a strict partial order:

$$X \triangleleft_\forall Y \iff X \trianglelefteq_\forall Y \text{ and } Y \not\trianglelefteq_\forall X.$$

A different approach is to state that Y is preferred to X if, for each alternative from X, Y contains an alternative that is at least as good:

$$\forall x \in X \exists y \in Y (x \leq y).$$

We denote this by $X \trianglelefteq_\exists Y$. For some contexts, $\trianglelefteq_\exists$-preferences are more appropriate than $\trianglelefteq_\forall$-preferences. Consider the case of a two-person turn-based game. If X and Y are sets of positions reachable from the current position in one turn, preferences between X and Y are $\trianglelefteq_\exists$-like for the player whose turn it is, since this player has control over which position from X or Y gets chosen. For the other player, $\trianglelefteq_\forall$-preferences are more appropriate.

Preferences over propositions are defined as preferences over their sets of models. Thus, ϕ is preferred to ψ if every model of ϕ is preferred to every model of ψ (for $\trianglelefteq_\forall$-preferences) or, for every model of ψ, there is a "better" model of ϕ (for $\trianglelefteq_\exists$-preferences).

With a pinch of salt (ignoring empty sets and inconsistent propositions), $\trianglelefteq_\exists$-preferences can be viewed as a relaxation of $\trianglelefteq_\forall$-preferences, but both types are *global* in that propositions are compared w.r.t. all their models. *Ceteris paribus* preferences put restrictions on which models should be taken into account by assuming "other things being equal" when comparing ϕ and ψ. We might want to explicitly specify which other things must be equal. In the version of preference logic from [21], this is done by parameterizing the modal operator corresponding to the preference relation by a set of

propositions Γ. The Γ-ceteris paribus version of $X \trianglelefteq_\forall Y$, which we denote by $X \trianglelefteq_\Gamma Y$, holds if

$$\forall x \in X \forall y \in Y (\forall \varphi \in \Gamma (x \models \varphi \iff y \models \varphi) \rightarrow x \leq y),$$

where $x \models \varphi$ means that φ is true in x. Thus, it is required that every alternative from Y is preferred to every alternative from X that satisfies exactly the same formulae from Γ. Clearly, this is a relaxation of the requirement specified by (1).

Interestingly, adding the ceteris paribus condition to the definition of $\trianglelefteq_\exists$-preferences results in stronger preferences. To say that $X \trianglelefteq_\exists Y$ holds ceteris paribus, we must find, for each alternative $x \in X$, an alternative in Y that is not only at least as good, but that is also sufficiently similar: it should satisfy exactly the same propositions from Γ that x does.

In this paper, we consider global $\trianglelefteq_\forall$- and $\trianglelefteq_\exists$-preferences and the ceteris paribus version of $\trianglelefteq_\forall$-preferences. Our discussion is restricted to rather simple propositions: we state preferences only over atomic conjunctions and allow only sets of atomic formulae as ceteris paribus conditions. Thus, in the ceteris paribus case, we consider preferences of the form $\phi \preccurlyeq_\Gamma \psi$, where ϕ and ψ are atomic conjunctions and Γ is a set of atomic formulae. We will work with ϕ, ψ, and Γ as with sets of attributes rather than as with logical formulae. The next section introduces formal concept analysis, which is the framework that we will use here.

3 FORMAL CONCEPT ANALYSIS

We start with a few definitions from FCA [12]. Given a *(formal) context* $\mathbb{K} = (G, M, I)$, where G is called a set of *objects*, M is called a set of *attributes*, and the binary relation $I \subseteq G \times M$ specifies which objects have which attributes, the *derivation operators* $(\cdot)^I$ are defined for $A \subseteq G$ and $B \subseteq M$ as follows:

$$A^I = \{m \in M \mid \forall g \in A (gIm)\}$$
$$B^I = \{g \in G \mid \forall m \in B (gIm)\}$$

A^I is the set of attributes shared by objects of A, and B^I is the set of objects having all attributes of B. Often, $(\cdot)'$ is used instead of $(\cdot)^I$. The double application of $(\cdot)'$ is a closure operator: $(\cdot)''$ is extensive, idempotent, and increasing. Sets A'' and B'' are said to be *closed*.

The left-hand side of Fig. 1 shows a context where objects are lunch options and attributes are menu items.[2] For instance, l_3 corresponds to the choice of pumpkin soup, vegetables, and ice cream.

A *(formal) concept* of the context (G, M, I) is a pair (A, B), where $A \subseteq G$, $B \subseteq M$, $A = B'$, and $B = A'$. In this case, A and B are closed. The set A is called the *extent* and B is called the *intent* of the concept (A, B). A concept (A, B) is *less general* than (C, D) if $A \subseteq C$. The set of all concepts ordered by this generality relation forms a lattice, called the *concept lattice* of the context $\mathbb{K}$.

A line diagram of the concept lattice of the context from Fig. 1 is shown in Fig. 2. Nodes correspond to concepts, with more general concepts placed above less general ones. Two concepts are connected with a line if one is less general than the other and there is no concept between the two. The extent of a concept can be read off by looking at the labels immediately below the corresponding node and below all nodes reachable by downward arcs. The intent consists of attributes indicated just above the node and those above nodes reachable by upward arcs. For example, the top-right node corresponds

[2] The example is inspired by the menu of the Parisian restaurant *Derrière*: `http://derriere-resto.com/restaurant/paris/derriere/menus/`. We use only a small part of the menu, though.

	pumpkin soup	black carrots	pork	vegetables	ice cream	chocolate mousse
l_1	×			×		×
l_2		×		×		×
l_3	×			×	×	
l_4		×	×			×
l_5	×		×			×

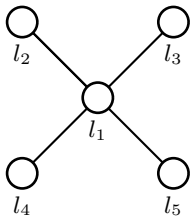


Figure 1. A preference context of lunch options

to the concept of all lunch options with pumpkin soup as a starter (l_1, l_3, and l_5). The node just below corresponds to its subconcept $(\{l_1, l_3\}, \{\text{pumpkin soup}, \text{vegetables}\})$.

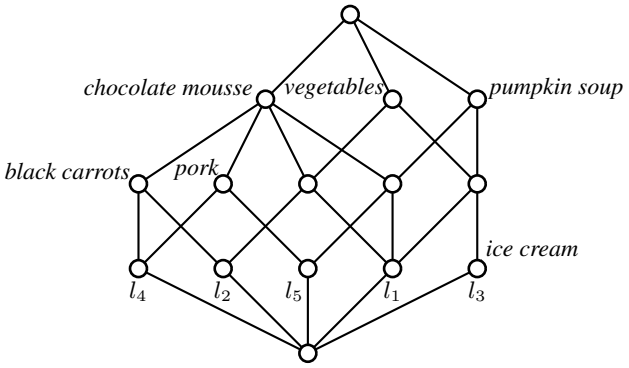


Figure 2. The concept lattice of the context in Fig. 1

It can be seen from this diagram that, if someone wants an ice cream for a dessert, she will have to have pumpkin soup as a starter and vegetables as the main dish. This is captured by the notion of an *implication*, which is, formally, an expression $A \rightarrow B$, where $A, B \subseteq M$ are attribute subsets. It *holds* or *is valid* in the context $\mathbb{K}$ (notation: $\mathbb{K} \models A \rightarrow B$) if $A' \subseteq B'$, i.e., every object of the context with all attributes from A also has all attributes from B.

If $A' = \varnothing$, then $(G, M, I) \models A \rightarrow M$. We use special notation for such zero-support implications: $A \rightarrow \bot$. Note that $A \rightarrow \bot$ is a headless Horn clause, whereas an implication $A \rightarrow B$ is a conjunction of definite Horn clauses with the same body.

The implications valid in a context are summarized by the Duquenne–Guigues basis [13], which has the minimal number of implications among equivalent implication sets. Nevertheless, it may be exponential in the size of the context, and determining its size is a #P-complete problem [16]. Other valid implications can be obtained from this basis using the Armstrong rules [4], which constitute a sound and complete inference system for implications.

The *preference context* $\mathbb{P} = (G, M, I, \leq)$ is defined in [18] as a context (G, M, I) supplied with a reflexive and transitive (as it is common in preference logics [21]) preference relation $\leq$ on G. We write $g < h$ if $g \leq h$ and $h \not\leq g$. The right-hand side of Fig. 1 shows preferences over lunch options: l_1 is better than l_4 and l_5, but worse than l_2 and l_3, while l_2 and l_3 are incomparable, as are l_4 and l_5. A

preference context can be regarded as a combination of two formal contexts: (G, M, I) and $(G, G, \leq)$. We use $(\cdot)'$ for the derivation operators of (G, M, I) and $(\cdot)^{\leq}$ and $(\cdot)^{\geq}$ for the derivation operators of $(G, G, \leq)$: $X^{\leq}$ ($X^{\geq}$) is the set of all objects that are at least (at most) as good as all objects from $X \subseteq G$.

4 MODELING PREFERENCES IN FCA

In Sects. 4.1 and 4.2, we recall (without proofs) results from [18] concerning preferences based on the relations $\trianglelefteq_\forall$ and $\trianglelefteq_\exists$. In Sect. 4.3, we present a new approach to modeling ceteris paribus preferences.

We define semantics for preferences by describing conditions under which a preference π is said to be valid in a preference context $\mathbb{P}$, denoted by $\mathbb{P} \models \pi$. We say that a preference π *follows from* (or *is a semantic consequence of*) a set of preferences Π (notation: $\Pi \models \pi$) if, whenever all preferences from Π are valid in some preference context $\mathbb{P}$ (Π is *sound* for $\mathbb{P}$; $\mathbb{P} \models \Pi$), the preference π is also valid in $\mathbb{P}$ ($\mathbb{P} \models \pi$). A set Π of preferences (of a certain kind) is said to be *complete* for $\mathbb{P}$ if, for all preferences π (of this kind), $\mathbb{P} \models \pi$ if and only if $\Pi \models \pi$. If, in addition, none of the preferences in Π follows from the other preferences, we say that Π is a *preference basis* of $\mathbb{P}$.

4.1 Universal preferences

It is possible to summarize $\trianglelefteq_\forall$-preferences over subsets of G by the concept lattice of the formal context $(G, G, \leq)$. Indeed, $X \trianglelefteq_\forall Y$ holds for $X, Y \subseteq G$ if and only if $Y \subseteq X^{\leq}$. Sets X and Y are maximal with respect to this property if and only if (X, Y) is a formal concept of $(G, G, \leq)$. At the same time, if $X \trianglelefteq_\forall Y$, then $U \trianglelefteq_\forall V$ for every $U \subseteq X$ and $V \subseteq Y$. Thus, concepts of $(G, G, \leq)$ provide a complete representation of $\trianglelefteq_\forall$-preferences over object sets.

Having defined preferences over object sets, there is an easy way to translate the definition into preferences over attribute sets by associating each attribute set A with the set of objects that have all attributes from A or, to put it in terms of formal concept analysis, with A':

Definition 1. *A set of attributes $B \subseteq M$ is* universally preferred *to a set of attributes $A \subseteq M$ in a preference context $\mathbb{P} = (G, M, I, \leq)$ if $A' \trianglelefteq_\forall B'$, i.e.,*

$$\forall x \in A' \forall y \in B' (x \leq y).$$

Notation: $\mathbb{P} \models A \preccurlyeq_\forall B$.

That is, $A \preccurlyeq_\forall B$ holds (or is valid) in $(G, M, I, \leq)$ if every object with all attributes from B is preferred to every object with all attributes from A. This is precisely the approach used in preference logics as described in Sect. 2.

It is easy to obtain the following characterization of universal preferences in terms of the derivation operators of the preference context:

Proposition 1. *$\mathbb{P} \models A \preccurlyeq_\forall B$ if and only if $B' \subseteq A'^{\leq}$.*

To give an example, in the preference context from Fig. 1, we have $\{\text{pork}\} \preccurlyeq_\forall \{\text{vegetables}\}$, since every option with vegetables is preferred to every option with pork.

Proposition 2. *A sound and complete inference system for universal preferences consists of a single rule:*

$$\frac{A \preccurlyeq_\forall B}{A \cup C \preccurlyeq_\forall B \cup D},$$

which allows one to add arbitrary attributes to both sides of a valid preference.

A universal preference basis of $\mathbb{P}$ can be found by representing universal preferences of $\mathbb{P}$ as implications in another formal context.

Definition 2. *Let $\mathbb{P} = (G, M, I, \leq)$ be a preference context. The* universal translation *of $\mathbb{P}$ is a formal context $\mathbb{K}^{\mathbb{P}}_\forall = (G \times G, (M \times \{1, 2\}) \cup \{\leq\}, I_\forall)$, where*

$$\begin{array}{lcl} (g_1, g_2) I_\forall m_1 & \iff & g_1 I m, \\ (g_1, g_2) I_\forall m_2 & \iff & g_2 I m, \\ (g_1, g_2) I_\forall \leq & \iff & g_1 \leq g_2. \end{array}$$

Here, m_1 and m_2 stand for $(m, 1)$ and $(m, 2)$ respectively, $m \in M$. We denote the derivation operators of $\mathbb{K}^{\mathbb{P}}_\forall$ by $(\cdot)^\forall$.

$T_\forall(A \preccurlyeq_\forall B)$, the translation of a universal preference *$A \preccurlyeq_\forall B$, is the implication*

$$(A \times \{1\}) \cup (B \times \{2\}) \to \{\leq\}$$

of the formal context $\mathbb{K}^{\mathbb{P}}_\forall$.

Proposition 3. *A universal preference $A \preccurlyeq_\forall B$ is valid in a preference context $\mathbb{P} = (G, M, I, \leq)$ if and only if its translation is valid in $\mathbb{K}^{\mathbb{P}}_\forall$:*

$$\mathbb{P} \models A \preccurlyeq_\forall B \quad \iff \quad \mathbb{K}^{\mathbb{P}}_\forall \models T_\forall(A \preccurlyeq_\forall B).$$

The context resulting from the translation has as its objects pairs of objects of the preference context and contains two copies of each original attribute; (g_1, g_2) is associated with the first copy of m if g_1 has m and with the second copy if g_2 has m. For example, the preference $\{\text{pork}\} \preccurlyeq_\forall \{\text{vegetables}\}$ valid in the preference context $\mathbb{P}$ from Fig. 1 is translated into $\{(\text{pork}, 1), (\text{vegetables}, 2)\} \to \{\leq\}$, which is a valid implication of $\mathbb{K}^{\mathbb{P}}_\forall$.

The following proposition describes the basis of universal preferences:

Proposition 4. *Let $\mathbb{P}$ be a preference context. The set*

$$\Sigma = \{A \preccurlyeq_\forall B \mid (A \times \{1\}) \cup (B \times \{2\}) \text{ is minimal w.r.t. } \mathbb{K}^{\mathbb{P}}_\forall \models (A \times \{1\}) \cup (B \times \{2\}) \to \{\leq\}\}$$

is the minimal (in the number of preferences) basis of the universal preferences valid in $\mathbb{P}$.

In other words, to compute the basis of universal preferences of $\mathbb{P}$, we need to find minimal (by set-inclusion) attribute sets of $\mathbb{K}^{\mathbb{P}}_\forall$ that have $\leq$ in their closure. Note that this can be done without explicit construction of $\mathbb{K}^{\mathbb{P}}_\forall$.

4.2 Existential preferences

In this section, we transfer the definition of $\trianglelefteq_\exists$-preferences to attribute sets similarly to how it was done for $\trianglelefteq_\forall$- preferences:

Definition 3. *A set of attributes $B \subseteq M$ is* existentially preferred *to a set of attributes $A \subseteq M$ in a preference context $\mathbb{P} = (G, M, I, \leq)$, denoted by $\mathbb{P} \models A \preccurlyeq_\exists B$, if $A' \trianglelefteq_\exists B'$, i.e.,*

$$\forall x \in A' \exists y \in B' (x \leq y).$$

Again, we can characterize existential preferences in terms of the derivation operators of the preference context:

Proposition 5. *$\mathbb{P} \models A \preccurlyeq_\exists B$ if and only if $A' \subseteq \bigcup_{g \in B'} g^{\geq}$.*

An example of an existential preference that does not hold universally in the context from Fig. 1 is $\varnothing \preccurlyeq_\exists \{\text{vegetables}\}$: for every lunch option, there is one with vegetables that is at least as good. On the other hand, {pumpkin soup, vegetables} is preferred to {black carrots, pork} both universally and existentially.

Existential preferences generalize implications:

Proposition 6. *For a preference context $\mathbb{P} = (G, M, I, \leq)$*

1. *If $(G, M, I) \models A \to B$, then $\mathbb{P} \models A \preccurlyeq_\exists B$.*
2. *If $\leq$ is the identity relation and $\mathbb{P} \models A \preccurlyeq_\exists B$, then $(G, M, I) \models A \to B$.*

Proposition 7. *A system of three rules*

$$\frac{}{X \preccurlyeq_\exists X}, \qquad \frac{X \preccurlyeq_\exists Y \cup U}{X \cup V \preccurlyeq_\exists Y}, \qquad \frac{X \preccurlyeq_\exists Y, \quad Y \preccurlyeq_\exists Z}{X \preccurlyeq_\exists Z}$$

is sound and complete with respect to existential preferences.

As universal preferences, existential preferences can also be translated into implications of a formal context, although the translation is of exponential size compared to the size the preference context.

Definition 4. *The* existential translation *of a preference context $\mathbb{P} = (G, M, I, \leq)$ is a formal context $\mathbb{K}^{\mathbb{P}}_\exists = (G, \mathfrak{P}(M), I_\exists)$, where $\mathfrak{P}(M)$ is the power set of M and*

$$g I_\exists A \quad \iff \quad g^{\leq} \cap A' \neq \varnothing.$$

Definition 5. *The* translation of an existential preference $A \preccurlyeq_\exists B$, *denoted by $T_\exists(A \preccurlyeq_\exists B)$, is the implication*

$$\{A\} \to \{B\}$$

of the formal context $\mathbb{K}^{\mathbb{P}}_\exists$.

Thus, existential preferences are translated into implications with single-element premises and conclusions (both elements are sets of original attributes). Such translation preserves the validity:

Proposition 8. *An existential preference $A \preccurlyeq_\exists B$ is valid in a preference context $\mathbb{P}$ if and only if its translation is valid in $\mathbb{K}^{\mathbb{P}}_\exists$:*

$$\mathbb{P} \models A \preccurlyeq_\exists B \quad \iff \quad \mathbb{K}^{\mathbb{P}}_\exists \models T_\exists(A \preccurlyeq_\exists B).$$

The set $\{A \preccurlyeq_\exists B \mid A$ is minimal and B is maximal w.r.t. $\mathbb{K}^{\mathbb{P}}_\exists \models \{A\} \to \{B\}\}$ is sound and complete (but possibly redundant) for $\mathbb{P}$.

Clearly, the existential translation of $(G, M, I, \leq)$ is infeasible for all but very small M. However, the representation size can be reduced by making use of the dependencies in the data. We address this issue in Sect. 5.

4.3 Ceteris paribus preferences

We now turn to context-based semantics for the ceteris paribus version of universal ($\trianglelefteq_\forall$) preferences, as described in Sect. 2.

Definition 6. *A set of attributes $B \subseteq M$ is* preferred ceteris paribus *to a set of attributes $A \subseteq M$ with respect to a set of attributes $C \subseteq M$ in a preference context $\mathbb{P} = (G, M, I, \leq)$ if $A' \trianglelefteq_C B'$, i.e.,*

$$\forall g \in A' \forall h \in B' (\{g\}' \cap C = \{h\}' \cap C \to g \leq h).$$

In this case, we say that the ceteris paribus preference $A \preccurlyeq_C B$ is valid *in $\mathbb{P}$.*

The preference $\{\text{chocolate mousse}\} \preccurlyeq_{\{\text{pumpkin soup}\}} \{\text{ice cream}\}$ holds in the context from Fig. 1 even though ice cream is preferred to chocolate mousse neither universally nor existentially.

Definition 7. *The* ceteris paribus translation *of $\mathbb{P} = (G, M, I, \leq)$ is a formal context $\mathbb{K}^{\mathbb{P}}_\sim = (G \times G, (M \times \{1, 2, 3\}) \cup \{\leq\}, I_\sim)$, where*

$$\begin{array}{lcl} (g_1, g_2) I_\sim (m, 1) & \iff & g_1 I m, \\ (g_1, g_2) I_\sim (m, 2) & \iff & g_2 I m, \\ (g_1, g_2) I_\sim (m, 3) & \iff & \{g_1\}' \cap \{m\} = \{g_2\}' \cap \{m\}, \\ (g_1, g_2) I_\sim \leq & \iff & g_1 \leq g_2. \end{array}$$

We denote the derivation operators of $\mathbb{K}^{\mathbb{P}}_\sim$ by $(\cdot)^\sim$.

$T_\sim(A \preccurlyeq_C B)$, the translation of a ceteris paribus preference *$A \preccurlyeq_C B$, is the implication*

$$(A \times \{1\}) \cup (B \times \{2\}) \cup (C \times \{3\}) \to \{\leq\}$$

of the formal context $\mathbb{K}^{\mathbb{P}}_\sim$.

This is similar to the universal translation, but here we have three copies of each original attribute. We associate (g_1, g_2) with the third copy of m if either both g_1 and g_2 have m or neither of them does.

Proposition 9. *$A \preccurlyeq_C B$ is valid in a preference context $\mathbb{P} = (G, M, I, \leq)$ if and only if its translation is valid in $\mathbb{K}^{\mathbb{P}}_\sim$:*

$$\mathbb{P} \models A \preccurlyeq_C B \quad \iff \quad \mathbb{K}^{\mathbb{P}}_\sim \models T_\sim(A \preccurlyeq_C B).$$

Proof. Suppose that $\mathbb{P} \models A \preccurlyeq_C B$ and $(A \times \{1\}) \cup (B \times \{2\}) \cup (C \times \{3\}) \subseteq (g_1, g_2)^\sim$ for some $g_1 \in G$ and $g_2 \in G$. Then, $A \subseteq \{g_1\}'$, $B \subseteq \{g_2\}'$, and $g_1 I c$ if and only if $g_2 I c$ for all $c \in C$. The latter means that $\{g_1\}' \cap C = \{g_2\}' \cap C$. Since $A \preccurlyeq_C B$ holds in $\mathbb{P}$, we have $g_1 \leq g_2$ and $(g_1, g_2) I_\sim \leq$ as required.

Conversely, assume $\mathbb{K}^{\mathbb{P}}_\sim \models (A \times \{1\}) \cup (B \times \{2\}) \cup (C \times \{3\}) \to \{\leq\}$. We need to show that $g_1 \leq g_2$ whenever $A \subseteq \{g_1\}'$, $B \subseteq \{g_2\}'$, and $\{g_1\}' \cap C = \{g_2\}' \cap C$. Indeed, in this case, we have $(A \times \{1\}) \cup (B \times \{2\}) \cup (C \times \{3\}) \subseteq \{(g_1, g_2)\}^\sim$ and, consequently, $(g_1, g_2) I_\sim \leq$, i.e., $g_1 \leq g_2$. □

Definition 8. *We say that a ceteris paribus preference $A \preccurlyeq_C B$ is in* canonical form *if $A \cap B = A \cap C = B \cap C$.*

For every preference $A \preccurlyeq_C B$, there is a unique preference in canonical form equivalent to $A \preccurlyeq_C B$ in the sense that it holds precisely in the same preference contexts:

$$A \cup (B \cap C) \preccurlyeq_{C \cup (A \cap B)} B \cup (A \cap C).$$

Proposition 10. *Let $\mathbb{P}$ be a preference context. The set*

$$\Pi = \{A \preccurlyeq_C B \mid (A \times \{1\}) \cup (B \times \{2\}) \cup (C \times \{3\}) \text{ is minimal}$$

$$\text{w.r.t. } \mathbb{K}^{\mathbb{P}}_\sim \models T_\forall(A \preccurlyeq_C B) \text{ and } A \cap B = A \cap C = B \cap C\}$$

is sound and complete for $\mathbb{P}$.

Proof. Due to Proposition 9, all ceteris paribus preferences from Π are valid in $\mathbb{P}$. To see that Π is complete, we consider, without loss of generality, an arbitrary preference $A \preccurlyeq_C B$ in the canonical form. If $\mathbb{P} \models A \preccurlyeq_C B$, then the implication $T_\forall(A \preccurlyeq_C B)$ holds and, therefore, either $A \preccurlyeq_C B \in \Pi$ or there are smaller sets $A_1 \subseteq A, B_1 \subseteq B$, and $C_1 \subseteq C$ such that $A_1 \preccurlyeq_{C_1} B_1 \in \Pi$. It is not hard to see that $\Pi \models A \preccurlyeq_C B$ holds then. □

We will not describe an inference system for ceteris paribus preferences similar to those provided by Propositions 2 and 7. Instead, we give an algorithm that decides whether a preference $A \preccurlyeq_C B$ follows from a set of preferences Π. By replacing A, B, and C in a valid preference $A \preccurlyeq_C B$ by their arbitrary supersets, we get valid preferences (cf. Proposition 2). The next definition captures preferences that can be obtained from other preferences in this way:

Definition 9. *Let* Π *be a set of ceteris paribus preferences. Then*

$$\Pi^\bullet = \{D \preccurlyeq_F E \mid \exists A \preccurlyeq_C B \in \Pi (A \subseteq D, B \subseteq E, C \subseteq F)\}.$$

Note that $\Pi \models \Pi^\bullet$. However, not all preferences that follow from Π are in $\Pi^\bullet$.

Proposition 11. *Let* Π *be a set of ceteris paribus preferences over* M*. For any preference* $A \preccurlyeq_C B$ *in canonical form, we have* $\Pi \models A \preccurlyeq_C B$ *if and only if* $\Pi^\bullet$ *contains all canonical-form preferences* $D \preccurlyeq_F E$ *such that* $A \subseteq D, B \subseteq E, C \subseteq F$*, and* $M = D \cup E \cup F$*.*

Proof. Let $D \preccurlyeq_F E \notin \Pi^\bullet$ be a preference satisfying the conditions above. Consider a preference context $\mathbb{P}$ with only two objects, $g_1 < g_2$, such that $\{g_1\}' = E$ and $\{g_2\}' = D$. The two objects have the same values for all attributes in F: each has all attributes in $E \cap F = D \cap F$ and none of the other attributes in F. The values of all attributes in $M \setminus F = (D \cup E) \setminus F$ are different for g_1 and g_2. Since $A \subseteq \{g_2\}', B \subseteq \{g_1\}'$, and $C \subseteq F$, we conclude that $\mathbb{P} \not\models A \preccurlyeq_C B$. Consider an arbitrary $P \preccurlyeq_R Q \in \Pi$. As $D \preccurlyeq_F E \notin \Pi^\bullet$, either $P \not\subseteq D$ or $Q \not\subseteq E$ or $R \not\subseteq F$. In all these cases, $\mathbb{P} \models P \preccurlyeq_R Q$. Thus, $\mathbb{P} \models \Pi$, but $\mathbb{P} \not\models A \preccurlyeq_C B$. It follows that $\Pi \not\models A \preccurlyeq_C B$.

For the other direction, suppose that $\Pi \not\models A \preccurlyeq_C B$. Then, there is a context $\mathbb{P}$ such that $\mathbb{P} \models \Pi$, but $\mathbb{P} \not\models A \preccurlyeq_C B$. This context must contain two objects, g_1 and g_2, for which $A \preccurlyeq_C B$ fails, i.e., $B \subseteq \{g_1\}', A \subseteq \{g_2\}', \{g_1\}' \cap C = \{g_2\}' \cap C$, but $g_2 \not\preccurlyeq g_1$. Denote $D = \{g_2\}', E = \{g_1\}'$, and $F = (M \setminus (D \cup E)) \cup (D \cap E)$. Obviously, $D \preccurlyeq_F E$ is a canonical-form preference satisfying the conditions listed in the proposition, but $\mathbb{P} \not\models D \preccurlyeq_F E$ and, therefore, $\Pi^\bullet$ cannot contain $D \preccurlyeq_F E$, which concludes the proof. □

Proposition 11 paves the way for Algorithm 1, which checks whether a preference $A \preccurlyeq_C B$ is a consequence of the set Π of ceteris paribus preferences. The algorithm starts by computing the canonical form of $A \preccurlyeq_C B$ and putting the result, $A_1 \preccurlyeq_{C_1} B_1$, on a stack: $A \preccurlyeq_C B$ follows from Π if and only if $A_1 \preccurlyeq_{C_1} B_1$ does. It then tries to find a canonical-form preference $D \preccurlyeq_F E \notin \Pi^\bullet$ such that $A_1 \subseteq D$, $B_1 \subseteq E$, $C_1 \subseteq F$, and $M = D \cup E \cup F$. We know from Proposition 11 that $\Pi \not\models A_1 \preccurlyeq_{C_1} B_1$ and, consequently, $\Pi \not\models A \preccurlyeq_C B$, if and only if such a preference can be found. The algorithm searches for it in a depth-first manner, by replacing the first preference $D \preccurlyeq_F E$ on the stack with three extensions adding an arbitrary attribute from $M \setminus (D \cup E \cup F)$ to either of D, E, and F. Note that the resulting preferences are still in canonical form. On the other hand, if we add the same attribute to exactly two of D, E, and F, the resulting preference will not be in canonical form. By adding the same attribute to all the three sets, we obtain a weaker canonical-form preference, which we we can ignore, since it is not contained in $\Pi^\bullet$ only if neither of the three other extensions is. If, at some point, the algorithm comes across a preference $D \preccurlyeq_F E \in \Pi^\bullet$, it simply removes it from the stack, because all its extensions must also be in $\Pi^\bullet$. Thus, if the stack becomes empty, we know that all canonical-form preferences of the sort required by Proposition 11 are in $\Pi^\bullet$ and conclude that $\Pi \models A \preccurlyeq_C B$. If we find a preference that cannot be extended with additional attributes and is not in $\Pi^\bullet$, we conclude that $\Pi \not\models A \preccurlyeq_C B$.

Algorithm 1 CETERIS PARIBUS CONSEQUENCE$(A \preccurlyeq_C B, \Pi)$

Input: A ceteris paribus preference $A \preccurlyeq_C B$ and a set Π of ceteris paribus preferences (over a universal set M).
Output: **true**, if $\Pi \models A \preccurlyeq_C B$; **false**, otherwise.

$\mathcal{S} := [A \cup (B \cap C) \preccurlyeq_{C \cup (A \cap B)} B \cup (A \cap C)]$ {stack}
repeat
  $D \preccurlyeq_F E := \text{pop}(\mathcal{S})$
  if $D \preccurlyeq_F E \notin \Pi^\bullet$ **then**
    $X := M \setminus (D \cup E \cup F)$
    if $X = \varnothing$ **then**
      return false
    choose $m \in X$
    push$(D \cup \{m\} \preccurlyeq_F E, \mathcal{S})$
    push$(D \preccurlyeq_F E \cup \{m\}, \mathcal{S})$
    push$(D \preccurlyeq_{F \cup \{m\}} E, \mathcal{S})$
until empty$(\mathcal{S})$
return true

Algorithm 1 is exponential in $|M|$ in the worst case, but there is little hope to do better. The reason is that, although Proposition 10 makes it possible to represent ceteris paribus preferences as implications, or Horn formulae, these Horn formulae are not sufficient to generate the theory implied by the preferences: we must add the disjunctions $\neg m_i \vee \neg m_j \vee m_k$ for different $i, j, k \in \{1, 2, 3\}$ and, crucially, $m_1 \vee m_2 \vee m_3$ for each $m \in M$. The last disjunction is not a Horn clause, which makes inference hard. However, the algorithm is linear in $|\Pi|$, which makes it efficient in applications where the language for describing preferences (and, thus, the number of attributes) is fixed and small compared to the number of preferences that need to be taken into account.

5 REDUCING BIAS

The presented approach to deriving preferences assumes that the attribute combinations in the context are the only ones that matter. In practice, the data may cover only a small fraction of possible combinations. Derived preferences hold in the data, but may not hold in the entire domain, being *biased* towards the observed part of the data.

In our lunch context, $\{\text{pork}\} \preccurlyeq_\forall \{\text{vegetables}\}$, but every option with pork there comes with chocolate mousse. It may well be that the subject does not like the combination and the true preference is weaker: $\{\text{pork, chocolate mousse}\} \preccurlyeq_\forall \{\text{vegetables}\}$.

In this section, we outline a conservative approach to preference learning, which separates knowledge about preferences from knowledge about the structure of the underlying context and makes it possible to reduce bias down to a certain well-defined point. We start by extending the definition of semantic consequence to cover both implications and preferences under the same hood. If $\mathcal{H}$ is a set of implications over M and Π is a set of preferences (of a certain kind) over subsets of M, we say that $\pi \in \Pi$ *follows from* (or *is a semantic consequence of*) $\mathcal{H} \cup \Pi$ (notation: $\mathcal{H} \cup \Pi \models \pi$) if, whenever all preferences from Π are valid in some preference context $\mathbb{P}$ over M satisfying all implications from $\mathcal{H}$ (i.e., $\mathbb{P} \models \Pi$ and $\mathbb{P} \models \mathcal{H}$), the preference π is also valid in $\mathbb{P}$ (i.e., $\mathbb{P} \models \pi$).

Definition 10. *The* Horn bias *induced by a preference context* $\mathbb{P} = (G, M, I, \leq)$ *is the set of implications that hold in* (G, M, I).

The Horn bias induced by a preference context is simply the implicational (i.e., Horn) theory behind its "non-preferential" part.

Definition 11. *Let $\mathcal{H}$ be the Horn bias induced by $\mathbb{P} = (G, M, I, \leq)$ and Π be the set of all preferences (of a certain kind) that hold in $\mathbb{P}$. We say that a preference $\pi \in \Pi$ is* Horn-biased *in $\mathbb{P}$ if there is $\Pi_1 \subseteq \Pi \setminus \{\pi\}$ such that $\Pi_1 \not\models \pi$ and $\mathcal{H} \cup \Pi_1 \models \pi$.*

Intuitively, a Horn-biased preference is one that can be deduced from other—weaker—preferences given that we know the Horn theory behind the data, but not without this additional knowledge. In the example above, $\{\text{pork}\} \preccurlyeq_\forall \{\text{vegetables}\}$ is Horn-biased, since it is a consequence of $\mathcal{H} \cup \{\{\text{pork, chocolate mousse}\} \preccurlyeq_\forall \{\text{vegetables}\}\}$, where $\mathcal{H}$ is the set of all implications valid in the context including $\{\text{pork}\} \to \{\text{chocolate mousse}\}$.

For universal and existential preferences, the Horn bias can be avoided by considering only preferences over closed attribute sets. Any preference of the form $A'' \preccurlyeq_\forall B''$ or $A'' \preccurlyeq_\exists B''$ is guaranteed not to be Horn-biased, but all other universal and existential preferences are Horn-biased. A'' and B'' are concept intents of (G, M, I); thus, unbiased preferences are preferences over formal concepts.

Technically, there are at least two ways to achieve an unbiased representation of universal preferences without constructing the basis from Proposition 4. One is to build the Duquenne–Guigues basis of (G, M, I), transform its implications into *background knowledge*, and then build the basis of $\mathbb{K}^{\mathbb{P}}_\forall$ relative to this background knowledge (see [18] for more details). The other is to build the so-called *minimal hypotheses* for $\leq$ in $\mathbb{K}^{\mathbb{P}}_\forall$ [11]. The results of the two approaches are identical: it is the minimal basis of unbiased universal preferences.

If we want to keep preferences unbiased, but be able to derive biased preferences, too, we can do this using implications and a hybrid inference system that combines the Armstrong rules [4] for implications, the rule from Proposition 2, and three additional rules:

$$\frac{X \to \bot}{\varnothing \preccurlyeq_\forall X, \quad X \preccurlyeq_\forall \varnothing}, \qquad \frac{X \to Y, \quad X \cup Y \preccurlyeq_\forall Z}{X \preccurlyeq_\forall Z},$$

$$\frac{X \to Y, \quad Z \preccurlyeq_\forall X \cup Y}{Z \preccurlyeq_\forall X}.$$

For existential preferences, an unbiased representation is actually easier to compute than a biased one: in this case, not all attribute sets are needed for the translation, but only concept intents of (G, M, I).

Definition 12. *The* conceptual existential translation *of a preference context $\mathbb{P}$ is a formal context $\mathbb{C}^{\mathbb{P}}_\exists = (G, \underline{\mathfrak{B}}(G, M, I), I_\exists)$, where $\underline{\mathfrak{B}}(G, M, I)$ is the concept set of (G, M, I) and*

$$g I_\exists (A, B) \quad \iff \quad g^{\leq} \cap A \neq \varnothing.$$

For the lunch example, this translation produces a context with fifteen attributes corresponding to the concepts shown in Fig. 2 compared to 64 attributes produced by the existential translation.

Definition 13. *The* conceptual translation of an existential preference $A \preccurlyeq_\exists B$, *denoted by $T^{\mathbb{C}}_\exists(A \preccurlyeq_\exists B)$, is the implication*

$$\{(A', A'')\} \to \{(B', B'')\}$$

of the formal context $\mathbb{C}^{\mathbb{P}}_\exists$.

The conceptual translation preserves the validity of existential preferences and provides another way to summarize them:

$$\{A \preccurlyeq_\exists B \mid \mathbb{C}^{\mathbb{P}}_\exists \models \{(A', A)\} \to \{(B', B)\} \text{ and } B \not\subseteq A\}$$

is a complete set of existential preferences *relative* to the implications of (G, M, I). A hybrid inference system for implications and existential preferences includes Armstrong rules [4], the rules for existential preferences from Proposition 7, and the rule

$$\frac{A \to B}{A \preccurlyeq_\exists B}.$$

For ceteris paribus preferences, bias can be reduced even further.

Definition 14. *We call the expression $[A, B]C \Rightarrow D[E, F]$ a* doubly conditional functional dependency *and say that it* holds *in (G, M, I) if, for every $g, h \in G$ such that $g \in A', h \in B'$, and $\{g\}' \cap C = \{h\}' \cap C$, we have $g \in E', h \in F'$, and $\{g\}' \cap D = \{h\}' \cap D$.*

This generalizes both implications and conditional functional dependencies from [9]. Thus, the induced bias, which we call the 2CFD bias, includes the Horn bias.

Definition 15. *The* 2CFD bias *induced by a preference context $\mathbb{P} = (G, M, I, \leq)$ is the set of doubly conditional functional dependencies that hold in the formal context (G, M, I).*

Doubly conditional functional dependencies are in one-to-one correspondence with implications of the context obtained from $\mathbb{K}^{\mathbb{P}}_\sim$ by removing the $\leq$ attribute. To avoid the 2CFD bias, we should consider only preferences translated into implications of $\mathbb{K}^{\mathbb{P}}_\sim$ whose left-hand side X is minimal w.r.t. $X \cup \{\leq\}$ being a concept intent of $\mathbb{K}^{\mathbb{P}}_\sim$. These correspond to minimal hypotheses for $\leq$ [11].

6 CONCLUSION

We have proposed a formalism based on concept lattices for modeling several types of preferences, including preferences that hold only *ceteris paribus* and showed how such preference models can be learned from data. Our approach may seem limited for, taken literally, it is only concerned with preferences over conjunctions of boolean variables; even negations of variables are not covered. Compare this to other approaches, such as *cp-theories* as defined in [23]. In this framework, one works with a set of variables V, each of which has an associated set of values. A *conditional preference* is a statement of the form $u : x_1 > x_2[W]$, where u is an assignment to $U \subseteq V$, x_1 and x_2 are different assignments to some $X \in V$, and W is a subset of $V \setminus (U \cup \{X\})$. Such preference is interpreted as follows: between two alternatives satisfying u, the one with $X = x_1$ is preferred to the one with $X = x_2$ provided that they agree on all other variables with a possible exception of those in W. This may be regarded as a generalization of CP-nets [6] and TCP-nets [7].

To model such preferences in our framework, we can build a preference context whose attribute set M consists of expressions of the form $X = x$, where $X \in U$ and x ranges over possible values of X. For the boolean case, this would mean adding a negated copy for each attribute. Then, a strict conditional preference $u : x_1 > x_2[W]$ would have the following weak counterpart in our framework:

$$u \cup \{X = x_2\} \preccurlyeq_{\{M \setminus W\}} u \cup \{X = x_1\}.$$

To express strict conditional preferences, we can start with a strict preference relation over objects. On the other hand, the language of conditional preferences only allows preferences of a single variable, whereas, with our approach, we can express (and learn from data) more general preferences such as

$$u \cup \{X = x_2, Y = y_2\} \preccurlyeq_{\{M \setminus W\}} u \cup \{X = x_1, Y = y_1\}.$$

Furthermore, for variables with ordinal values, we could use attributes of the form, e.g., $x_1 \leq X \leq x_2$ instead of just $X = x$. In FCA, this is done by *scaling* so-called *many-valued contexts*, in which attributes are not necessarily boolean [12]. Also, the ceteris paribus conditions in the translated context from Definition 7, which are specified through attributes from $M \times \{3\}$, could be customized to specify relations other than equality. This would make it possible to express preferences like the following: "Between two ways of travel, I prefer a cheap one provided that it is at least as fast as the other." We leave a thorough treatment of these issues and a proper comparison to other approaches to preference modeling for further research.

We also plan to develop algorithms for learning preferences from queries [2]. Such algorithms exist, e.g., for CP-nets [15]. Since, in our framework, preferences can be translated into Horn clauses, it might be possible to adapt the output-polynomial algorithm for learning Horn theories from [3] (adaptation is needed, because the translation is not surjective, i.e., not all Horn clauses over a given set of variables correspond to preferences). However, this algorithm uses equivalence queries, which are hard to answer. An alternative approach is a similar technique from FCA, called *attribute exploration* [10, 19, 20], which only uses queries on the validity of implications (even though, in theory, the number of such questions may be exponentially large). Note that, with this approach, the user is not asked to specify preferences between two given examples, but rather to confirm or reject a stated preference. When rejecting a preference, the user must point out two objects contradicting this preference. A precise specification of such query learning algorithm and its computational complexity are a matter of further research.

In application to real-life data analysis, it may be useful to introduce some statistical considerations into the theory presented here. One obvious approach is to replace the semantics based on implications by one based on association rules [1], thus, allowing exceptions in derived preferences, but making sure that these preferences are supported by a sufficiently large volume of data. On the other hand, methods for pruning concept lattices by selecting only the most interesting (in some sense) concepts [14, 5] may be of value in deriving "unbiased" preferences from Sect. 5, which are interpreted as preferences over concepts.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami, 'Mining association rules between sets of items in large databases', *SIGMOD Rec.*, **22**(2), 207–216, (June 1993).

[2] Dana Angluin, 'Queries and concept learning', *Machine Learning*, **2**, 319–342, (1988).

[3] Dana Angluin, Michael Frazier, and Leonard Pitt, 'Learning conjunctions of Horn clauses', *Machine Learning*, **9**, 147–164, (1992).

[4] William Ward Armstrong, 'Dependency structures of data base relationships', *Proc. IFIP Congress*, 580–583, (1974).

[5] Mikhail Babin and Sergei O. Kuznetsov, 'Approximating concept stability', in *Formal Concept Analysis*, eds., Florent Domenach, Dmitry Ignatov, and Jonas Poelmans, volume 7278 of *Lecture Notes in Computer Science*, 7–15, Springer Berlin / Heidelberg, (2012).

[6] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole, 'Reasoning with conditional ceteris paribus preference statements', in *UAI*, eds., Kathryn B. Laskey and Henri Prade, pp. 71–80. Morgan Kaufmann, (1999).

[7] Ronen I. Brafman and Carmel Domshlak, 'Introducing variable importance tradeoffs into cp-nets', in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pp. 69–76. Morgan Kaufmann Publishers Inc., (2002).

[8] Claudio Carpineto and Giovanni Romano, *Concept Data Analysis: Theory and Applications*, Wiley, 2004.

[9] Wenfei Fan, Floris Geerts, Laks V.S. Lakshmanan, and Ming Xiong, 'Discovering conditional functional dependencies', in *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pp. 1231–1234, (2009).

[10] Bernhard Ganter, 'Attribute exploration with background knowledge', *Theoretical Computer Science*, **217**(2), 215–233, (1999).

[11] Bernhard Ganter and Sergei O. Kuznetsov, 'Formalizing hypotheses with concepts', in *ICCS*, eds., Bernhard Ganter and Guy W. Mineau, volume 1867 of *Lecture Notes in Computer Science*, pp. 342–356. Springer, (2000).

[12] Bernhard Ganter and Rudolf Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer, Berlin/Heidelberg, 1999.

[13] Jean-Luc Guigues and Vincent Duquenne, 'Famille minimale d'implications informatives résultant d'un tableau de données binaires', *Mathématiques et Sciences Humaines*, **24**(95), 5–18, (1986).

[14] Mikhail Klimushkin, Sergei Obiedkov, and Camille Roth, 'Approaches to the selection of relevant concepts in the case of noisy data', in *Formal Concept Analysis*, eds., Léonard Kwuida and Baris Sertkaya, volume 5986 of *Lecture Notes in Computer Science*, 255–266, Springer Berlin / Heidelberg, (2010).

[15] Frédéric Koriche and Bruno Zanuttini, 'Learning conditional preference networks', *Artificial Intelligence*, **174**(11), 685 – 703, (2010).

[16] Sergei O. Kuznetsov and Sergei Obiedkov, 'Some decision and counting problems of the Duquenne–Guigues basis of implications', *Discrete Applied Mathematics*, **156**(11), 1994–2003, (2008).

[17] Fenrong Liu, *Changing for the Better. Preference Dynamics and Agent Diversity*, Ph.D. dissertation, Universiteit van Amsterdam, 2008.

[18] Sergei Obiedkov, 'Modeling preferences over attribute sets in formal concept analysis', in *Formal Concept Analysis*, volume 7278 of *Lecture Notes in Artificial Intelligence*, 227–243, Springer Berlin / Heidelberg, (2012).

[19] Sergei Obiedkov, Derrick G. Kourie, and J.H.P. Eloff, 'Building access control models with attribute exploration', *Computers & Security*, **28**(1-2), 2–7, (2009).

[20] Artem Revenko and Sergei O. Kuznetsov, 'Attribute exploration of properties of functions on sets', *Fundamenta Informaticae*, **115**(4), 377–394, (2012).

[21] Johan van Benthem, Patrick Girard, and Olivier Roy, 'Everything else being equal: A modal logic for *ceteris paribus* preferences', *J. Philosophical Logic*, **38**(1), 83–125, (2009).

[22] Georg H. von Wright, *The Logic of Preference*, Edinburgh University Press, 1963.

[23] Nic Wilson, 'Computational techniques for a simple theory of conditional preferences', *Artif. Intell.*, **175**(7-8), 1053–1091, (2011).

A Statistical Approach to Calibrating the Scores of Biased Reviewers: The Linear vs. the Nonlinear Model[1]

Magnus Roos[2] and **Jörg Rothe**[2] and **Joachim Rudolph**[3] and **Björn Scheuermann**[4] and **Dietrich Stoyan**[5]

Abstract. Two methods are proposed for aggregating the scores of reviewers in a peer-reviewing rating system. Both methods are of a statistical nature. The simpler method, which is based on a classical statistical approach from the field of linear models, uses the analysis of variance and can thus be realized by means of existing statistical software. The more advanced method, which is a slight modification of the method proposed by Roos et al. [13], uses a nonlinear model and numerical optimization based on a least-squares approach. Under reasonable statistical assumptions, both approaches—the linear and the nonlinear one—can be seen as using the maximum likelihood principle. Application of either method implies also an evaluation of the reviewers. An application example with real conference data shows the power of the statistical methods, compared with the common naive approach of simply taking the average scores.

1 Introduction

Evaluation of persons, papers, products, etc. is a fundamental social activity. For example, students are evaluated by teachers, scientific papers by journal/conference reviewers, and sportsmen by referees, e. g., in figure skating and gymnastics. Even if all reviewers in a rating system are subjectively fair, some of them may be biased and produce scores systematically too high or too low. If then not all objects are reviewed by all reviewers, it becomes complicated to aggregate the scores given to the same objects in a fair way.

The present paper focuses on the problem of ranking scientific papers submitted to conferences, where usually the relative number of reviews per paper is small. The common procedure applied by popular conference management systems such as EasyChair[6] and ConfMaster[7] is described as (quoting from the EasyChair website): "When computing the average score, weight reviews by reviewer's confidence." This means that all scores given to a paper are simply averaged, possibly weighted by reviewer-specific weights, the confidence levels of the reviewers, which again are very subjective because every reviewer evaluates only him- or herself. Under these conditions it may happen that by good luck a weak scientific paper goes to some lenient or generous reviewers, whereas a good paper goes to a harsh reviewer and some normal reviewers. Then the weak paper might be accepted, but the good one is rejected.

The present paper aims to improve the common "naive" (as Lauw et al. [9] call it) approach where the overall scores of all objects are obtained by simply averaging all given scores of the object. Of course, paper scores can only provide some guidance on paper acceptance; the final decision is usually made on deeper considerations.

It is assumed here that external information about the reviewers is not used, such as weighting the scores. There is also no separate "training" phase in order to characterize the reviewers' tendencies. Instead, the proposed methods apply cross-classification techniques to determine the characteristics of both the reviewers and the judged objects simultaneously in one step. All reviewers are assumed to be "honest," to exercise their best judgments, without any personal relation to certain objects. Nevertheless, some reviewers may be biased in giving systematically high or low scores. As long as all papers are evaluated by all reviewers, this is not an obstacle to fair score aggregation by averaging. However, if there are only a few reviews per paper, problems are likely to arise. The following toy example taken from [8] shows what can happen.

Example 1 *Consider the data in Table 1. There are five reviewers (r_i) and five papers (p_j). The original scores y_{ij} from [8] are here multiplied by* 10 *and are thus in the range from 0 to 10. Consisting of only* 15 *scores in total, this data set is very small.*

Table 1. Data for a toy example taken from [8].

	p_1	p_2	p_3	p_4	p_5
r_1	6	6	6	–	–
r_2	3	–	–	4	–
r_3	3	–	–	–	4
r_4	–	3	3	4	4
r_5	–	3	3	4	4

The naive approach results in the same average score of 4.0 *for all five papers. This seems to be highly questionable: in their preliminary discussion, Lauw et al. [8] point out that reviewer r_1 is very likely to be lenient, causing too high aggregated scores for papers p_1, p_2, and p_3. In Section 2.2, this example is to be continued to show the results that can be obtained by means of statistical methods.*

Related Work

Preference aggregation is a wide field that has been intensely studied by various scientific communities, ranging from multiagent systems

[1] This work was supported in part by DFG grants RO 1202/12-1 and RO 1202/15-1, the European Science Foundation's EUROCORES program LogICCC, an SFF grant of HHU Düsseldorf, ARC grant DP110101792, and a DAAD-PPP grant in the PROCOPE project.

[2] Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, 40225 Düsseldorf, Germany, email: {roos, rothe}@cs.uni-duesseldorf.de

[3] Institut für Sozialwissenschaften, Humboldt-Universität zu Berlin, 10099 Berlin, Germany

[4] Institut für Informatik 4, Universität Bonn, 53113 Bonn, Germany, email: scheuermann@cs.uni-bonn.de

[5] Institut für Stochastik, TU Bergakademie Freiberg, 09596 Freiberg, Germany, email: stoyan@math.tu-freiberg.de

[6] `http://www.easycair.org`

[7] `http://www.confmaster.net`

to computational social choice. The topic of this paper—aggregating the scores in reviewing scientific papers—has also been investigated, although from other angles and using different methods. For example, Douceur [4] encoded the aggregation problem into a corresponding problem on directed multigraphs and focuses on rankings (i. e., ordinal preferences) rather than ratings (i. e., cardinal preferences obtained by assigning review scores). By contrast, Haenni [6] presents an algebraic framework to study the problem of aggregating individual scores.

The present paper uses methods of analysis of variance from the field of statistics, see [7]. The setting is called *two-way classification* there, where one "way" relates to reviewers and the other to papers. This classical statistical approach from the field of linear models is adapted here. This leads to fairer overall scores for the papers, where "fairer" in a technical sense refers to the fact that the proposed method leads to unbiased estimators for certain model parameters (see Section 2.2 for details). At the same time in parallel, the method also allows for an evaluation of the reviewers.

The papers by Lauw et al. [8, 9] tackle the same problem as the present paper, yet with quite a different approach. They apply a so-called "differential model," which is an ad-hoc nonlinear model. Their model includes an unknown model parameter α, which appears not to be statistically estimable. No random errors occur in this model, although in real review processes such effects are well conceivable to play a role.

We will first present the simple linear approach in Section 2.2. It can be realized by existing statistical software. This approach is then refined in Section 2.3 by a nonlinear method, which applies techniques from quadratic programming. Under some statistical assumptions, both approaches—the linear and the nonlinear one—can be seen as using the maximum likelihood principle.

The nonlinear model is inspired by a solution to the offline synchronization problem in broadcast networks, as discussed by Scheuermann et al. [14]. In that work, the problem of synchronizing timestamps in a set of event log files is addressed, where each log file has been generated with a different, potentially deviating, local clock. "Reviewers" in the present paper take the role of "network nodes" there, the role of "papers" here corresponds to "network packet transmissions" there, and "review scores" here are in line with "reception timestamps" there. However, the setting and assumptions in Scheuermann et al. differ in some central aspects. In particular, random packet reception time delays, which correspond to random components in review scores, follow exponential distribtions and are not Gaussian. More technically, the resulting optimization problem is linear in [14], while it is (semi-definite) quadratic here.

There is a significant body of existing work in the area of preference aggregation, i. e., on the question how to aggregate individual preferences into a common, global ranking. Some of these works use related estimators in different settings. For example, Conitzer and Sandholm [3], Conitzer, Rognlie, and Xia [1], and Xia et al. [18, 17] apply maximum likelihood estimation to model the "noise" in voting. Relatedly, Pini et al. [12] study the issue of aggregating partially ordered preferences with respect to Arrovian impossibility theorems. However, their framework differs from the model used here: they consider ordinal preferences, whereas peer-reviewing is commonly based on scores, i. e., on cardinal preferences. Note that cardinal preferences are more expressive than ordinal preferences, as they also provide a notion of distance.

2 Models

2.1 Basic Assumptions

In the reviewing process considered, reviewers not only comment on the weaknesses and strengths of the papers, but give a score to each paper reviewed. The following analysis focuses on only the scores. These scores are assumed to be integers, to which situation most evaluation processes can be transformed, even if decimal numbers with one or two decimals are given. High scores mean good quality.

There are I reviewers r_i and J papers p_j. For each pair (i, j), there exists a binary number e_{ij}, where $e_{ij} = 1$ means that reviewer r_i reviews paper p_j, and $e_{ij} = 0$ otherwise. The matrix $(e_{ij})_{1\le i\le I, 1\le j\le J}$ is called *incidence matrix*. Let $E = \{(i, j) \mid e_{ij} = 1\}$. The scores corresponding to pairs $(i, j) \in E$ are denoted by y_{ij}.

2.2 The Linear Model

Adapting the classical statistical linear modeling approach, the following model is used:

$$y_{ij} = \mathscr{D}\left(\mu + \alpha_i + \beta_j + \varepsilon_{ij}\right) \quad \text{for } (i, j) \in E. \tag{1}$$

Here, $\mathscr{D}$ is a discretization operator that transforms any real number x into the integer score $\mathscr{D}(x)$. The other symbols have the following meanings:

- μ is the overall mean of all scores given,
- α_i is the mean difference between the scores of reviewer r_i and μ,
- β_j is the mean difference between the scores of paper p_j and μ,
- ε_{ij} is a random error for $(i, j) \in E$.

The α_i are closely related to the "leniencies" of reviewers discussed by Lauw et al. [8, 9], and the β_j to their paper "qualities." The idea is that reviewer r_i does not assign a score to paper p_j based on its true quality β_j (which r_i does not know), but based on r_i's own noisy view of p_j's quality, which is $\beta_j + \varepsilon_{ij}$. This judgment is then linearly shifted according to the reviewer's "leniency". Simplifying more general models, it is assumed that there is no interaction between reviewers and papers (which, if desired, could be expressed by parameters $(\alpha\beta)_{ij}$).

The strategy in the following is to ignore the discretization in the statistics and to assume that the discretized data belong to the truly linear model

$$y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij} \quad \text{for } (i, j) \in E. \tag{2}$$

with

$$\mathbf{E}\,\varepsilon_{ij} \equiv 0 \quad \text{and} \quad \mathbf{var}\,\varepsilon_{ij} \equiv \sigma^2 \quad \text{for } (i, j) \in E, \tag{3}$$

where the ε_{ij} are independent and **E** and **var** denote the expectation and variance, respectively. The error of this simplified approach will be discussed in the full version of this paper.

Model (2) is called *two-way classification* in the analysis of variance, see, e. g., the book by Draper and Smith [5].

As mentioned above, the naive estimators of the sums $\mu + \beta_j$, here denoted by $\widehat{\mu + \beta_j}$, are the averages of all review scores assigned to the respective paper:

$$\widehat{\mu + \beta_j} = \overline{y}_{*j} = \frac{1}{n_{*j}} \sum_{i:(i,j)\in E} y_{ij}, \tag{4}$$

where n_{*j} is the number of reviews for paper p_j. No serious statistician will use them, since these estimators are not unbiased and better estimators are possible.

Theory says that only the differences of the effects α_i and β_j can be estimated without bias. Fortunately, for the problem of ranking papers it completely suffices to have estimates of the differences $\beta_j - \beta_1$. And for evaluating the reviewers, estimates of the differences $\alpha_i - \alpha_1$ are fully sufficient. Thus, one may assume that

$$\sum_{i=1}^{I} \alpha_i = 0 \qquad \text{and} \qquad \sum_{j=1}^{J} \beta_j = 0. \tag{5}$$

In many statistical textbooks such as [5] and [15], it is assumed that for each pair (i, j) a fixed, strictly positive number n of observations is given (where, in typical settings, $n \gg 1$). If so, least-squares estimates of μ, α_i, and β_j are easy to determine. They directly follow from the means

$$\overline{y}_{**} = \frac{1}{IJ}\sum_{i=1}^{I}\sum_{j=1}^{J} y_{ij}, \qquad \overline{y}_{i*} = \frac{1}{J}\sum_{j=1}^{J} y_{ij}, \quad \text{and} \quad \overline{y}_{*j} = \frac{1}{I}\sum_{i=1}^{I} y_{ij}$$

as $\mu = \overline{y}_{**}$, $\alpha_i = \overline{y}_{i*} - \overline{y}_{**}$, and $\beta_j = \overline{y}_{*j} - \overline{y}_{**}$. These estimators are unbiased. In this case, the naive approach is the best. However, in the situation typical for peer reviewing, the "observation" counts n_{ij} are 0 (reviewer i does not review paper j) or 1 (reviewer i reviews paper j). (Note that $n_{ij} = 2$ would mean that reviewer i reviews paper j twice, independently.) We are confronted with a so-called "incomplete" (and "unbalanced") experimental design. The corresponding theory is described by Koch [7, Sections 3.4.2 and 3.4.3]. The case of interest here is there referred to as *two-way cross-classification*.

The parameters are estimated by the least-squares approach, i. e., the sum over all

$$(y_{ij} - \mu - \alpha_i - \beta_j)^2$$

is minimized. To this end, Koch [7] describes numerical approaches based on normal equations. Standard statistical software offers various ways to obtain estimators of the α_i, the β_j, and of μ, which differ in the so-called reparametrization conditions.

The model variance σ^2 is estimated by the mean squared error, which is the sum of quadratic deviations $(y_{ij} - \hat{y}_{ij})^2$ with $\hat{y}_{ij} = \hat{\mu} + \hat{\alpha}_i + \hat{\beta}_j$ divided by their number minus one. The estimators obtained are unbiased and in some sense "best." In the case of normally distributed ε_{ij}, the least-squares estimators are also maximum likelihood estimators.

For the practical statistical analysis, the statistical software package IBM-SPSS Statistics 20 (which we abbreviate by SPSS), procedure UNIANOVA, was used. The procedure UNIANOVA does not use the conditions (5), but it is preset such that α_I and β_J are set to zero in the model discussed here.

Alternatively, also the program that will be mentioned in the next section (see Algorithm 1) can be used by setting $\gamma_i = 1$ in (6) below, which leads to (1). Both programs yield identical results.

The parameters determined by SPSS can easily be transformed into the parameters μ, α_i, and β_j. Simulations and direct calculation of model parameters are easily possible based on the matrix module of SPSS.

Example 2 (continuing Example 1) *Table 2 shows the values for the parameters in the linear model; the parameter μ is estimated as* 4.0*. The model parameters indicate that reviewer r_1 indeed has to be considered as lenient, while the other reviewers are estimated to have the same degree of rigor. The papers are now divided into two classes: p_1, p_2, and p_3 seem to be weaker papers with lower scores, while the other two papers appear to be of the same higher quality. It cannot surprise that Lauw et al. [8] arrive at the same conclusions for this extremely simple example.*

Table 2. Parameters for the toy example from [8].

$i,\ j$	α_i	β_j
1	2.4	−0.4
2	−0.6	−0.4
3	−0.6	−0.4
4	−0.6	0.6
5	−0.6	0.6

Note that in the example above, the estimated parameter values exactly reproduce the scores from Table 1 when used in (2) with all $\varepsilon_{ij} = 0$. Essentially, this means that no random deviations at all are necessary to "explain" the reviewers' scores. Therefore, this example has to be considered extremely simple.

2.3 The Nonlinear Model

The linear model from the previous section is now refined to a nonlinear model, which modifies the method proposed by Roos et al. [13] so as to generalize (1) to

$$y_{ij} = \mathscr{D}\left(\mu + \gamma_i(\alpha_i + \beta_j + \varepsilon_{ij})\right) \quad \text{for } (i,j) \in E \tag{6}$$

with positive parameters γ_i. For the special case of $\gamma_i \equiv 1$, (6) coincides with (1). The term $\gamma_i(\alpha_i + \beta_j + \varepsilon_{ij})$ models the interaction between reviewer r_i and paper p_j; γ_i is a proportionality factor; and μ, α_i, β_j, and ε_{ij} have the same meaning as in the linear case.

Reviewer r_i's perceived, noisy quality level $\beta_j + \varepsilon_{ij}$ is, just like in the linear model, added to this reviewer's systematic bias α_i. In addition, though, the result is transformed by multiplication with the reviewer-specific scaling factor γ_i. This factor models r_i's individual rigor: in essence, γ_i describes by how much reviewer i's review score changes, given a fixed change in (perceived) paper quality.

Even though this nonlinear model is relatively simple, it allows to capture a wide range of reviewer characteristics.

An assumption similar to $\sum_{i=1}^{I} \alpha_i = 0$ in the linear case (see Section 2.2) is now done by

$$\alpha_I \quad = \quad 0. \tag{7}$$

This leads to a problem slightly smaller than that with $\sum_{i=1}^{I} \alpha_i = 0$. Both restrictions are possible and plausible, and the results can simply be transformed to each other by choosing a suitable parameter μ. The aim is to estimate the parameters α_i, β_j, γ_i, and μ. Again the least-squares approach is used, which minimizes the sum of squared errors ε_{ij},

$$\sum_{(i,j)\in E} \left(\frac{y_{ij}}{\gamma_i} - \frac{\mu}{\gamma_i} - \alpha_i - \beta_j \right)^2. \tag{8}$$

Since this does not affect the optimization itself, in this setting μ can be set to zero. After getting the result, one may shift the values so that a condition like $\sum_{j=1}^{J} \beta_j = 0$ as in (5) is fulfilled. It is easy to see that the resulting parameter estimators are maximum likelihood estimators if the errors ε_{ij} are i. i. d. Gaussian as in (3).

Numerically, the minimization procedure is carried out by means of a direct optimization program such as a so-called quadratic program, see, e. g., the book by Nocedal and Wright [11]. In general, a *quadratic program* (QP) is an optimization problem of the form:

$$\text{minimize} \qquad \frac{1}{2}x^T Q x + c^T x \tag{9}$$

$$\text{subject to} \qquad Ax \geq b, \tag{10}$$

where (letting $\mathbb{Q}$ denote the set of rational numbers) $x \in \mathbb{Q}^n$, $Q \in \mathbb{Q}^{n\times n}$, $c \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m\times n}$, and $b \in \mathbb{Q}^m$. The solution of a QP is a vector x that minimizes the expression in (9), simultaneously fulfilling all constraints in (10).

With the simple substitution $\tilde{\gamma}_i = 1/\gamma_i$ in (8) one obtains

$$\sum_{(i,j)\in E} \left(y_{ij}\tilde{\gamma}_i - \mu\tilde{\gamma}_i - \alpha_i - \beta_j\right)^2, \tag{11}$$

which can be transformed into the form of a QP as required by (9) and (10). In the following, the estimators of α_i, β_j, and $\tilde{\gamma}_i$ are denoted by $\hat{\alpha}_i$, $\hat{\beta}_j$, and $\hat{\gamma}_i$. With respect to the QP discussed so far, note that a trivial solution can be achieved by setting $\hat{\gamma}_i$, $\hat{\alpha}_i$, and $\hat{\beta}_j$ each to zero, which clearly is not reasonable. Assuming typical reviewers to be "rational", one may require the normalization constraint:

$$\frac{1}{I}\sum_{i=1}^{I}\hat{\gamma}_i \quad = \quad 1. \tag{12}$$

Defining a vector $x = \left(\hat{\beta}_1,\ldots,\hat{\beta}_J,\hat{\gamma}_1,\ldots,\hat{\gamma}_I,\hat{\alpha}_1,\ldots,\hat{\alpha}_I\right)^T$, containing the variables to estimate, one obtains the QP:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Qx \\ \text{subject to} & Ax \geq b \end{array} \tag{13}$$

with a square matrix Q (see lines 2–13 of Algorithm 1 below), and a matrix A representing the normalization constraint (12).

A QP with a positive definite matrix Q has a unique solution and can be solved in polynomial time using interior-point methods, see, e. g., [16]. In this specific QP, the matrix Q is at least positive semi-definite, i. e., all eigenvalues of A are nonnegative, because it can be written as $H \cdot H^T$ (see Algorithm 1 below for the definition of matrix H). Analogously to the linear model in Section 2.2, one does not have any global, absolute "reference" to which the overall scores could be adjusted. This leads to an additional degree of freedom in the optimization, which precludes obtaining a unique maximum. In fact, a similar issue also occurred in the work of Scheuermann et al. [14], and along similar lines as there it is easy to overcome: one may set $\alpha_I = 0$, thus using one reviewer as a "fixed" reference point. In this paper, the last reviewer is picked for this constraint, see Equation (7). Yet, also with this modification it is *still* possible to come up with pathological instances where the solution is not unique. This lies in the nature of the problem: For instance, it is impossible to compare the relative "rigor" of two groups of reviewers, if there is no paper that has been reviewed by at least one reviewer out of each of the two groups. In general, such ambiguities are easily identified and can always be resolved by introducing additional constraints as needed (or, alternatively, by assigning additional reviews). This then yields a positive definite matrix Q and consequently a unique solution of the QP.

To solve the resulting QP, one can use existing solvers such as MINQ [10], a MATLAB script for bound constrained indefinite quadratic programming. Algorithm 1 illustrates this approach. The scores y_{ij} for $(i,j) \in E$ are assumed to be nonnegative for line 5 to work. Any negative number (e. g., -1) at position (i,j) in the input matrix M indicates that reviewer r_i did not review submission p_j (i. e., $(i,j) \notin E$). M thus encodes both E and the review scores y_{ij}. Note that the resulting estimated scores in $\hat{\beta}$ may exceed the interval of the input scores. This can, however, be overcome by subsequently scaling to results as desired, as discussed above; this yields the scaled score estimates, in the following denoted by β_j^*, for all submissions.

Algorithm 1 Computing the estimated scores

1: **Input:** $M \in \mathbb{Q}^{m\times n}$ *// M contains the given scores.*
2: $H = \begin{bmatrix}0\end{bmatrix} \in \mathbb{Q}^{(2m+n)\times(m\cdot n)}$
3: **for** $j \in \{1,2,\ldots,m\}$ **do**
4: **for** $k \in \{1,2,\ldots,n\}$ **do**
5: **if** $M_{(j,k)} \geq 0$ **then**
6: $H_{(k,(k-1)\cdot m+j)} = 1$
7: $H_{(n+j,(k-1)\cdot m+j)} = -M_{(j,k)}$
8: $H_{(n+m+j,(k-1)\cdot m+j)} = 1$
9: **end if**
10: **end for**
11: **end for**
12: remove the last row from H *// normalization*
13: $Q = 2\cdot H\cdot H^T$
14: $h_1 = \begin{pmatrix}0 & \cdots & 0\end{pmatrix} \in \mathbb{Q}^n$
15: $h_2 = \begin{pmatrix}1 & \cdots & 1\end{pmatrix} \in \mathbb{Q}^m$
16: $h_3 = \begin{pmatrix}0 & \cdots & 0\end{pmatrix} \in \mathbb{Q}^{m-1}$
17: $A = \begin{bmatrix} h_1 & \frac{1}{m}\cdot h_2 & h_3 \\ h_1 & -\frac{1}{m}\cdot h_2 & h_3 \end{bmatrix}$
18: $b = \begin{pmatrix}1\\-1\end{pmatrix}$
19: solve: $\min \frac{1}{2}x^T Qx$ subject to $Ax \geq b$
20: $\hat{\beta} = \begin{pmatrix}x_1 & \cdots & x_n\end{pmatrix}^T$
21: **Output:** $\hat{\beta} \in \mathbb{Q}^n$

3 A Case Study

The following discusses data from the *Third International Workshop on Computational Social Choice* (COMSOC-2010) that took place in September 2010 in Düsseldorf, Germany [2]. There were 57 submissions (where submissions that had to be rejected on formal grounds are disregarded) and 20 reviewers. Every submission was reviewed by at least two reviewers; a third reviewer was assigned to some submissions later on, and one paper was even reviewed by four reviewers. (The fact that these extra reviews were somehow related to the evaluation of the papers in the first two reports is ignored in the following.) Table 3 shows the data, the results of the reviewing process. It contains the scores given by the reviewers to the papers, where "–" means "no review." As is common in EasyChair, the scores were integers between -3 and 3, which are here shifted to the integers between 1 and 7, where 7 is the best possible score.

Table 4 shows the main results of applying the methods proposed in this paper to real conference data: the estimated COMSOC-2010 paper scores obtained by the two approaches presented here, which are closely related to the β_j. The acceptance threshold of the conference was around 4.5, based on the naive approach. This led to acceptance of a total of 40 submissions, while 17 were rejected.

Table 5 shows the parameters α_i and γ_i of the reviewers, which allow to evaluate them as well. This is simpler in the linear than in the nonlinear approach. According to the linear approach, reviewer 7 with $\alpha_7 = 2.3662$ is the most lenient reviewer. In the nonlinear approach, the relatively large value of $\gamma_7 = 6.1283$ also leads to high review scores even if the paper quality is only moderate. By contrast, reviewer r_{19} with $\alpha_{19} = -0.8523$ (in the linear model) has some tendency of being harsh. The parameters in the nonlinear approach, $\alpha_{19} = -0.6411$ and $\gamma_{19} = 1.8889$, allow for a more differentiated representation of this reviewer's mapping of paper quality to review score.

The differences in modeling and reducing reviewer bias between the approaches results in different paper rankings. Consider, for ex-

Table 3. Input data from the review process for COMSOC-2010. The scores of 20 reviewers for 57 papers are shown. (Note that the data matrix given here is transposed compared with Table 1.) The papers are ordered with respect to their rank obtained by the naive approach.

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}	r_{17}	r_{18}	r_{19}	r_{20}
p_1	–	–	–	–	–	–	–	–	–	–	7	–	–	–	7	–	–	–	–	7
p_2	7	–	–	–	–	–	–	–	–	7	–	–	–	–	–	–	–	–	–	–
p_3	–	–	–	–	–	–	–	–	–	–	7	–	–	–	7	–	–	–	–	7
p_4	–	–	–	–	–	–	–	–	–	–	–	7	–	–	7	–	–	–	–	–
p_5	–	7	–	–	–	–	–	–	–	–	–	–	–	6	–	–	–	–	–	–
p_6	–	–	–	–	–	–	–	–	–	–	–	7	–	6	–	–	–	–	–	–
p_7	–	–	–	–	–	–	–	–	–	7	–	–	–	–	6	–	–	–	–	–
p_8	–	–	–	–	–	–	–	–	–	–	–	–	7	–	–	–	–	–	6	–
p_9	–	–	–	–	–	–	–	–	–	–	–	–	–	–	7	–	–	–	–	6
p_{10}	–	–	–	–	6	–	–	–	–	–	–	–	–	–	–	7	–	–	–	–
p_{11}	6	–	–	–	–	–	–	–	–	–	–	7	–	–	–	–	–	–	–	–
p_{12}	7	–	–	–	–	–	–	–	–	6	–	–	–	6	–	–	–	–	–	–
p_{13}	–	–	–	–	–	–	–	–	–	–	–	–	–	–	7	–	–	–	5	–
p_{14}	6	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	6	–	–
p_{15}	–	6	–	–	–	6	–	–	–	–	–	–	–	–	–	–	–	–	–	–
p_{16}	–	6	–	–	–	–	–	–	–	–	6	–	–	–	–	–	6	–	–	–
p_{17}	–	–	–	–	–	–	6	–	–	6	–	–	–	–	–	–	–	–	–	–
p_{18}	–	–	–	–	–	–	–	–	6	–	–	–	–	6	–	–	–	–	–	–
p_{19}	6	–	–	–	–	–	–	–	–	–	–	–	–	–	6	–	–	–	–	–
p_{20}	–	–	–	6	–	–	–	–	–	–	–	–	–	–	–	6	–	–	–	–
p_{21}	–	–	–	–	6	–	–	–	–	–	–	–	6	–	–	–	–	–	–	–
p_{22}	–	–	6	–	–	–	–	–	–	–	–	–	–	–	–	6	–	–	–	–
p_{23}	–	–	–	–	5	–	–	–	–	–	–	–	–	–	–	–	–	–	7	–
p_{24}	7	–	–	–	5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
p_{25}	–	–	–	–	5	–	–	–	6	–	–	–	–	–	–	–	–	–	–	–
p_{26}	–	–	–	6	–	–	5	–	–	–	–	–	–	–	–	–	–	–	–	–
p_{27}	–	6	5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
p_{28}	–	–	–	–	–	–	–	5	–	6	–	–	–	–	–	–	–	–	–	–
p_{29}	–	–	–	–	–	–	–	–	–	–	–	5	–	–	–	6	–	–	–	–
p_{30}	–	–	6	–	–	–	–	–	5	–	–	–	–	–	–	–	–	–	–	–
p_{31}	–	–	–	–	–	–	–	5	–	–	–	–	–	–	–	6	–	–	–	–
p_{32}	–	–	–	–	–	5	6	–	–	–	–	–	–	–	–	–	–	–	–	–
p_{33}	–	5	–	–	–	–	–	6	–	–	–	–	–	–	–	–	–	–	–	–
p_{34}	–	–	–	5	–	–	6	–	–	–	–	–	–	–	–	–	–	–	–	–
p_{35}	–	–	–	–	–	–	–	5	6	–	–	–	–	–	–	–	–	–	–	–
p_{36}	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	5	–	6
p_{37}	–	–	–	–	–	–	–	–	5	–	–	–	5	–	–	–	–	–	–	–
p_{38}	–	–	–	–	–	7	–	–	–	–	5	–	–	3	–	–	–	–	–	–
p_{39}	–	–	–	–	–	–	–	–	–	–	7	–	–	–	–	–	–	–	3	4
p_{40}	–	–	–	–	–	–	–	–	–	–	–	–	5	–	–	–	–	–	–	4
p_{41}	–	4	–	5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
p_{42}	–	–	–	–	–	4	–	–	–	–	3	–	–	–	–	6	–	–	–	–
p_{43}	–	–	–	–	5	–	–	–	–	–	3	5	–	–	–	–	–	–	–	–
p_{44}	–	–	4	–	–	6	–	–	–	–	3	–	–	–	–	–	–	–	–	–
p_{45}	–	–	–	–	–	–	–	–	–	–	2	5	–	–	–	–	5	–	–	–
p_{46}	–	–	3	–	–	6	–	–	–	–	–	–	–	3	–	–	–	–	–	–
p_{47}	–	–	–	–	–	–	3	–	–	–	–	–	–	–	–	–	–	4	–	–
p_{48}	–	–	–	5	–	–	–	2	–	–	–	–	–	–	–	–	–	–	–	–
p_{49}	–	–	–	–	–	–	–	–	5	–	–	–	–	–	–	–	–	–	2	–
p_{50}	–	–	–	–	–	–	–	–	–	–	3	–	4	–	–	–	3	–	–	–
p_{51}	–	1	–	–	–	–	7	–	–	–	–	–	–	–	–	–	1	4	–	–
p_{52}	–	–	–	–	–	–	–	4	–	2	–	–	–	–	–	–	–	–	–	–
p_{53}	–	–	3	–	–	–	–	–	–	–	–	–	–	–	–	–	–	3	–	–
p_{54}	–	–	–	–	–	–	–	3	–	–	–	–	–	–	–	–	–	3	–	–
p_{55}	–	–	–	2	–	–	–	–	–	–	–	–	–	–	–	–	–	3	–	–
p_{56}	–	–	–	–	–	–	–	–	–	–	–	1	–	–	–	2	–	–	–	–
p_{57}	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	1	–	1	–

Table 4. The scores in all three approaches. The β_j in the linear approach are shifted by $\mu_{lin} = 0.6698$ and the nonlinear β_j by $\mu_{nonlin} = 2.8864$ in order to achieve the same average scores as in the naive approach. Note that this means a slightly modified righthand side in (5).

Number of paper	Naive approach score	Naive approach rank	Linear model score	Linear model rank	Nonlinear model score	Nonlinear model rank
1	7.000	1	7.557	1	6.549	7
2	7.000	2	6.831	8	6.132	15
3	7.000	3	7.557	2	6.549	8
4	7.000	4	6.315	15	7.538	2
5	6.500	5	7.305	3	6.230	13
6	6.500	6	6.815	9	6.957	5
7	6.500	7	6.602	10	5.150	29
8	6.500	8	7.195	4	7.229	3
9	6.500	9	6.965	6	6.477	10
10	6.500	10	6.249	17	7.651	1
11	6.500	11	6.123	19	6.352	11
12	6.333	12	6.588	12	6.179	14
13	6.000	13	6.891	7	6.482	9
14	6.000	14	5.552	28	5.913	19
15	6.000	15	5.697	25	5.194	28
16	6.000	16	6.598	11	5.462	22
17	6.000	17	5.124	33	5.078	31
18	6.000	18	6.528	13	6.550	6
19	6.000	19	5.989	20	4.922	34
20	6.000	20	5.783	24	5.039	32
21	6.000	21	6.303	16	7.205	4
22	6.000	22	6.483	14	5.227	25
23	6.000	23	7.130	5	6.323	12
24	6.000	24	6.228	18	5.931	18
25	5.500	25	5.846	22	5.971	16
26	5.500	26	4.162	43	4.719	36
27	5.500	27	5.964	21	5.218	26
28	5.500	28	5.509	31	5.456	23
29	5.500	29	4.644	38	4.405	47
30	5.500	30	5.687	26	4.806	35
31	5.500	31	4.917	34	5.210	27
32	5.500	32	4.095	46	4.550	39
33	5.500	33	5.791	23	5.660	21
34	5.500	34	4.162	44	4.513	43
35	5.500	35	5.514	30	5.962	17
36	5.500	36	5.527	29	5.784	20
37	5.000	37	4.911	35	4.691	37
38	5.000	38	5.243	32	4.999	33
39	4.667	39	5.644	27	5.444	24
40	4.500	40	4.769	36	5.089	30
41	4.500	41	4.264	41	4.647	38
42	4.333	42	3.796	47	4.507	44
43	4.333	43	4.668	37	4.532	41
44	4.333	44	4.271	40	4.204	50
45	4.000	45	4.349	39	4.544	40
46	4.000	46	4.136	45	4.434	45
47	3.500	47	3.718	48	4.183	51
48	3.500	48	2.344	54	4.247	48
49	3.500	49	3.047	49	3.855	53
50	3.333	50	2.936	51	4.235	49
51	3.250	51	3.009	50	4.515	42
52	3.000	52	4.238	42	4.430	46
53	3.000	53	2.903	52	3.614	55
54	3.000	54	2.729	53	3.649	54
55	2.500	55	1.702	56	2.973	56
56	1.500	56	0.644	57	−3.745	57
57	1.000	57	2.034	55	3.962	52

Table 5. The reviewers' parameters. Note that the zeros in the α columns of the last row result from the normalization according to (7).

Number i of reviewer	Linear model α_i	Nonlinear model α_i	Nonlinear model γ_i
1	0.9511	4.0540	0.9190
2	0.1620	−0.7132	3.0569
3	0.2494	0.3388	2.1501
4	1.6499	1.3767	1.7379
5	−0.0676	10.3078	0.3896
6	1.7839	0.2520	2.7372
7	2.3662	−0.7730	6.1283
8	0.5962	0.9447	1.4482
9	0.7156	9.8857	0.4435
10	0.7260	−0.8439	3.3569
11	−0.0703	-0.2022	2.2902
12	1.1419	7.9980	0.5621
13	0.8011	4.3951	0.7558
14	−0.4330	0.3932	1.4713
15	0.4097	12.2399	0.4336
16	1.9088	11.6348	0.4356
17	0.1235	−0.7309	4.0056
18	1.2852	2.7802	0.9436
19	−0.8523	−0.6411	1.8889
20	0	0	1.8305

ample, papers p_{17} and p_{23}: p_{17} was (by good luck for the authors) reviewed by reviewers r_7 and r_{10}. As noted above, reviewer r_7 tends to be lenient; the same appears to apply (though to a lesser extent) to reviewer r_{10}. Thus, in the naive approach, paper p_{17} is likely to have been ranked higher than merited. Paper p_{23} was reviewed by r_5 and r_{19}. Reviewer r_5 seems to be neutral with at most a slight tendency of being harsh, reviewer r_{19} exhibits a more distinct tendency towards harshness. Thus, in the two approaches presented here, paper p_{23} is assigned better scores and jumps from rank 23 in the naive approach to rank 5 in the linear and to rank 12 in the nonlinear model. The corresponding mean squared errors (where $n = 116$ is the total number of reviews) are 0.4533 for the linear model and 0.1739 for the nonlinear model. It is not surprising that the additional parameters γ_i reduce the error.

4 Conclusions

In this paper, we introduced two statistical methods for fairer rating (and thus, ranking) of scientific papers based on scores of potentially biased, partially blindfolded reviewers. These methods work well also in cases where each paper is reviewed only by a small number of reviewers; in particular, there is no need for every reviewer to assess each paper. This approach clearly improves on the classical, naive, yet currently common method of averaging the individual reviewers' scores. The linear approach can be carried out by means of existing statistical standard software. The nonlinear approach, however, allows for a more detailed modeling of the behavior of reviewers. On the other hand, it requires more sophisticated software tools to be carried out. The authors assume that Section 3 provides sufficient information for its use, and they offer their help in analyzing data based on a data table like Table 3. We applied both methods to real data from a scientific conference, and pointed out some effects and implications that are visible in the results. This displays their potential to improve decision-making in peer-reviewed scientific publication venues.

Acknowledgments: We thank the M-PREF-2012 reviewers as well as the AAAI-2011 reviewers for their helpful comments.

References

[1] V. Conitzer, M. Rognlie, and L. Xia, 'Preference functions that score rankings and maximum likelihood estimation', in *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pp. 109–115. IJCAI, (July 2009).

[2] *Proceedings of the 3rd International Workshop on Computational Social Choice*, eds., V. Conitzer and J. Rothe, Universität Düsseldorf, 2010.

[3] V. Conitzer and T. Sandholm, 'Common voting rules as maximum likelihood estimators', in *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence*, pp. 145–152. AUAI Press, (2005).

[4] J. Douceur, 'Paper rating vs. paper ranking', *ACM SIGOPS Operating Systems Review*, **43**, 117–121, (2009).

[5] N. Draper and H. Smith, *Applied Regression Analysis*, Wiley Series in Probability and Statistics, John Wiley & Sons, 3rd edn., 1998.

[6] R. Haenni, 'Aggregating referee scores: An algebraic approach', in *Proceedings of the 2nd International Workshop on Computational Social Choice*, eds., U. Endriss and P. Goldberg, pp. 277–288. University of Liverpool, (2008).

[7] K. Koch, *Parameter Estimation and Hypothesis Testing in Linear Models*, Springer, 2nd edn., 1999.

[8] H. Lauw, E. Lim, and K. Wang, 'Summarizing review scores of "unequal" reviewers', in *Proceedings of the 7th SIAM International Conference on Data Mining*. SIAM, (April 2007).

[9] H. Lauw, E. Lim, and K. Wang, 'Bias and controversy in evaluation systems', *IEEE Transactions on Knowledge and Data Engineering*, **20**, 1490–1504, (2008).

[10] A. Neumaier. MINQ – general definite and bound constrained indefinite quadratic programming. WWW document, 1998. Available at http://www.mat.univie.ac.at/~neum/software/minq.

[11] J. Nocedal and S. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, 2nd edn., 2006.

[12] M. Pini, F. Rossi, K. Venable, and T. Walsh, 'Aggregating partially ordered preferences', *Journal of Logic and Computation*, **19**(3), 475–502, (2009).

[13] M. Roos, J. Rothe, and B. Scheuermann, 'How to calibrate the scores of biased reviewers by quadratic programming', in *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pp. 255–260. AAAI Press, (August 2011).

[14] B. Scheuermann, W. Kiess, M. Roos, F. Jarre, and M. Mauve, 'On the time synchronization of distributed log files in networks with local broadcast media', *IEEE/ACM Transactions on Networking*, **17**(2), 431–444, (2009).

[15] R. Sokal and F. Rohlf, *Biometry: The Principles and Practice of Statistics in Biological Research*, W. H. Freeman, 4th edn., 2012.

[16] S. Wright, *Primal-Dual Interior-Point Methods*, SIAM, 1997.

[17] L. Xia and V. Conitzer, 'A maximum likelihood approach towards aggregating partial orders', in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 446–451. IJCAI, (July 2011).

[18] L. Xia, V. Conitzer, and J. Lang, 'Aggregating preferences in multi-issue domains by using maximum likelihood estimators', in *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 399–408. IFAAMAS, (May 2010).

Proportional Representation as Resource Allocation: Approximability Results

Piotr Skowron[1] and **Piotr Faliszewski**[2] and **Arkadii Slinko**[3]

Abstract. We model Monroe's and Chamberlin and Courant's multiwinner voting systems as a certain resource allocation problem. We show that for many restricted variants of this problem, under standard complexity-theoretic assumptions there are no constant-factor approximation algorithms. Yet, we also show cases where good approximation algorithms exist (these variants correspond to optimizing total voter satisfaction under Borda scores, within Monroe's and Chamberlin and Courant's voting systems).

1 Introduction

Resource allocation is one of the most important issues in multiagent systems, equally important both to human societies and to artificial software agents [20]. For example, if there is a set of items (or a set of bundles of items) to distribute among agents then we may use one of many auction mechanisms (see, e.g., [15,20] for an introduction and a review, and numerous recent papers on auction theory for current results). Typically, in auctions if an agent obtains an item (a resource) then this agent has exclusive access to it. In this paper we consider resource allocation for items that can be shared, and we are interested in computing (approximately) optimal assignments (focusing on cases that reduce to multiwinner voting). We do not make any strategic considerations.

Let us explain our resource allocation problem through an example. Consider a company that wants to provide free sport classes to its employees. We have a set $N = \{1, \ldots, n\}$ of employees and a set $A = \{a_1, \ldots, a_m\}$ of classes. Naturally, not every class is equally appealing to each employee and, thus, each employee orders the classes from the most desirable one to the least desirable one. For example, the first employee might have preference order $a_1 \succ a_3 \succ \cdots \succ a_m$, meaning that for him or her a_1 is the most attractive class, a_3 is second, and so on, until a_m, which is least appealing. Further, each class a_i has some maximum capacity cap_{a_i}, that is, a maximum number of people that can comfortably participate, and a cost, denoted c_{a_i}, of opening the class (independent of the number of participants). The company wants to assign the employees to the classes so that it does not exceed its sport-classes budget B and the employees' satisfaction is maximal (or, equivalently, their dissatisfaction is minimal).

There are many ways to measure (dis)satisfaction. For example, we may measure an employee's dissatisfaction as the position of the class to which he or she was assigned in his or her preference order (and satisfaction as m less the voter's dissatisfaction). We may demand that, for example, the maximum dissatisfaction of an employee is as low as possible (minimal satisfaction is as high as possible; in economics this corresponds to egalitarian social welfare) or that the sum of dissatisfactions is minimal (the sum of satisfactions is maximal; this corresponds to the utilitarian approach in economics).

It turns out that our model generalizes two well-known multiwinner voting rules; namely, those of Monroe [13] and of Chamberlin and Courant [7]. Under both these rules voters from the set N submit preference orders regarding alternatives from the set A, and the goal is to select K candidates (the representatives) best representing the voters. For simplicity, let us assume that K divides $\|N\|$.[4] Under Monroe's rule we have to match each selected representative to $\frac{\|N\|}{K}$ voters so that each voter has a unique representative and so that the sum of voters' dissatisfactions is minimal (dissatisfaction is, again, measured by the position of the representative in the voter's preference order). Chamberlin and Courant's rule is similar except that there are no restrictions on the number of voters a given alternative represents (in this case it is better to think of the alternatives as political parties rather than particular politicians). It is easy to see that both methods are special cases of our setting: For example, for Monroe it suffices to set the "cost" of each alternative to be 1, to set the budget to be K, and to set the "capacity" of each alternative to be $\frac{\|N\|}{K}$. We can consider variants of these two systems using different measures of voter (dis)satisfaction, as indicated above (see also the works of Potthoff and Brams [17], Betzler et al. [3] and of Lu and Boutilier [12]).

It is known that both Monroe's method and Chamberlin and Courant's method are NP-hard to compute in essentially all nontrivial settings [3,12,18]. This holds even if various natural parameters of the election are low [3]. Notable exceptions include, e.g., the case where K is bounded by a fixed constant and the case where voter preferences are single-peaked [3].

Nonetheless, Lu and Boutilier [12]—starting from a very different motivation and context—propose to rectify the high computational complexity of Chamberlin and Courant's system by designing approximation algorithms. In particular, they show that if one focuses on the sum of voters' satisfactions, then there is a polynomial-time approximation algorithm with approximation ratio $(1 - \frac{1}{e}) \approx 0.63$ (i.e., their

[1] University of Warsaw, Warsaw, Poland
[2] AGH University of Science and Technology, Krakow, Poland
[3] University of Auckland, Auckland, New Zealand

[4] This assumption does not affect our results. Our algorithms maintain their quality without it. Yet, modeling Monroe's and Chamberlin and Courant's systems without it would be more tedious.

algorithm outputs an assignment that achieves no less than about 0.63 of optimal voter satisfaction). Unfortunately, total satisfaction is a tricky measure. For example, under standard Chamberlin and Courant's system, a $\frac{1}{2}$-approximation algorithm is allowed to match each voter to an alternative somewhere in the middle of this voter's preference order, even if there is a feasible solution that matches each voter to his or her most preferred candidate. On the other hand, it seems that a $\frac{1}{2}$-approximation focusing on total dissatisfaction would give results of very high quality.

The goal of this paper is to provide an analysis of our resource allocation scenario, focusing on approximation algorithms for the special cases of Monroe's and Chamberlin and Courant's voting systems. We obtain the following results:

1. Monroe's and Chamberlin and Courant's systems are hard to approximate up to any constant factor for the case where we measure dissatisfaction, irrespective of whether we measure the total dissatisfaction or the dissatisfaction of the most dissatisfied voter (Theorems 1 and 2).
2. Monroe's and Chamberlin and Courant's systems are hard to approximate within any constant factor for the case where we measure satisfaction of the least satisfied voter (Theorems 3 and 4). However, there are good approximation algorithms for total satisfaction—for the Monroe's system we achieve approximation ratio arbitrarily close to 0.715 (and often a much better one; see Section 4). For Chamberlin and Courant's system we give a polynomial-time approximation scheme (Theorem 9).

Related work. Hardness of winner determination for multiwinner voting rules was studied by Procaccia, Rosenschein, and Zohar [18], by Lu and Boutilier [12] (who also gave the first approximation algorithm for Chamberlin and Courant's system), and by Betzler, Slinko and Uhlman [3]. Naturally, there is also a well-established line of work on winner-determination for single-winner voting rules, with results for, for example, Dodgson's rule [2,5,6,10], Ranked Pairs method [4], and many others.

In the context of resource allocation, our model resembles multi-unit resource allocation with single-unit demand [20, Chapter 11] (see also the work of Chevaleyre et al. [8] for a survey of the most fundamental issues in the multiagent resource allocation theory). The problem of multi-unit resource allocation is mostly addressed in the context of auctions (and so it is referred in the literature as multi-unit auctions); in contrast, we consider the problem of finding a solution maximizing the social welfare given the agents' preferences. More generally, our model is similar to resource allocation with sharable indivisible goods [1,8]. The most substantial difference is that we require each agent to be assigned to exactly one alternative. In the context of resource allocation with sharable items, it is often assumed that the agents' satisfaction is affected by the number of agents using the alternatives (the congestion on the alternatives; compare to congestion games [19]). Finally, it is worth mentioning that in the literature on resource allocation it is common to consider other criteria of optimality, such as envy-freeness [11], Pareto optimality, Nash equilibria [1], and others.

Our paper is very close in spirit (especially in terms of the motivation of the resource allocation problem) to the recent work of Darmann et al. [9].

2 Preliminaries

Alternatives, Profiles, Positional Scoring Functions. For each $n \in \mathbb{N}$, we take $[n]$ to mean $\{1, \ldots, n\}$. We assume that there is a set $N = [n]$ of *agents* and a set $A = \{a_1, \ldots a_m\}$ of *alternatives*. Each agent i has *weight* $w_i \in \mathbb{N}$, and each alternative a has *capacity* $\mathrm{cap}_a \in \mathbb{N}$ and cost $c_a \in \mathbb{N}$. The weight of an agent corresponds to its size (measured in some abstract way). An alternative's capacity gives the total weight of the agents that can be assigned to it, and its cost gives the price of selecting the alternative (the price is the same irrespective of the weight of the agents assigned to the alternative). Further, each agent i has a *preference order* $\succ_i$ over A, i.e., a strict linear order of the form $a_{\pi(1)} \succ_i a_{\pi(2)} \succ_i \cdots \succ_i a_{\pi(m)}$ for some permutation π of $[m]$. For an alternative a, by $\mathrm{pos}_i(a)$ we mean the position of a in i'th agent's preference order. For example, if a is the most preferred alternative for i then $\mathrm{pos}_i(a) = 1$, and if a is the most despised one then $\mathrm{pos}_i(a) = m$. A collection $V = (\succ_1, \ldots, \succ_n)$ of agents' preference orders is called a *preference profile*. We write $\mathcal{L}(A)$ to denote the set of all possible preference orders over A. Thus, for preference profile V of n agents we have $V \in \mathcal{L}(A)^n$.

In our computational hardness proofs, we will often include subsets of alternatives in the descriptions of preference orders. For example, if A is the set of alternatives and B is some nonempty strict subset of A, then by saying that some agent has preference order of the form $B \succ A - B$, we mean that this agent ranks all the alternatives in B ahead of all the alternatives outside of B, and that the order in which this agent ranks alternatives within B and within $A - B$ is irrelevant (and, thus, one can assume any easily computable order).

A *positional scoring function* (PSF) is a function $\alpha^m : [m] \to \mathbb{N}$. A PSF α^m is an *increasing positional scoring function* (IPSF) if for each $i, j \in [m]$, if $i < j$ then $\alpha^m(i) < \alpha(j)$. Analogously, a PSF α^m is a *decreasing positional scoring function* (DPSF) if for each $i, j \in [m]$, if $i < j$ then $\alpha^m(i) > \alpha^m(j)$.

Intuitively, if β^m is an IPSF then $\beta^m(i)$ gives the *dissatisfaction* that an agent suffers from when assigned to an alternative that is ranked i'th on his or her preference order. Thus, we assume that for each IPSF β^m it holds that $\beta^m(1) = 0$ (an agent is not dissatisfied by his or her top alternative). Similarly, a DPSF γ^m measures an agent's satisfaction and we assume that for each DPSF γ^m it holds that $\gamma^m(m) = 0$.

We will often speak of families α of IPSFs (DPSFs) of the form $\{\alpha^m \mid m \in \mathbb{N}, \alpha^m \text{ is a PSF}\}$, where the following holds:

1. If we are dealing with IPSFs, then for each $m \in \mathbb{N}$ it holds that $(\forall i \in [m])[\alpha^{m+1}(i) = \alpha^m(i)]$.
2. If we are dealing with DPSFs, then for each $m \in \mathbb{N}$ it holds that $(\forall i \in [m])[\alpha^{m+1}(i+1) = \alpha^m(i)]$.

In other words, we build our families of IPSFs (DPSFs) by appending (prepending) values to functions with smaller domains. We assume that each function α^m from a family can be computed in polynomial time with respect to m. To simplify notation, we will refer to such families of IPSFs (DPSFs) as *normal* IPSFs (normal DPSFs).

We are particularly interested in normal IPSFs (normal DPSFs) corresponding to the Borda count method. That is, in the families of IPSFs $\alpha^m_{\mathrm{B,inc}}(i) = i - 1$ (in the families of DPSFs $\alpha^m_{\mathrm{B,dec}}(i) = m - i$).

Our Resource Allocation Problem. We consider a prob-

lem of finding function $\Phi : N \to A$ that assigns each agent to some alternative (we will call Φ an *assignment function*). We say that Φ is feasible if for each alternative a it holds that the total weight of the agents assigned to it does not exceed its capacity cap_a. Further, we define the cost of assignment Φ to be $\text{cost}(\Phi) = \sum_{a:\Phi^{-1}(a)\neq\emptyset} c_a$.

Given an IPSF (DPSF) α^m, we consider two *dissatisfaction functions*, $\ell_1^\alpha(\Phi)$ and $\ell_\infty^\alpha(\Phi)$, (two *satisfaction functions*, $\ell_1^\alpha(\Phi)$ and $\min^\alpha(\Phi)$):

1. $\ell_1^\alpha(\Phi) = \sum_{i=1}^n \alpha(\text{pos}_i(\Phi(i)))$.
2. $\ell_\infty^\alpha(\Phi) = \max_{i=1}^n \alpha(\text{pos}_i(\Phi(i)))$ (or, $\min^\alpha(\Phi) = \min_{i=1}^n \alpha(\text{pos}_i(\Phi(i)))$).

The former one measures agents' total dissatisfaction (satisfaction), whereas the latter one considers the most dissatisfied (the least satisfied) agent only. In welfare economics and multiagent resource allocation theory the two metrics correspond to, respectively, utilitarian and egalitarian social welfare. We define our resource allocation problem as follows.

Definition 1 *Let α be a normal IPSF. An instance of α-*Assignment-Inc *problem consists of a set of agents $N = [n]$, a set of alternatives $A = \{a_1, \dots a_m\}$, a preference profile V of the agents, a sequence $(w_1, \dots, w_n)$ of agents' weights, sequences $(\text{cap}_{a_1}, \dots, \text{cap}_{a_m})$ and $(c_{a_1}, \dots, c_{a_m})$ of alternatives' capacities and costs, respectively, and budget $B \in \mathbb{N}$. We ask for the assignment function Φ such that: (1) $\text{cost}(\Phi) \leq B$, (2) $\forall_{a\in A} \sum_{i:\Phi(i)=a} w_i \leq cap_a$, and (3) $\ell_1^\alpha(\Phi)$ is minimized.*

In other words, in α-Assignment-Inc we ask for a feasible assignment that minimizes the total dissatisfaction of the agents without exceeding the budget.

Problem α-Assignment-Dec is defined identically except that α is a normal DPSF and in the third condition we seek to maximize $\ell_1^\alpha(\Phi)$ (that is, in α-Assignment-Dec our goal is to maximize total satisfaction). If we replace ℓ_1^α with ℓ_∞^α in α-Assignment-Inc then we obtain problem α-Minmax-Assignment-Inc, where we seek to minimize the dissatisfaction of the most dissatisfied agent. If we replace ℓ_1^α with $\min^\alpha$ in α-Assignment-Dec then we obtain problem α-Minmax-Assignment-Dec, where we seek to maximize the satisfaction of the least satisfied agent.

Focusing on either satisfaction or dissatisfaction is immaterial from the perspective of the optimal solution, but leads to very different approximation properties.

Clearly, each of our four Assignment problems is NP-complete: Even without costs they reduce to the standard NP-complete Partition problem, where we ask if a set of integers (in our case these integers would be agents' weights) can be split evenly between two sets (in our case, two alternatives with the capacities equal to half of the total agent weight). However, in very many applications (for example, in the sport classes example from the introduction) it suffices to consider unit-weight agents and we focus on this case.

Our four problems can be viewed as generalizations of Monroe's [13] and Chamberlin and Courant's [7] multiwinner voting systems (see the introduction for their definitions). For Monroe's system, it suffices to set the budget $B = K$, the cost of each alternative to be 1, and the capacity of each alternative to be $\frac{\|N\|}{K}$ (for simplicity, throughout the paper we assume that K divides $\|N\|$). We will refer to such variants of our problems as Monroe-Assignment variants. For Chamberlin and Courant's system, it suffices to take the same restrictions as for Monroe's system, except that each alternative has capacity equal to $\|N\|$. We will refer to such variants of our problems as CC-Assignment variants.

Approximation Algorithms. For many normal IPSFs α (e.g., for Borda count), even the above-mentioned restricted versions of the original problem, namely, α-Monroe-Assignment-Inc, α-Minmax-Monroe-Assignment-Inc, α-CC-Assignment-Inc, and α-Minmax-CC-Assignment-Inc are NP-complete [3,18] (the same holds for the Dec variants). Thus, we seek approximate solutions.

Definition 2 *Let β be a real number such that $\beta \geq 1$ ($0 < \beta \leq 1$) and let α be a normal IPSF (a normal DPSF). An algorithm is a β-approximation algorithm for α-*Assignment-Inc *problem (for α-*Assignment-Dec *problem) if on each instance I it returns a feasible assignment Φ that meets the budget restriction and such that $\ell_1^\alpha(\Phi) \leq \beta \cdot \text{OPT}$ (and such that $\ell_1^\alpha(\Phi) \geq \beta \cdot \text{OPT}$), where OPT is the optimal aggregated dissatisfaction (satisfaction) $\ell_1^\alpha(\Phi_{\text{OPT}})$.*

We define β-approximation algorithms for the Minmax variants analogously. For example, Lu and Boutilier [12] present a $(1 - \frac{1}{e})$-approximation algorithm for the case of CC-Assignment-Dec.

Throughout this paper, we will consider each of the Monroe-Assignment and CC-Assignment variants of the problem and for each we will either prove inapproximability with respect to any constant β (under standard complexity-theoretic assumptions) or we will show an approximation algorithm. We use the following NP-complete problems.

Definition 3 *An instance I of* Set-Cover *consists of set $U = [n]$ (called the ground set), family $\mathcal{F} = \{F_1, F_2, \dots, F_m\}$ of subsets of U, and positive integer K. We ask if there exists a set $I \subseteq [m]$ such that $\|I\| \leq K$ and $\bigcup_{i\in I} F_i = U$.* X3C *is a special case of* Set-Cover *where $\|U\|$ is divisible by 3, each member of $\mathcal{F}$ has exactly three elements, and $K = \frac{n}{3}$.*

X3C is NP-hard even if we assume that n is divisible by 2 and each member of U appears in at most 3 sets from $\mathcal{F}$.

3 Hardness of Approximation

In this section we present our inapproximability results for Monroe-Assignment and CC-Assignment variants of the resource allocation problem. In particular, we show that if we focus on voter dissatisfaction (i.e., on the Inc variants) then for each $\beta > 1$, neither Monroe's nor Chamberlin and Courant's system has a polynomial-time β-approximation algorithm. Further, we show that analogous results hold if we focus on the satisfaction of the least satisfied voter.

Naturally, these inapproximability results carry over to more general settings. In particular, unless P = NP, there are no polynomial-time constant-factor approximation algorithms for the general resource allocation problem for the case where we focus on voter dissatisfaction. On the other hand, our results do not preclude good approximation algorithms for the case where we measure agents' total satisfaction.

Theorem 1 *For each normal IPSF α and each constant factor β, $\beta > 1$, there are no polynomial-time β-approximation algorithms for either of α-*Monroe-Assignment-Inc *and α-*Minmax-Monroe-Assignment-Inc*, unless* P = NP.

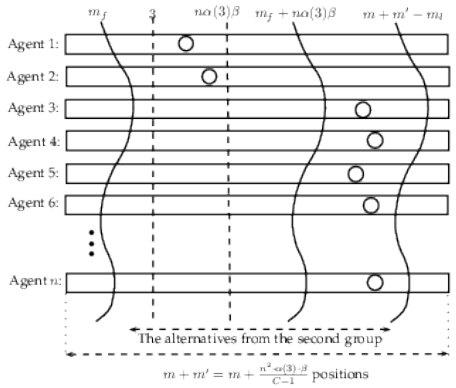


Figure 1. The alignment of the positions in the preference orders of the agents. The positions are numbered from left to right. The left wavy line shows the positions $m_f(\cdot)$, each no greater than 3. The right wavy line shows the positions $m_l(\cdot)$, each higher than $n \cdot \alpha(3) \cdot \beta$. The alternatives from A_2 (one such alternative is illustrated with a circle) are placed only between the peripheral wavy lines. Each alternative from A_2 is placed on the left from the middle wavy line exactly 2 times.

Proof We give a proof for the case of α-Monroe-Assignment-Inc only. Let us fix a normal IPSF α and let us assume, for the sake of contradiction, that there is some constant β, $\beta > 1$, and a polynomial-time β-approximation algorithm $\mathcal{A}$ for α-Monroe-Assignment-Inc.

Let I be an instance of X3C with ground set $U = [n]$ and family $\mathcal{F} = \{F_1, \ldots, F_m\}$ of 3-element subsets of U. W.l.o.g., we assume that n is divisible by 6 and that each member of U appears in at most 3 sets from $\mathcal{F}$.

Given I, we build instance I_M of α-Monroe-Assignment-Inc as follows. We set $N = U$ (that is, the elements of the ground set are the agents) and we set $A = A_1 \cup A_2$, where $A_1 = \{a_1, \ldots, a_m\}$ is a set of alternatives corresponding to the sets from the family $\mathcal{F}$ and A_2, $\|A_2\| = \frac{n^2 \cdot \alpha(3) \cdot \beta}{2}$, is a set of dummy alternatives needed for our construction. We let $m' = \|A_2\|$ and we rename the alternatives in A_2 so that $A_2 = \{b_1, \ldots, b_{m'}\}$. We set $K = \frac{n}{3}$.

We build agents' preference orders using the following algorithm. For each $j \in N$, set $M_f(j) = \{a_i \mid j \in F_i\}$ and $M_l = \{a_i \mid j \notin F_i\}$. Set $m_f(j) = \|M_f(j)\|$ and $m_l(j) = \|M_l(j)\|$; as the frequency of the elements from U is bounded by 3, $m_f(j) \leq 3$. For each agent j we set his or her preference order to be of the form $M_f(j) \succ_j A_2 \succ_j M_l(j)$, where the alternatives in $M_f(j)$ and $M_l(j)$ are ranked in an arbitrary way and the alternatives from A_2 are placed at positions $m_f(j) + 1, \ldots, m_f(j) + m'$ in the way described below (see Figure 1 for a high-level illustration of the construction).

We place the alternatives from A_2 in the preference orders of the agents in such a way that for each alternative $b_i \in A_2$ there are at most two agents that rank b_i among their $n \cdot \alpha(3) \cdot \beta$ top alternatives. The following construction achieves this effect. If $(i + j) \bmod n < 2$, then alternative b_i is placed at one of the positions $m_f(j) + 1, \ldots, m_f(j) + n \cdot \alpha(3) \cdot \beta$ in j's preference order. Otherwise, b_i is placed at a position with index higher than $m_f(j) + n \cdot \alpha(3) \cdot \beta$ (and, thus, at a position higher than $n \cdot \alpha(3) \cdot \beta$). This construction can be implemented because for each agent j there are exactly $m' \cdot \frac{2}{n} = n \cdot \alpha(3) \cdot \beta$ alternatives $b_{i_1}, b_{i_2}, b_{i_{n\alpha(3)\beta}}$ such that $(i_k + j) \bmod n < 2$.

Let Φ be an assignment computed by $\mathcal{A}$ on I_M. We will show that $\ell_1^\alpha(\Phi) \leq n \cdot \alpha(3) \cdot \beta$ if and only if I is a *yes*-instance.

($\Leftarrow$) If there exists a solution for I (i.e., an exact cover of U with $\frac{n}{3}$ sets from $\mathcal{F}$), then we can easily show an assignment in which each agent j is assigned to an alternative from the top $m_f(j)$ positions of his or her preference order (namely, one that assigns each agent j to the alternative $a_i \in A_1$ that corresponds to the set F_i, from the exact cover of U, that contains j). Thus, for the optimal assignment Φ_{OPT} it holds that $\ell_1^\alpha(\Phi_{\text{OPT}}) \leq \alpha(3) \cdot n$. In consequence, $\mathcal{A}$ must return an assignment with the total dissatisfaction at most $n \cdot \alpha(3) \cdot \beta$.

($\Rightarrow$) Let us now consider the opposite direction. We assume that $\mathcal{A}$ found an assignment Φ such that $\ell_1^\alpha(\Phi) \leq n \cdot \alpha(3) \cdot \beta$ and we will show that I is a *yes*-instance of X3C. Since we require each alternative to be assigned to either 0 or 3 agents, if some alternative b_i from A_2 were assigned to some 3 agents, at least one of them would rank him or her at a position worse than $n \cdot \alpha(3) \cdot \beta$. This would mean that $\ell_1^\alpha(\Phi) \geq n \cdot \alpha(3) \cdot \beta + 1$. Analogously, no agent j can be assigned to an alternative that is placed at one of the $m_l(j)$ bottom positions of j's preference order. Thus, only the alternatives in A_1 have agents assigned to them and, further, if agents x, y, z, are assigned to some $a_i \in A_1$, then it holds that $F_i = \{x, y, z\}$ (we will call each set F_i for which alternative a_i is assigned to some agents x, y, z *selected*). Since each agent is assigned to exactly one alternative, the selected sets are disjoint. Since the number of selected sets is $K = \frac{n}{3}$, it must be that the selected sets form an exact cover of U. So I is a *yes*-instance of X3C. □

Is Theorem 1 an artifact of our strict bound on the cost? This seems unlikely as there is also no β-γ-approximation algorithm that finds an assignment with the following properties: (1) the aggregated dissatisfaction $\ell_1^\alpha(\Phi)$ is at most β times higher than the optimal one, (2) the number of alternatives to which agents are assigned is at most γK and (3) each selected alternative, is assigned to no more than $\gamma\lceil\frac{n}{K}\rceil$ and no less than $\frac{1}{\gamma}\lceil\frac{n}{K}\rceil$ agents.

Result analogous to Theorem 1 holds for CC as well.

Theorem 2 *For each normal IPSF α and each constant factor β, $\beta > 1$, there are no polynomial-time β-approximation algorithms for either of* α-CC-Assignment-Inc *and* α-Minmax-CC-Assignment-Inc, *unless* P = NP.

The above results show that approximating the minimal dissatisfaction of agents is difficult. On the other hand, if we focus on agents' total satisfaction then constant-factor approximation exist (see [12] and the next section). Yet, the case of satisfying the least satisfied voter remains hard.

Theorem 3 *For each normal DPSF α (where each entry is coded in unary) and each constant factor β, $0 < \beta \leq 1$, there is no β-approximation algorithm for* α-Minmax-Monroe-Assignment-Dec *unless* P = NP.

Unfortunately, for the case of Minmax-CC-Assignment-Dec family of problems our inapproximability argument holds for the case of Borda DPSF only and we show a weaker collapse of W[2] to FPT. (See the book of Niedermeier [14] for an overview of parametrized complexity theory.)

Theorem 4 *Let $\alpha^m_{\text{B,dec}}$ be the Borda DPSF ($\alpha^m_{\text{B,dec}}(i) = m - i$). For each constant factor β, $0 < \beta \leq 1$, there is no β-approximation algorithm for* $\alpha^m_{\text{B,dec}}$-Minmax-CC-Assignment-Dec *unless* FPT = W[2].

Monroe	Chamberlin-Courant
Maximizing total satisfaction (Borda scores)	
Randomized algorithms: (a) $0.715-\epsilon$; (b) $\frac{1+\frac{K}{m}-\frac{K^2}{m^2-m}+\frac{K^3}{m^3-m^2}}{2}-\epsilon$ Deterministic algorithm: $1-\frac{K-1}{2(m-1)}-\epsilon$	Deterministic algorithm: $1-\frac{2\mathrm{w}(K)}{K}$, PTAS For general DPSFs, there is a $(1-\frac{1}{e})$-approximation algorithm [12]
Minimizing total (minimal) dissatisfaction, maximizing minimal satisfaction	
Inapproximability: Theorems 1 and 3	Inapproximability: Theorems 2 and 4

Table 1. Summary of results for MONROE and CC variants.

4 Approximation Algorithms

We now turn to approximation algorithms for Monroe's and Chamberlin and Courant's rules. Indeed, if one focuses on agents' total satisfaction then it is possible to obtain high-quality approximation results. We show the first non-trivial approximation algorithms for Monroe's system and the first polynomial-time approximation scheme (PTAS) for Chamberlin-Courant's system. These results stand in a sharp contrast to those from the previous section, where we have shown that approximation is hard for essentially all remaining variants of the problem.

Hardness of α-MONROE/CC-ASSIGNMENT lays in selecting the alternatives to assign to agents. Given those, finding the optimal assignment is easy through network-flow arguments (this is implicit in the paper of Betzler et al. [3]).

Monroe's System. A natural iterative approach to solve $\alpha_{\mathrm{B,dec}}$-MONROE-ASSIGNMENT-DEC is to in each step pick some not-yet-assigned alternative a_i (using some criterion) and assign him or her to those $\lceil\frac{N}{K}\rceil$ agents that (a) are not assigned to any other alternative yet, and (b) whose satisfaction of being matched with a_i is maximal. This idea—implemented formally in Algorithm 1—works very well in many cases. (For each positive integer k, we let $H_k=\sum_{i=1}^{k}\frac{1}{i}$ be the k'th harmonic number. Recall that $H_k=\Theta(\log k)$.)

Lemma 5 *Algorithm 1 is a $(1-\frac{K-1}{2(m-1)}-\frac{H_K}{K})$-approximation algorithm for $\alpha_{\mathrm{B,dec}}$-MONROE-ASSIGNMENT-DEC that runs in polynomial time.*

Algorithm 1: The algorithm for MONROE-ASSIGNMENT.

Notation: $\Phi\leftarrow$ a map defining a partial assignment, iteratively built by the algorithm.
$\Phi^{\leftarrow}\leftarrow$ the set of agents for which the assignment is already defined.
$\Phi^{\rightarrow}\leftarrow$ the set of alternatives already used in the assignment.
if $K\leq 2$ **then**
  return the solution given by the algorithm of Betzler et al. [3].
$\Phi=\{\}$
for $i\leftarrow 1$ **to** K **do**
  $score\leftarrow\{\}$, $bests\leftarrow\{\}$
  foreach $a_i\in A\setminus\Phi^{\rightarrow}$ **do**
    $agents\leftarrow$ sort $N\setminus\Phi^{\leftarrow}$ so that $j\prec k$ in $agents$ $\implies pos_j(a_i)\leq pos_k(a_i)$
    $bests[a_i]\leftarrow$ chose first $\lceil\frac{N}{K}\rceil$ elements from $agents$
    $score[a_i]\leftarrow\sum_{j\in bests}(m-pos_j(a_i))$
  $a_{best}\leftarrow\mathrm{argmax}_{a\in A\setminus\Phi^{\rightarrow}}score[a]$
  foreach $j\in bests[a_{best}]$ **do**
    $\Phi[j]\leftarrow a_{best}$

Proof Our algorithm computes an optimal solution for $K\leq 2$. Thus we assume $K\geq 3$. Let us consider the situation in the algorithm after the i'th iteration of the outer loop (we have $i=0$ if no iteration has been executed yet). So far, the algorithm has picked i alternatives and assigned them to $i\frac{n}{K}$ agents (recall that for simplicity we assume that K divides n evenly). Hence, each agent has $\lceil\frac{m-i}{K-i}\rceil$ unassigned alternatives among his or her $i+\lceil\frac{m-i}{K-i}\rceil$ top-ranked alternatives. By pigeonhole principle, this means that there is an unassigned alternative a_ℓ who is ranked among top $i+\lceil\frac{m-i}{K-i}\rceil$ positions by at least $\frac{n}{K}$ agents. To see this, note that there are $(n-i\frac{n}{K})\lceil\frac{m-i}{K-i}\rceil$ slots for unassigned alternatives among the top $i+\lceil\frac{m-i}{K-i}\rceil$ positions in the preference orders of unassigned agents, and that there are $m-i$ unassigned alternatives. As a result, there must be an alternative a_ℓ for whom the number of agents that rank him or her among the top $i+\lceil\frac{m-i}{K-i}\rceil$ positions is at least: $\frac{1}{m-i}\left((n-i\frac{n}{K})\lceil\frac{m-i}{K-i}\rceil\right)\geq\frac{n}{m-i}\left(\frac{K-i}{K}\right)\left(\frac{m-i}{K-i}\right)=\frac{n}{K}$. In consequence, the $\lceil\frac{n}{K}\rceil$ agents assigned in the next step of the algorithm will have the total satisfaction at least $\lceil\frac{n}{K}\rceil\cdot(m-i-\lceil\frac{m-i}{K-i}\rceil)$. Thus, summing over the K iterations, the total satisfaction guaranteed by the assignment Φ computed by Algorithm 1 is at least the following value (see the comment below for the fourth inequality; for the last inequality we assume $K\geq 3$): $\ell_1^{\alpha_b}(\Phi)\geq\sum_{i=0}^{K-1}\frac{n}{K}\cdot\left(m-i-\lceil\frac{m-i}{K-i}\rceil\right)\geq\sum_{i=0}^{K-1}\frac{n}{K}\cdot\left(m-i-\frac{m-i}{K-i}-1\right)=\sum_{i=1}^{K}\frac{n}{K}\cdot\left(m-i-\frac{m-1}{K-i+1}+\frac{i-2}{K-i+1}\right)=\frac{n}{K}\left(\frac{K(2m-K-1)}{2}-(m-1)H_K+K(H_K-1)-H_K\right)>(m-1)n\left(1-\frac{K-1}{2(m-1)}-\frac{H_K}{K}\right)$. The fourth equality holds because $K(H_K-1)-H_K=\sum_{i=1}^{K}\left(\frac{K}{i}-1\right)-H_K=\sum_{i=1}^{K}\left(\frac{K}{K-i+1}-1\right)-H_K=\sum_{i=1}^{K}\frac{i-1}{K-i+1}-H_K=\sum_{i=1}^{K}\frac{i-2}{K-i+1}$. If each agent were assigned to his or her top alternative, the total satisfaction would be equal to $(m-1)n$. Thus we get that $\frac{\ell_1^{\alpha_{\mathrm{B,dec}}}(\Phi)}{\mathrm{OPT}}\leq 1-\frac{K-1}{2(m-1)}-\frac{H_K}{K}$. □

In the above proof we measure the quality of our assignment against a perhaps-impossible solution, where each agent is assigned to his or her top alternative. Thus for relatively large m and K, and small $\frac{K}{m}$ ratio, the algorithm can achieve a close-to-ideal solution irrespective of the voters' preference orders. This is an argument in favor of Monroe's system.

Betzler et al. [3] showed that for each fixed constant K, $\alpha_{\mathrm{B,dec}}$-MONROE-ASSIGNMENT-DEC can be solved in polynomial time. Thus, for small values of K for which the fraction $\frac{H_K}{K}$ affects the approximation guarantees of Algorithm 1 too much, we can use this polynomial-time algorithm to find an optimal solution. This means that we can essentially disregard the $\frac{H_K}{K}$ part of Algorithm 1's approximation ratio. In

consequence, the quality of the solution produced by Algorithm 1 most strongly depends on the ratio $\frac{K-1}{m-1}$. In most cases we can expect it to be small. If it is not, we can use an algorithm that randomly samples K alternatives and matches them optimally to the agents.

Lemma 6 *A single sampling step of the randomized algorithm for* $\alpha_{\mathrm{B,dec}}$-Monroe-Assignment-Dec *achieves expected approximation ratio of* $\frac{1}{2}(1+\frac{K}{m}-\frac{K^2}{m^2-m}+\frac{K^3}{m^3-m^2})$. *Let* p_ϵ *denote the probability that the relative deviation between the obtained total satisfaction and the expected total satisfaction is higher than* ϵ*; for* $K \geq 8$ *we have* $p_\epsilon \leq \exp\left(-\frac{K\epsilon^2}{128}\right)$.

The threshold for $\frac{K}{m}$, where the randomized algorithm is (in expectation) better than the greedy algorithm is about 0.57. Combining the two algorithms, we get the next result.

Theorem 7 *For each fixed* ϵ*, there is an algorithm that provides a* $(0.715-\epsilon)$*-approximate solution for the problem* $\alpha_{\mathrm{B,dec}}$-Monroe-Assignment-Dec *with probability* λ*, in time polynomial with respect to the input size and* $-\log(1-\lambda)$.

Chamberlin and Courant's System. Let us now move on to the Chamberlin and Courant's system. It turns out that the additional freedom of this system allows us to design a polynomial-time approximation scheme for $\alpha_{\mathrm{B,dec}}$-CC-Assignement-Dec.

The idea of our method is to compute a certain value x and to greedily seek an assignment that (approximately) maximizes the number of agents assigned to their top-x alternatives (and match the remaining agents arbitrarily; recall that for nonnegative real numbers, Lambert's W function, $\mathrm{w}(x)$, is defined to be the solution of the equality $x = \mathrm{w}(x)e^{\mathrm{w}(x)}$.)

Lemma 8 *There is a polynomial-time* $(1-\frac{2\mathrm{w}(K)}{K})$*-approximation algorithm for* $\alpha_{\mathrm{B,dec}}$-CC-Assignement-Dec.

(Independently, Oren [16] gave a sampling-based algorithm with expected approximation ratio of $(1-\frac{1}{K+1})(1+\frac{1}{m})$.) Since for each $\epsilon > 0$ there is a value K_ϵ such that for each $K > K_\epsilon$ it holds that $\frac{2\mathrm{w}(K)}{K} < \epsilon$, and $\alpha_{\mathrm{B,dec}}$-CC-Assignment problem can be solved optimally in polynomial time for each fixed constant K (see the work of Betzler et al. [3]), there is a polynomial-time approximation scheme (PTAS) for $\alpha_{\mathrm{B,dec}}$-CC-Assignment (i.e., a family of algorithms such that for each fixed β, $0 < \beta < 1$, there is a polynomial-time β-approximation algorithm for $\alpha_{\mathrm{B,dec}}$-CC-Assignment).

Theorem 9 *There is a PTAS for* $\alpha_{\mathrm{B,dec}}$-CC-Assignment.

5 Conclusions

We have defined a certain (shared) resource allocation problem and have shown that it generalizes multiwinner voting rules of Monroe and of Chamberlin and Courant. Since these rules are hard to compute [3,12,18], we have investigated the possibility of computing approximate solutions. Our results are summarized in Table 1. Except for the case of maximizing total voter satisfaction, both rules turned out to be hard to approximate. However, for the the case of maximizing total voter satisfaction, we have obtained the first nontrivial approximation algorithms for Monroe's rule (our randomized algorithm obtains approximation ratios arbitrarily close to 0.715) and the first PTAS for Chamberlin and Courant's rule. Natural open problems include seeking a PTAS for Monroe's system and empirical evaluation of our algorithms.

Acknowledgements The authors were supported in part by AGH Univ. grant 11.11.120.865, by the Foundation for Polish Science's Homing/Powroty program, and by EU's Human Capital Program "National PhD Programme in Mathematical Sciences" carried out at the University of Warsaw.

REFERENCES

[1] S. Airiau and U. Endriss. Multiagent resource allocation with sharable items: Simple protocols and nash equilibria. In *Proc. of AAMAS-2010*, pages 167–174, May 2010.
[2] J. Bartholdi, III, C. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989.
[3] N. Betzler, A. Slinko, and J. Uhlmann. On the computation of fully proportional representation. Technical report, U. of Auckland, November 2011.
[4] M. Brill and F. Fisher. The price of neutrality for the ranked pairs method. In *Proc. of AAAI-2012*, 2012. To appear.
[5] I. Caragiannis, J. Covey, M. Feldman, C. Homan, C. Kaklamanis, N. Karanikolas, A. Procaccia, and J. Rosenschein. On the approximability of Dodgson and Young elections. In *Proc. of SODA-2009*, pages 1058–1067, January 2009.
[6] I. Caragiannis, C. Kaklamanis, N. Karanikolas, and A. Procaccia. Socially desirable approximations for Dodgson's voting rule. In *Proc. of EC-2010*, pages 253–262, June 2010.
[7] B. Chamberlin and P. Courant. Representative deliberations and representative decisions: Proportional representation and the borda rule. *American Political Science Review*, 77(3):718–733, 1983.
[8] Y. Chevaleyre, P. Dunne, U. Endriss, J. Lang, M. Lematre, N. Maudet, J. Padget, S. Phelps, J. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.
[9] A. Darmann, E. Elkind, S. Kurz, J. Lang, J. Schauer, and G. Woeginger. Group activity selection problem. In *Workshop Notes of COMSOC-2012*, 2012. To appear.
[10] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6):806–825, 1997.
[11] R. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *Proc. of ACM EC-2004*, pages 125–131, May 2004.
[12] T. Lu and C. Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proc. of IJCAI-2011*, pages 280–286, 2011. See also a version from COMSOC-2010.
[13] B. Monroe. Fully proportional representation. *American Political Science Review*, 89(4):925–940, 1995.
[14] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
[15] N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.
[16] J. Oren. Personal communication, 2012.
[17] R. F. Potthoff and S. J. Brams. Proportional representation: Broadening the options. *Journal of Theoretical Politics*, 10(2):147–178, 1998.
[18] A. Procaccia, J. Rosenschein, and A. Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30(3):353–362, April 2008.
[19] R. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.
[20] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

A Non-Monotonic Goal Specification Language for Planning with Preferences

Tran Cao Son[*] and **Enrico Pontelli**[*] and **Chitta Baral**[+ 1]

Abstract. This paper introduces a default logic based approach to defining goal specification languages that can be non-monotonic and allow for the specification of inconsistencies and priorities among goals. The paper starts by presenting a basic goal specification language for planning with preferences. It then defines goal default theories (resp. with priorities) by embedding goal formulae into default logic (resp. prioritizing default logic). It is possible to show that the new language is general, as it can express several features of previously developed goal specification languages. The paper discusses how several other features can be subsumed by extending the basic goal specification language. Finally , we identify features that might be important in goal specification that cannot be expressed by our language.

1 Introduction

An important component of autonomous agent design is *goal specification*. In classical planning, goals deal with reaching one of a particular set of states. Nevertheless, often goals of agents are not just about reaching a particular state; goals are often about satisfying desirable *conditions* imposed on the trajectory. For example, a person can have the following desire in preparing travel plans to conferences:

> **(*)** *I prefer to fly to the conference site (since it is usually too far to drive).*

The user's preference restricts the means that can be used in achieving her goal of reaching the conference site, which leads to the selection of a plan that reaches the conference site by airplane, whenever possible. Ultimately, this affects what actions the person should take in order to achieve the goal.

These observations led to the development of languages for the specification of *soft goals* in planning, e.g., $\mathcal{PP}$ introduced in [14] and modified in [6]. In $\mathcal{PP}$, a *basic desire* is a temporal formula describing desirable properties of a plan. *Atomic* and *general preferences* are particular classes of formulae built over basic desires. A preference formula Φ defines a preference order $\prec_\Phi$ among the trajectories that achieve the *hard goal* of the problem, i.e., for every pair of trajectories α and β, $\alpha \prec_\Phi \beta$ indicates that α is preferable to β. $\prec_\Phi$ is often a partial order and its definition relies on the notion of satisfaction between trajectories and a preference specification. Similar ideas have been considered in the planning community and led to extensions of the planning domain description language PDDL, with features for representing classes of preferences over plans using temporal extended preferences (e.g., [10]).

[1] [*]Department of Computer Science New Mexico State University, Las Cruces, New Mexico, USA, email: tson|epontell@cs.nmsu.edu and [+]Department of Computer Science and Engineering, Arizona State University, Tempe, Arizona, USA, email: chitta@asu.edu

In [4], the authors argue that a goal specification language should be *non-monotonic* for various reasons, such as elaboration tolerance and simplicity of goal specification. For example, the same traveler with the preference **(*)** would probably not mind driving at most three hours to the conference site if the only flight to the destination requires to travel the day before the conference starts. In this case, her preference becomes:

> **(**)** *Normally, I prefer to fly to the conference site (since it is usually too far to drive). However, if there are no flights on the same day of the conference and the driving time is at most three hours, then I will drive.*

To address this issue, an extension of LTL [11], called N-LTL, has been proposed, allowing weak and strong exceptions to certain rules. A weakness of this language is that it requires the classification of weak and strong exceptions when a goal is specified. In [5], the language ER-LTL is introduced to address this limitation of N-LTL. Similarly to $\mathcal{PP}$, the semantics of N-LTL and ER-LTL relies on the notion of satisfaction between plans and N-LTL or ER-LTL specifications. Observe that the issue of non-monotonicity is dealt within $\mathcal{PP}$ and in the extensions of PDDL by revising the soft goals, which is an approach that N-LTL specifically tries to avoid.

We observe that the focus of the work in [1, 4, 5, 6, 10] is on classical planning, i.e., the planning domains are deterministic and the initial state is complete, while the work in [14] considers non-deterministic domains and only discusses preferences among weak plans. In [2], it is argued that plans for non-deterministic domains should be *policies* (i.e., a partial function from the set of states to the set of actions) and the language π-CTL* is developed for specifying goals in non-deterministic domains. π-CTL* is an extension of CTL* [9] with two modalities A_π and E_π for considering all or some trajectories w.r.t. a given policy. In [3], the language π-CTL* is extended with quantifiers over policies to increase its expressiveness. Policies satisfying a goal specification are viewed as the solutions of a planning problem.

In this paper, we explore an approach based on prioritizing default logic for defining a goal specification language. The new language, called *goal default theories with priorities*, is a variation of prioritizing default logic, in which formulae occurring within a default can be temporal extended preference formulae. We show that the core of the new language subsumes several features from existing goal languages and can be extended to subsume several other features from other goal languages. Finally, we discuss the possible applications of the new language in the study of existing goal languages and the development of new ones.

2 Background

In this section, we briefly review the basic definitions of planning, linear temporal logic (LTL) and its extension for specifying prefer-

ences in planning.

2.1 LTL and Temporal Extended Preferences

Let $\mathcal{L}$ be a propositional language. By $\langle p \rangle$ we denote a propositional formula from $\mathcal{L}$. LTL-formulae are defined by the following syntax

$$\begin{array}{rcl} \langle f \rangle & ::= & \langle p \rangle \mid \langle f \rangle \wedge \langle f \rangle \mid \langle f \rangle \vee \langle f \rangle \mid \\ & & \neg\langle f \rangle \mid \bigcirc \langle f \rangle \mid \Box\langle f \rangle \mid \Diamond\langle f \rangle \mid \langle f \rangle \mathsf{U} \langle f \rangle \end{array} \quad (1)$$

The semantics of LTL-formulae is defined with respect to sequences of interpretations of $\mathcal{L}$. For later use, we will refer to an interpretation of $\mathcal{L}$ as a *state* and a possibly infinite sequence of interpretations of $\mathcal{L}$, $s_0, s_1, \ldots$, as a *trajectory*. For a trajectory $\sigma = s_0, s_1, \ldots$, by σ_i we denote the suffix $s_i, s_{i+1}, \ldots$ of σ. A trajectory $\sigma = s_0, s_1, \ldots$ *satisfies* an LTL-formula f, denoted by $\sigma \models f$, if $\sigma_0 \models f$ where

- $\sigma_j \models p$ iff $s_j \models p$
- $\sigma_j \models \neg f$ iff $\sigma_j \not\models f$
- $\sigma_j \models f_1 \wedge f_2$ iff $\sigma_j \models f_1$ and $\sigma_j \models f_2$
- $\sigma_j \models f_1 \vee f_2$ iff $\sigma_j \models f_1$ or $\sigma_j \models f_2$
- $\sigma_j \models \bigcirc f$ iff $\sigma_{j+1} \models f$
- $\sigma_j \models \Box f$ iff $\sigma_k \models f$, for all $k \geq j$
- $\sigma_j \models \Diamond f$ iff $\sigma_i \models f$ for some $i \geq j$
- $\sigma_j \models f_1 \mathsf{U} f_2$ iff there exists $k \geq j$ such that $\sigma_k \models f_2$ and for all i, $j \leq i < k$, $\sigma_i \models f_1$.

A finite trajectory $s_0, \ldots, s_n$ satisfies an LTL-formula f if its extension $s_0, \ldots, s_n, s_{n+1}, \ldots$ satisfies f, where $s_k = s_n$ for $k > n$. In order to deal with planning problems, LTL is extended with the following constructs

$$\begin{array}{c} \texttt{at\_end}\ \langle p \rangle \mid \langle p \rangle\ \texttt{sometime\_before}\ \langle p \rangle \mid \\ \langle p \rangle\ \texttt{sometime\_after}\ \langle p \rangle \end{array} \quad (2)$$

Formulae of the extended LTL are referred to as *Temporal Extended Preferences (TEP)*. Note that the last two are syntactic sugars for LTL formulae. Temporal extended preferences are interpreted over finite trajectories. The notion of satisfaction for standard LTL-formulae is defined as above, while satisfaction of TEP formulae is as follows: given a finite trajectory $\sigma = s_0, \ldots, s_n$:

- $\sigma \models \texttt{at\_end}\ p$ iff $s_n \models p$;
- $\sigma \models p_1\ \texttt{sometime\_before}\ p_2$ iff for every i, $0 \leq i \leq n$, if $\sigma_i \models p_1$ then $\sigma_j \models p_2$ for some $i \leq j \leq n$; and
- $\sigma_j \models p_1\ \texttt{sometime\_after}\ p_2$ iff for every i, $0 \leq i \leq n$, if $\sigma_i \models p_1$ then $\sigma_j \models p_2$ for some $0 \leq j < i \leq n$.

2.2 Planning

In this paper, we describe a dynamic domain as a labeled transition system $T = (F, A, S, L)$, where:

- F is a set of fluents (or propositions),
- A is a set of actions,
- S is a set of interpretations (or states) of F, and
- $L \subseteq S \times A \times S$.

Each triple $\langle s_1, a, s_2 \rangle \in L$ indicates that the execution of the action a in the state s_1 might result in the state s_2. T is *deterministic* if for each state s and action a, L contains at most one triple $\langle s, a, s_2 \rangle$; otherwise, T is non-deterministic.

Given a transition system T, a finite or infinite sequence $s_0 a_0 s_1 a_1 \ldots s_n a_n s_{n+1} \ldots$ of alternate states and actions is called a *run* if $\langle s_i, a_i, s_{i+1} \rangle \in L$ for every $i = 0, \ldots$ A *policy* π in a transition system T is a partial function $\pi : S \to A$ from the set of states to the set of actions. A run $s_0 a_0 s_1 a_1 \ldots s_k a_k s_{k+1} \ldots$ is said to be induced by a policy π if $a_i = \pi(s_i)$ for every $i = 0, \ldots, k, \ldots$

Definition 1. *A* planning problem *is a triple $\langle T, S_i, S_f \rangle$ where $T = (F, A, S, L)$ is a transition system, $S_i \subseteq S$ is the set of* initial *states, and $S_f \subseteq S$ is the set of* final *states.*

Intuitively, a planning problem asks for a *plan* which transforms the transition system from any state belonging to S_i to some state in S_f. In the rest of the discussion, we assume S_i and S_f to be finite sets. We distinguish two classes of planning problems:

- *Deterministic planning*: in this case, T is deterministic and a *solution* (or *plan*) of $\langle T, S_i, S_f \rangle$ is an action sequence $[a_0; \ldots; a_n]$ such that, for every $s_0 \in S_i$, $s_0 a_0 s_1 a_1 \ldots a_n s_{n+1}$ is a run in T and $s_{n+1} \in S_f$;
- *Non-deterministic planning*: in this case, T is non-deterministic and a solution (or plan) of $\langle T, S_i, S_f \rangle$ is a policy π such that, for every $s_0 \in S_i$ and every run induced by π in T, π is finite and is of the form $s_0 a_0 s_1 a_1 \ldots s_k a_k s_{k+1}$ where $s_{k+1} \in S_f$.

In the following, whenever we refer to a possible plan in a transition system T, we mean a sequence of actions (resp. a policy) if T is deterministic (resp. non-deterministic) that can generate a correct run. Let us illustrate these basic definitions using the following simple example.

Example 1. *Consider a transportation robot. There are different locations, say $l_1, \ldots, l_k$, whose connectivity is given by a graph and there might be different objects at each location. Let O be a set of objects. The robot can travel between two directly connected locations. It can pick up objects at a location, hold them, drop them, and carry them between locations. We assume that, for each pair of connected locations l_i and l_j, the robot has an action $a_{i,j}$ for traveling from l_i to l_j. The robot can hold only one object at a time. The domain can be represented by a transition system $T_1 = (F, A, S, L)$:*[2]

- *F contains the following types of propositions:*
  - *$at(i)$ denotes that the robot is at the location l_i;*
  - *$o\_at(o, i)$ denotes that the object o is at the location l_i;*
  - *$h(o)$ denotes that the robot is holding the object o.*
- *A contains of the following types of actions:*
  - *$a_{i,j}$ the robot moves from l_i to l_j;*
  - *$release(o)$ the robot drops the object o;*
  - *$pickup(o)$ the robot picks up the object o.*
- *S contains the interpretations of F which satisfy the basic constraints, such as the robot is at one location at a time, it holds only one object, etc.*
- *L contains transitions of the form $\langle s, a, s' \rangle$ such that s' is the result of the execution of a in s; for example, if $a = a_{i,j}$ and $at(i) \in s$ then $s' = s \setminus \{at(i)\} \cup \{at(j)\}$.*

T_1 is a deterministic transition system. We will also refer to T_2 as the non-deterministic version of T_1 by defining $T_2 = (F, A, S, L')$ where $L' = L \cup \{\langle s_i, a_{i,j}, s_i \rangle \mid a_{i,j} \in A\}$ and $at(i) \in s$. Intuitively, T_2 encodes the fact that the action $a_{i,j}$ might fail and, when it does, the robot will stay where it was after the execution of $a_{i,j}$.

A planning problem P in this domain is given by specifying the initial location of the robot and of the objects and the final location of the robot and of the objects. It is deterministic (resp. non-deterministic) if T_1 (resp. T_2) is considered.

For example, $P_i = \langle T_i, \{\{at(1)\}\}, S_f \rangle$ where for each $s \in S_f$, $at(k) \in s$ is a planning problem for T_i. A solution for P_1 is a sequence $[a_{1,2}; \ldots; a_{k-1,k}]$. On the other hand, a solution for P_2 is a policy π defined by $\pi(s) = a_{t,t+1}$ iff $at(t) \in s$ for $t < k$.

[2] We simplify the definitions of S and L for readability.

3 A Basic Goal Specification Language for Planning with Preferences

In the literature, a planning problem with preferences is defined as a pair (P, Φ) of a planning problem $P = \langle T, S_i, S_f \rangle$, where $T = (F, A, S, L)$, and a preference formula Φ in a goal specification language. A plan δ of P is called a *preferred plan* if it is a plan for P and satisfies Φ, where the notion of satisfaction of a preference formula by a plan is language dependent.

In general, we can characterize a goal specification language $\mathcal{G}$ over a transition system T by a set of preference formulae $\mathcal{F}$ and a satisfaction relation $\models_{\mathcal{G}}$ between the set of possible plans of T and formulae in $\mathcal{F}$. We will write $\delta \models_{\mathcal{G}} \Phi$ to denote that the plan δ satisfies the formula Φ under the language $\mathcal{G}$.

For later use, we will define a *basic goal specification language* for a transition system $T = (F, A, S, L)$, written as $\mathcal{G}_b = (\mathcal{F}_b, \models_{\mathcal{G}_b})$, as follows:

- the set of preference formulae $\mathcal{F}_b$ is the set of TEP-formulae over $F \cup A$, and
- for a planning problem $P = \langle T, S_i, S_f \rangle$, $\models_{\mathcal{G}_b}$ is defined as follows:
  - if T is deterministic, a plan $\delta = [a_0, \ldots, a_n]$ for a planning problem P is said to satisfy a formula Φ in $\mathcal{F}_b$ if for every $s_0 \in S_i$, $s_0 a_0 s_1 a_1 \ldots a_n s_{n+1}$ is a run in T and $(s_0 \cup \{a_0\}), \ldots, (s_n \cup \{a_n\}), s_{n+1}$ is a trajectory satisfying Φ (in the TEP-language over $F \cup A$);
  - if T is non-deterministic, a solution (policy) π for P is said to satisfy a formula Φ in $\mathcal{F}_b$ if for every $s_0 \in S_i$ and every run $s_0 a_0 s_1 a_1 \ldots s_k a_k s_{k+1}$ in T induced by π, $(s_0 \cup \{a_0\}), \ldots, (s_n \cup \{a_n\}), s_{n+1}$ is a trajectory satisfying Φ (in the TEP-language over $F \cup A$).

In the following, we will assume that any goal specification language $\mathcal{G}$ is a conservative extension of $\mathcal{G}_b$, i.e., (*i*) $\mathcal{G}$ contains all formulae in $\mathcal{G}_b$; and (*ii*) for every planning problem P and a formula Φ in $\mathcal{G}$, if $\Phi \in \mathcal{G}_b$ and $\delta \models_{\mathcal{G}_b} \Phi$ with respect to $\mathcal{G}_b$ then $\delta \models_{\mathcal{G}} \Phi$ with respect to $\mathcal{G}$.

Example 2. *Some preference formulae in $\mathcal{G}_b$ for the transition systems in Ex. 1 are:*

- $\Diamond at(2)$*: the robot should visit the location l_2 during the execution of the plan;*
- $at(1) \wedge \Diamond at(2)$*: the robot must (*i*) start in a state satisfying $at(1)$ (or the robot is at the location l_1 initially); and (*ii*) visit the location l_2 at some point during the execution of the plan;*
- $\Box[at(2) \Rightarrow (\bigvee_{i \neq 2} a_{2i})]$*: whenever the robot visits l_2, it should leave that location immediately by executing an action going to one of its neighbors;*
- $h(o) \Rightarrow \bigcirc \bigcirc \neg h(o)$*: if the robot holds an object o in the initial state then it should release o after the execution of one action;*
- $\Box[h(o) \Rightarrow \bigcirc \bigcirc \neg h(o)]$*: whenever the robot holds an object o it should release o after the execution of an action;*
- $h(o)$ `sometime_before` $at(5)$*: whenever the robot holds the object o, it must visit the location l_5 thereafter before reaching the goal;*
- `at_end` $[\bigwedge_{o \in O} \neg h(o)]$*: at the end, the robot should not hold any object.* □

With a slight abuse of notation, let us view a state s as a formula $\bigwedge_{s \models f} f \wedge \bigwedge_{s \models \neg f} \neg f$. Let S_i and S_f be two sets of states and

$$\Phi = \left[\underbrace{\bigvee_{s \in S_i} s}_{\Phi_1} \wedge \texttt{at\_end} \underbrace{[\bigvee_{s \in S_f} s]}_{\Phi_2} \right]$$

It is easy to see that any plan satisfying Φ requires its execution to start from a state satisfying Φ_1, which is one of the states in S_i, and end in a state satisfying Φ_2, which is one of the states in S_f. For this reason, the description of the initial and final states can be folded into a preference formula. We will therefore define planning problems as follows.

Definition 2. *Given a transition system T and a goal specification language $\mathcal{G} = (\mathcal{F}, \models_{\mathcal{G}})$ over T, a goal formula Φ in $\mathcal{F}$ is called a* planning problem*. A solution of Φ is a* plan *δ in T such that $\delta \models_{\mathcal{G}} \Phi$.*

By Def. 2, a goal formula represents a planning problem. The literature is quite diversified when a user faces two or more goal formulae which are contradictory with each other. For example, the formula $\Diamond at(2)$ is contradictory with $\Box \neg at(2)$; $\Box \neg (\bigwedge_{o \in O} h(o))$ conflicts with $\Diamond h(o_1)$; etc. A possibility is to consider a possible plan as solution if it satisfies some goal formulae. Another possibility is to rank the goal formulae and identify solutions as plans that satisfy the formula with the highest possible ranking. In the following, we will show that a uniform framework for dealing with conflicting goal formulae can be obtained by embedding goal formulae into Reiter's default logic.

4 Goal Default Theories

In this section, we will introduce a new goal specification language, called *goal default theory*. A goal default theory is a variation of Reiter's default theory [12], whose defaults can contain preference formulae. Goal default theories provide a possible treatment of planning with multiple goal formulae.

A goal default theory is defined over a transition system $T = (F, A, S, L)$ and a goal specification language $\mathcal{G} = (\mathcal{F}, \models_{\mathcal{G}})$.

Given a goal specification language $(\mathcal{F}, \models_{\mathcal{G}})$, we say that two formulae φ, ψ in $\mathcal{F}$ are equivalent w.r.t. $\models_{\mathcal{G}}$ if, for each plan δ of T, we have that $\delta \models_{\mathcal{G}} \varphi \Leftrightarrow \psi$.[3] We can easily extend this notion to define the notion of logical consequence w.r.t. $\models_{\mathcal{G}}$—if S is a set of formulae from $\mathcal{F}$ and f is another formula in $\mathcal{F}$, then $S \models_{\mathcal{G}} f$ if for each plan δ of T we have that $\delta \models_{\mathcal{G}} \bigwedge_{\varphi \in S} \varphi$ implies $\delta \models_{\mathcal{G}} f$. Given a set of formulae S, we define $Decl(S) = \{\varphi \mid \varphi \in \mathcal{F}, S \models_{\mathcal{G}} \varphi\}$.

A *preference default* (or *p-default*) d over $\mathcal{G}$ is of the following form

$$\frac{\alpha \,:\, \beta}{\gamma} \tag{3}$$

where α, β, and γ are formulae in $\mathcal{F}$. We call α the *precondition*, β the *justification*, and γ the *consequence* of d, and we denote them with $prec(d)$, $just(d)$, and $cons(d)$, respectively. A default d is said to be

- *Normal* if its justification is equivalent to its conclusion;
- *Prerequisite-free* if its precondition is equivalent to *true*; and
- *Supernormal* if it is normal and prerequisite-free.

Given a set of formulae S from $\mathcal{F}$, a default d is said to be defeated in S if $S \models \neg just(d)$. Some preferences and their representation as p-defaults over $\mathcal{G}_b$ for the domain from Example 1 are given next.

Example 3. *In these examples, o denotes a particular object in the domain.*

[3] $\varphi \Leftrightarrow \psi$ is a shorthand for $(\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$.

- *If there is no evidence that the robot is initially at the location l_2, then it should go to l_2:*

$$\frac{\top : \neg at(2)}{\Diamond at(2)} \tag{4}$$

- *Assume that objects might be defective, represented by the proposition $defective$. We can write*

$$\frac{\top \ : \ \Box[\neg \texttt{defective}(o)]}{\Box[at(2) \Rightarrow h(o)]} \tag{5}$$

to indicate that normally, we would like that the robot holds the object o whenever it is at the location l_2. An exception to this rule is possible if the object o is defective.

- *If the robot is not required to hold the object o in the final state and there is no evidence that it initially holds o, then it should not execute the action of picking up the object o:*

$$\frac{\top \quad : \quad \texttt{at\_end}\,(\neg h(o)) \wedge \neg h(o)}{\Box[\neg pickup(o)]} \tag{6}$$

- *If there is no evidence that the object o is initially in the wrong place then the robot should not start by executing the action of picking up the object o:*

$$\frac{\texttt{at\_end}\,(o\_at(o,i)) \ : \ \bigwedge_{i \neq j} \neg o\_at(o,j)}{\neg pickup(o)} \tag{7}$$

- *A stronger version of (7) is*

$$\frac{\texttt{at\_end}\,(o\_at(o,i)) \ : \ \bigwedge_{i \neq j} \neg o\_at(o,j)}{\Box \neg pickup(o)} \tag{8}$$

indicates that the robot should never pick up the object o if o could already be in the desired final location.

- *If there is the possibility that the robot might reach location l_2, then it must leave the location immediately after its arrival at l_2.*

$$\frac{\top \ : \ \Diamond[at(2)]}{\Box[at(2) \Rightarrow \bigcirc \bigvee_{i \neq 2} a_{2,i}]} \tag{9}$$

- *If there is no evidence that an object o will ever appear in location i then the robot should never go there.*

$$\frac{\top \ : \ \Box[\neg o\_at(o,i)]}{\Box[\bigvee_{j \neq i} \neg a_{j,i}]} \tag{10}$$

In the following, we will refer to the p-defaults in (4)-(9) by $p_1, \ldots, p_6$, respectively. □

We next define the notion of a goal default theory.

Definition 3. *A* goal default theory *over a goal language $\mathcal{G} = (\mathcal{F}, \models_{\mathcal{G}})$ and a transition system T is a pair $\Sigma = (D, W)$ where D is a set of p-defaults over $\mathcal{G}$ and $W \subseteq \mathcal{F}$.*

Given a set of p-defaults D, we denote with $cons(D)$ the set $cons(D) = \{cons(d) \mid d \in D\}$. A p-default d is *applicable* w.r.t. a set of $\mathcal{F}$ formulae S if $S \models_{\mathcal{G}} prec(d)$ and $S \not\models_{\mathcal{G}} \neg just(d)$. Let us denote with $\Pi_D(S)$ the set of p-defaults from D that are applicable w.r.t. S.

Definition 4 (From [12])**.** *Let $\Sigma = (D, W)$ be a goal default theory over $\mathcal{G} = (\mathcal{F}, \models_{\mathcal{G}})$ and T. An* extension *of Σ is a minimal set $E \subseteq \mathcal{F}$ that satisfies the condition $E = Decl(W \cup Cons(\Pi_D(E)))$. We say that Σ is* consistent *if it has at least one extension.*

From this definition, any default over the propositional language $F \cup A$ is a p-default, and any Reiter's default theory over the language $F \cup A$ is a goal default theory.

Definition 5. *Given a transition system $T = (F, A, S, L)$ and a goal specification language $\mathcal{G} = (\mathcal{F}, \models_{\mathcal{G}})$ over T, a* planning problem *over T and $\mathcal{G}$ is a goal default theory $\Sigma = (D, W)$ over $\mathcal{G}$ and T.*

The notion of a solution to a planning problem is modified as follows.

Definition 6. *Given a transition system $T = (F, A, S, L)$, a goal specification language $\mathcal{G} = (\mathcal{F}, \models_{\mathcal{G}})$ over T, and a planning problem Σ over T and $\mathcal{G}$, a* solution *of Σ is a plan δ in T such that $\delta \models_{\mathcal{G}} E$ for some extension E of Σ.*

Some planning problems over the transition systems in Exp. 1 and the language $\mathcal{G}_b$ are given in the next example.

Example 4 (Continuation of Example 3)**.**

- *Let $\Sigma_1 = (\{p_1\}, \{at(1), \texttt{at\_end}\ at(5)\})$ where p_1 is the default (4). Intuitively, we have that Σ_1 identifies plans where the robot starts at location l_1, goes through the location l_2, and ends in location l_5.*
- *Let $\Sigma_2 = (\{p_6\}, \{at(1), \texttt{at\_end}\ at(5)\})$ where p_6 is the default (9). This identifies plans where the robot starts at location l_1, ends in location l_5, and either (*i*) never goes through the location l_2; or (*ii*) never stays in the location l_2 within two consecutive steps.* □

The planning problems in Example 4 are simple, in that they are specified by goal default theories whose set of defaults is a singleton. Let us consider a more complicated example. Assume that we have two temporal formulae Φ and Ψ such that there exists no plan that can satisfy both Φ and Ψ. In this case, the use of goal default theory as a goal formula comes in handy. Indeed, every solution of the planning problem expressed by the goal default theory

$$\Sigma_{\Phi,\Psi} = \left(\left\{ \frac{\top : \neg\Psi}{\Phi}, \frac{\top : \neg\Phi}{\Psi} \right\}, \emptyset \right) \tag{11}$$

satisfies either Φ or Ψ. The following result generalizes this observation.

Proposition 1. *Let $T = (F, A, S, L)$ be a transition system, $\mathcal{G} = (\mathcal{F}, \models_{\mathcal{G}})$ be a goal specification language, and $\Delta = \{\Phi_1, \ldots, \Phi_n\}$ be a set of preference formulae in $\mathcal{F}$. Furthermore, let*

$$\Sigma_\Delta = \left(\left\{ \ \frac{\top : \Psi}{\Psi} \quad \middle| \quad \Psi \in \Delta \ \right\}, \emptyset \right) \tag{12}$$

- *For every solution δ to the problem Σ_Δ there exists a maximal (w.r.t. $\subseteq$) set of preferences $\Delta_\delta \subseteq \Delta$ such that $\delta \models_{\mathcal{G}} \bigwedge_{\Psi \in \Delta_\delta} \Psi$;*
- *For every pair of solutions δ and δ' of Σ_Δ, either $\Delta_\delta = \Delta_{\delta'}$ or $\Delta_\delta \not\subseteq \Delta_{\delta'}$ and $\Delta_{\delta'} \not\subseteq \Delta_\delta$.*

5 Goals Default Theories with Priorities

Proposition 1 shows that goal default theories can be used to specify planning problems with multiple preferences which might not be consistent with each other. For instance, consider a traveler from New York to San Francisco who has two preferences: reach the destination as fast as possible (Φ_1) and spend the least amount of money (Φ_2). In general, these two preferences cannot be satisfied at the same time. In this case, it is more reasonable to assume that a plan satisfying one of the criteria is an acceptable solution. Thus, $\Sigma_{\{\Phi_1,\Phi_2\}}$ is a reasonable goal specification if the traveler is impartial about Φ_1 and Φ_2. On the other hand, if the traveler prefers Φ_1 over Φ_2 (or vice versa), we will need to change the goal specification or provide additional ways for the traveler to specify this priority. As it turns out, the literature is rich with approaches for adding priorities to default theories [7, 8] which can be easily adapted to goal default theories. We next define

goal default theories with priorities by adapting the work of [7] to goal default theories.

Let us start by introducing static priorities, encoded by a well-ordering relation $\prec$ among p-defaults—i.e., $\prec$ is transitive, irreflexive, and each set of elements admits the least element in the ordering. We denote with $\min_{\prec}(X)$ the least element of X with respect to $\prec$. We define goal default theory with priorities as follows.

Definition 7. *A* goal default theory with priorities *over a goal language $\mathcal{G} = (\mathcal{F}, \models_{\mathcal{G}})$ and a transition system T is a triple $(D, W, \prec)$ where D is a set of p-defaults over $\mathcal{G}$, $\prec$ is a well-ordering relation over D, and $W \subseteq \mathcal{F}$.*

Following the general design of prioritizing default theory [7], the notion of preferred extension can be defined by successively simplifying the structure of the defaults.

Let us identify a construction of preferred extension through the application of defaults according to the ordering imposed by $\prec$. Let us introduce the $\mathcal{PR}$ operator which computes the next "preferred" set of goal formulae from an existing one:

- $\mathcal{PR}_{\prec}(S) = Decl(S \cup \{cons(d)\})$
  if $\Pi^*_D(S) \neq \emptyset \wedge d = \min_{\prec}(\{x \mid x \in \Pi^*_D(S)\})$;
- $\mathcal{PR}_{\prec}(S) = S$ if $\Pi^*_D(S) = \emptyset$

where $\Pi^*_D(S) = \{d \mid d \in \Pi_D(S), S \not\models cons(d)\}$. If the elements in D (for a goal default theory (D, W)) are supernormal, then it is possible to use $\mathcal{PR}_{\prec}$ to produce a monotone sequence of goal formulae, by setting $S_0 = Decl(W)$, $S_{i+1} = \mathcal{PR}_{\prec}(S_i)$ for any successor ordinal $i+1$ and $S_i = Decl(\bigcup_{j \leq i} S_j)$ for any limit ordinal i. We will denote the result of this construction as $Pref_{\prec}(D, W) = \bigcup_{i \geq 0} S_i$.

The process of determining a preferred extension will apply $Pref_{\prec}$ on a reduced version of the theory, in a style similar to that used in the Gelfond-Lifschitz reduct. Following the model proposed in [7], the reduct of a goal default theory with priorities $(D, W, \prec)$ w.r.t. a set of goal formulae S, denoted $(D^S, W, \prec^S)$, is obtained as follows:

- Determine $D' = \{\frac{\top \,:\, just(d)}{cons(d)} \mid d \in D, S \models_{\mathcal{G}} prec(d)\}$
- Determine $D^S = \{d \in D' \mid cons(d) \notin S \text{ or } S \not\models_{\mathcal{G}} \neg just(d)\}$ and $\prec^S$ is such that $d'_1 \prec^S d'_2$ if $d_1 \prec d_2$ and d_1 (d_2) is the $\prec$-least element that introduced d'_1 (d'_2) in D'.

We define preferred extensions as follows.

Definition 8. *Let $(D, W, \prec)$ be a goal default theory with priorities over $\mathcal{G} = (\mathcal{F}, \models_{\mathcal{G}})$ and T. A* preferred extension *E of $(D, W, \prec)$ is a set of goal formulae in $\mathcal{F}$ such that E is an extension of (D, W) and $E = Pref_{\prec^E}(D^E, W)$.*

Similar to [7], we can generalize the above definitions and define (*i*) a goal default theory with priorities as a triple $(D, W, \prec)$ where (D, W) is a goal default theory and $\prec$ is a partial order among defaults in D; and (*ii*) a set of formulae E is a preferred extension of $(D, W, \prec)$ if it is a preferred extension of some $(D, W, \prec_E)$ for some well-ordering $\prec_E$ which is an extension of $\prec$. For brevity, we omit the precise definitions. Definitions 5 and 6 can be extended in the obvious way: a planning problem is a goal default theory with priorities $(D, W, \prec)$ and its solutions are preferred extensions of $(D, W, \prec)$.

Example 5. *Let us consider the domain in Example 1. Let us assume that, among the objects, there is a very valuable object o_1 and a dangerous object o_2. Furthermore, let us assume that the robot is equipped with actions that can detect the object o_2 whenever the robot is at the same location as o_2. However, the equipment might not be working. We will denote with* working *the fact that the equipment is working properly. Let us consider the two formulae:*

- $\varphi := \Diamond h(o_1)$*: the robot should try to get the object o_1*
- $\psi := \Box[\bigwedge_{i \in \{1,\ldots,k\}}(o\_at(o_2, i) \Rightarrow \neg at(i))]$*: the robot should not be at the same place with object o_2 at any time.*

With these formulae, we can define the following p-defaults:

$$g_1 \equiv \frac{\top \,:\, working}{\psi \wedge \varphi} \qquad g_2 \equiv \frac{\top \,:\, \neg working}{\varphi}$$

g_1 indicates that if the equipment is initially working, then the robot will get o_1 while trying to avoid o_2. g_2 states that if the equipment is not working, then the robot will only worry about getting o_1. The theory $(\{g_1, g_2\}, \emptyset, \{g_1 \prec g_2\})$ states that we prefer that the robot tries to satisfy g_1 before trying to satisfy g_2.

6 Related Work and Discussion

In this section, we relate goal default theories with priorities to existing goal specification languages. We then discuss possible applications of the new language.

- **TEP formulae**: TEP formulae have been implemented in a planner in [1]. Given a set of TEP formulae $\Delta = \{\Phi_1, \ldots, \Phi_n\}$, a planning problem is an optimization problem that maximizes the rewards obtained by satisfying the formulae in Δ. Formally, the reward over a plan δ is

  $$\Sigma_{\Phi_i \in \Delta, \delta \models \Phi_i} reward(\Phi_i) - \Sigma_{\Phi_i \in \Delta, \delta \not\models \Phi_i} penalty(\Phi_i)$$

  where $reward(\Phi)$ and $penalty(\Phi)$ denote the reward and penalty for satisfying and not satisfying Φ, respectively.
  The planning problem can be expressed by a goal default theory with priorities as follows. Let S be a set of formulae, $S \subseteq \Delta$, and d_S be the default

  $$\frac{\top \,:\, \bigwedge_{\Phi \in S} \Phi \wedge \bigwedge_{\Phi \in \Delta \setminus S} \neg \Phi}{\bigwedge_{\Phi \in S} \Phi \wedge \bigwedge_{\Phi \in \Delta \setminus S} \neg \Phi}$$

  Let $D_\Delta = \{d_S \mid S \subseteq \Delta\}$ and $\prec_\Delta$ be the partial order over D_Δ where $d_S \prec_\Delta d_{S'}$ if

  $$\Sigma_{\Phi_i \in S} reward(\Phi_i) - \Sigma_{\Phi_i \notin S} penalty(\Phi_i) \geq \Sigma_{\Phi_i \in S'} reward(\Phi_i) - \Sigma_{\Phi_i \notin S'} penalty(\Phi_i).$$

  We can show that $(D_\Delta, \emptyset, \prec_\Delta)$ is a goal default theory with priorities representing the given planning problem, i.e., any preferred solution of $(D_\Delta, \emptyset, \prec_\Delta)$ is a solution of the original planning problem and vice versa.
- $\mathcal{PP}$: The language $\mathcal{PP}$ allows the specification of three types of preferences. A *basic desire* φ is a preference over a trajectory and therefore is a part of the basic goal language. An atomic preference is an ordering among basic desires $\Phi = \Phi_1 \lhd \Phi_2 \ldots \lhd \Phi_k$ and expresses that the preference Φ_i is more important than Φ_{i+1} for $1 \leq i < k-1$. An *atomic preference* Φ can be represented by the following goal default theory with priorities

  $$\left(\left\{ \frac{\top \,:\, \Phi_i}{\Phi_i} \quad \middle| \quad i = 1, \ldots, k \right\}, \emptyset, \prec_\Phi\right)$$

  where $\prec_\Phi$ is defined by $\frac{\top \,:\, \Phi_i}{\Phi_i} \prec_\Phi \frac{\top \,:\, \Phi_j}{\Phi_j}$ for $1 \leq i < j \leq k$.
  A general preference is either an atomic preference or a combination of general preferences, such as $\Phi \& \Psi$, $\Phi | \Psi$, and $!\Phi$, where Φ and Ψ are general preferences. Intuitively, general preferences add finitely many levels to the specification of preferences and

thus cannot be easily represented by goal default theories which assume ceteris paribus over the preferences. Adding priorities allows only an extra layer of comparison between preferences. We view this as a weakness of goal default theories and plan to further investigate this issue.

- **N-LTL** and **ER-LTL**: These two languages allow the specification of weak and strong exceptions within goal formulae represented as LTL-formulae by introducing labels to LTL-formulae. By compiling away the labels as in [4], we can show that $\mathcal{G}_b$ subsumes N-LTL and ER-LTL.
  Observe that the constructs used in N-LTL and ER-LTL are fairly close to default logic. This leads us to believe that interesting collections of N-LTL (ER-LTL) theories can be translated into goal default theories—which would provide a reasonable semantics for N-LTL (ER-LTL) theories with loops that have not been considered so far.
  Finally, we would like to note that $\mathcal{G}_b$ can be easily extended to consider N-LTL (ER-LTL) formulae by
  - extending $\mathcal{F}_b$ with N-LTL (ER-LTL) formulae; and
  - extending $\models_{\mathcal{G}_b}$ to define that $\delta \models_{\mathcal{G}_b} S$ iff $\delta \models_{\mathcal{G}_b} c(S)$ where $c(S)$, a LTL formula, denotes the result of compiling S to an LTL formula as described in [4, 5].
- π-CTL* and P-CTL*: These two languages consider non-deterministic domains and define goals over policies but do not consider preferences among goals. In addition, these languages introduce the operators A, E, A_π, and E_π over paths and the two quantifiers $\mathcal{EP}$ and $\mathcal{AP}$ over state formulae. Nevertheless, we can show that the CTL* part of π-CTL* can be expressed in $\mathcal{G}_b$. Furthermore, $\mathcal{G}_b$ can be extended to allow formulae of π-CTL*. However, the two new state quantifiers are not expressible in our goal language. We observe that as the goal language is parameterized with the satisfaction relation, $\mathcal{G}_b$ can be easily extended with these operators. We strongly believe that these extensions will be sufficient for goal default theories with priorities to capture P-CTL*.

The above discussion highlights features from existing goal languages that can (or cannot) be expressed by our goal language. This also shows that the proposed language can serve as a unified language for evaluating goal languages. The use of default theories as the basic language also provides us with an advantage in the study of computational complexity of goal languages. In this effort, we expect that well-known complexity results on prioritized default theories [13] will be extremely useful. This will provide us with insights for the use of existing goal languages as well as the development of new goal languages.

7 Conclusions and Future Work

In this paper, we describe a default logic based approach to defining non-monotonic goal specification languages. We start with a basic goal specification language and use default logic (or prioritizing default logic) to provide a natural way for dealing with inconsistency and priorities over goals. We show that the new language subsumes some goal languages in the literature and can describe several features from other goal languages. We identify desirable features that cannot be easily expressed by our goal language, among them is the multi-level of preferences between goals, which we intend to investigate in the near future. We also discuss possible applications of the proposed goal language.

Acknowledgments: The first two authors were partially supported by the NSF grants IIS-0812267.

REFERENCES

[1] Jorge A. Baier, Fahiem Bacchus, and Sheila A. McIlraith, 'A heuristic search approach to planning with temporally extended preferences', *Artif. Intell.*, **173**(5-6), 593–618, (2009).
[2] Chitta Baral and Jicheng Zhao, 'Goal specification in presence of non-deterministic actions', in *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, eds., Ramon López de Mántaras and Lorenza Saitta, pp. 273–277. IOS Press, (2004).
[3] Chitta Baral and Jicheng Zhao, 'Goal specification, non-determinism and quantifying over policies', in *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*. AAAI Press, (2006).
[4] Chitta Baral and Jicheng Zhao, 'Non-monotonic temporal logics for goal specification', in *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, ed., Manuela M. Veloso, pp. 236–242, (2007).
[5] Chitta Baral and Jicheng Zhao, 'Non-monotonic temporal logics that facilitate elaboration tolerant revision of goals', in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, eds., Dieter Fox and Carla P. Gomes, pp. 406–411. AAAI Press, (2008).
[6] M. Bienvenu, C. Fritz, and S. McIlraith, 'Planning with qualitative temporal preferences', in *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR06)*, pp. 134–144, Lake District, UK, (June 2006).
[7] G. Brewka and T. Eiter, 'Prioritizing default logic', in *Intellectics and Computational Logic*, volume 19 of *Applied Logic Series*, 27–45, Kluwer, (2000).
[8] James Delgrande and Torsten Schaub, 'Expressing preferences in default logic', *Artificial Intelligence*, **123**, 41–87, (2000).
[9] E. A. Emerson, 'Temporal and modal logic', in *Handbook of theoretical computer science: volume B*, ed., J. van Leeuwen, 995–1072, MIT Press, (1990).
[10] M. Fox and D. Long, 'PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains', *Journal of Artificial Intelligence Research*, **20**, 61–124, (2003).
[11] A. Pnueli, 'The temporal logic of programs', in *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 46–57, (1977).
[12] R. Reiter, 'A logic for default reasoning', *Artificial Intelligence*, **13**(1,2), 81–132, (1980).
[13] Jussi Rintanen, 'Complexity of prioritized default logics', *Journal of Artificial Intelligence Research*, **9**, 423–461, (1998).
[14] Tran Cao Son and Enrico Pontelli, 'Planning with Preferences using Logic Programming', *Theory and Practice of Logic Programming*, **6**, 559–607, (2006).

Optimization and Elicitation with the Maximin Utility Criterion

Paolo Viappiani[1] and **Christian Kroer**[2]

Abstract. We investigate robust decision-making under utility uncertainty, using the *maximin* criterion, which optimizes utility for the worst case setting. We show how it is possible to efficiently compute the maximin optimal recommendation in face of utility uncertainty, even in large configuration spaces. We then introduce a new decision criterion, *setwise maximin utility (SMMU)*, for constructing optimal recommendation sets: we develop algorithms for computing SMMU, and prove (analogously to previous results related to regret-based and Bayesian elicitation) that SMMU determines choice sets for queries that are myopically optimal. We also present experimental results showing performance of SMMU on randomly generated elicitation problems.

1 Introduction

Learning the preferences of the user [10] is an important problem in many domains, including decision support and recommender systems, personal agents and cognitive assistants. Because acquiring user preferences is expensive (with respect to time and cognitive cost), it is essential to provide techniques that can reason with partial preference (utility) information, and that can effectively elicit the most relevant preference information.

Following recent works in AI, we cast decision-making and elicitation as a problem of optimization under uncertainty. Adaptive utility elicitation [5] tackles the challenges posed by preference elicitation by representing the system knowledge about the user in form of *beliefs*, that are updated following user responses. Elicitation queries can be chosen adaptively given the current belief. In this way, one can often make good (or even optimal) recommendations with sparse knowledge of the user's utility function.

In this paper, we investigate the problem of producing robust recommendations using the *maximin* criterion. Maximin is the most pessimistic decision criterion; the recommended decision or option is the one associated with the highest utility in the worst case.

We examine the *strict* uncertainty setting: all we are given is a set of constraints that encode the possible utility functions (usually obtained through some form of user feedback, such as responses to elicitation queries of the type: *"Which of these products do you prefer ?"*). We argue that maximin can be adopted as a suitable robust decision criterion for decision making in the presence of such utility function uncertainty. Furthermore, we extend this idea to sets, defining the *setwise maximin* utility criterion, and we discuss the problem of interactive elicitation (which can be viewed as active preference learning). Finally, we show how linear and mixed integer programming techniques can be used to efficiently optimize both singleton recommendations and sets in large configuration spaces.

[1] Aalborg University, Denmark, email: paolo@cs.aau.dk
[2] Carnegie Mellon University, USA, email: ckroer@cs.cmu.edu

1.1 Assumptions

We assume a recommendation system is charged with the task of recommending an option to a user in some multiattribute space, for instance, the space of possible product configurations from some domain (e.g., computers, cars, apartment rental, etc.). Products are characterized by a finite set of attributes $\mathcal{X} = \{X_1, ...X_n\}$, each with finite domains $Dom(X_i)$. Let $\mathbf{X} \subseteq Dom(\mathcal{X})$ denote the set of *feasible configurations*. For instance, attributes may correspond to the features of various apartments, such as size, neighborhood, distance from public transportation, etc., with $\mathbf{X}$ defined either by constraints on attribute combinations (e.g., constraints on computer components that can be put together), or by an explicit database of feasible configurations (e.g., a rental database).

The user has a *utility function* $u : Dom(\mathcal{X}) \rightarrow \mathbf{R}$. In what follows we will assume either a *linear* or *additive* utility function depending on the nature of the attributes [8]. In both additive and linear models, u can be decomposed as follows[3]:

$$u(\mathbf{x}) = \sum_i f_i(x_i) = \sum_i \lambda_i v_i(x_i)$$

where each *local* utility function f_i assigns a value to each element of $Dom(X_i)$. In classical utility elicitation, these values can be determined by assessing local value functions v_i over $Dom(X_i)$ that are normalized on the interval $[0, 1]$, and importance weights λ_i ($\sum_i \lambda_i = 1$) for each attribute [7, 8]. This sets $f_i(x_i) = \lambda_i v_i(x_i)$ and ensures that global utility is normalized on the interval $[0, 1]$. A simple additive model in the rental domain might be:

$$u(Apt) = f_1(Size) + f_2(Distance) + f_3(Nbrhd)$$

When $Dom(X_i)$ is drawn from some real-valued set, we often assume that v_i (hence f_i) is linear in X_i.[4]

We note that our framework subsumes the case of "unfactored" utilities (the utility of an option is an unknown latent value that does not factor into attributes or features); this case can be modeled by considering a parameter to represent the utility of the option.

$$u(x^i; w) = w_i \tag{1}$$

Vector $\mathbf{w} = (w_1, ..., w_n)$ is then composed of the utilities for each option. Prior knowledge can provide lower bounds and upper bounds

[3] In our notation, we use bold lowercase for vectors
[4] Our presentation relies heavily on the additive assumption, though our approach is easily generalized to more general models such as GAI [7, 4]. The assumption of linearity is simply a convenience; nothing critical depends on it.

for $w_1, ..., w_n$. W is then a "hyper rectangular" region of possible utility values. Unfactored models are of limited applicability. One main drawback is that we need one utility parameter for each available option. The advantages of a factored (multi-attribute) utility representation is that preference statements, such as responses to comparison queries between two options $\mathbf{x}$ and $\mathbf{y}$, can "generalize" to other options, that have some features in common.

Since a user's utility function is not generally known, we write $u(\mathbf{x}; w)$ to emphasize the dependence of u on user-specific parameters. In the additive case, the values $f_i(x_i)$ over $\cup_i\{Dom(X_i)\}$ serve as a sufficient parameterization of u (for linear attributes, a more succinct representation is possible). The optimal product for the user with utility parameters w is $argmax_{\mathbf{x}\in\mathbf{X}} u(\mathbf{x}; w)$. Our goal is to recommend, or help the user find, an optimal, or near optimal, product.

2 Decision-making with Maximin Utility

Much work in AI, decision analysis and operations research has been devoted to effective elicitation of preferences [13, 2, 6, 1, 14]. Adaptive preference elicitation generally differs from classical utility assessment in that it recognizes that good, even optimal, decisions can often be recommended with very sparse knowledge of a user's utility function [2]; and that the value of information associated with specific elicitation actions (e.g., queries)—in terms of its impact on decision quality—is often not worth the cost of obtaining it [6, 1]. This means we must often take decisions in the face of an incompletely specified utility function.

In this work, we adopt the notion of *maximin utility* as our decision criterion for robust decision making under utility function uncertainty.

Assume that through some interaction with a user, and possibly using some prior knowledge, we determine that her utility function w lies in some set W. (The form of W will become clearer when we discuss elicitation below). We define:

Definition 1 *Given a set of feasible utility functions W, the* min utility (MU) $MU(\mathbf{x}; W)$ *of $\mathbf{x} \in \mathbf{X}$ is defined as:*

$$MU(\mathbf{x}; W) = \min_{w\in W} u(\mathbf{x}; w)$$

Definition 2 *The* maximin utility $MMU(W)$ *of W and the corresponding* minimax optimal configuration $\mathbf{x}^*_W$ *are defined as follows:*

$$\begin{aligned} MMU(W) &= \max_{\mathbf{x}\in\mathbf{X}} MU(\mathbf{x}; W) = \max_{\mathbf{x}\in\mathbf{X}} \min_{w\in W} u(\mathbf{x}; w) \\ \mathbf{x}^*_W &= \arg\max_{\mathbf{x}\in\mathbf{X}} MU(\mathbf{x}; W) = \arg\max_{\mathbf{x}\in\mathbf{X}} \min_{w\in W} u(\mathbf{x}; w) \end{aligned}$$

Intuitively, $MU(\mathbf{x}; W)$ is the worst-case utility associated with recommending configuration $\mathbf{x}$; i.e., by assuming an adversary will choose the user's utility function $\mathbf{w}$ from W to minimize the utility. The maximin optimal configuration $\mathbf{x}^*_W$ is the configuration that maximizes this minimum utility. Any choice that is not maximin optimal has strictly lower utility than $\mathbf{x}^*_W$ for some $\mathbf{w} \in W$.

Maximin utility (as does minimax regret [15]) relies on relatively simple prior information in the form of bounds or constraints on user preferences (rather than probabilistic priors); and exact computation is much more tractable (in contrast with probabilistic models of utility that generally require reasoning with densities that have no closed form [1, 6]). In configuration problems, optimization over product space $\mathbf{X}$ is often formulated as a CSP or mixed integer program (MIP). In such domains, maximin utility computation can be formulated as a MIP, and solved practically for large problems using techniques such as Bender's decomposition and constraint generation [2, 4].

3 Optimal Recommendation Sets

In general, there is a tension between recommending the best options to the user, and acquiring informative feedback from the user. Since utility is uncertain, there is often value in recommending a *set* of options from which the user can choose her most preferred. Picking a "diverse" set of recommended options increases the odds of recommending at least one item with high utility. Intuitively, such a set of "shortlisted" recommendations should include options that are *diverse* in the following sense: recommended options should be highly preferred relative to a wide range of "likely" user utility functions (relative to the current belief) [11, 3]. This stands in contrast to some recommender systems that define diversity relative to product attributes [12], with no direct reference to beliefs about user utility. It is not hard to see that "top k" systems, those that present the k options with highest expected utility, do not generally result in good recommendation sets [11].

Among the many possible types of queries, we focus on *choice queries*. Such queries are commonly used in conjoint analysis and product design [9], requiring a user to indicate which choice/product is most preferred from a set of k options. Hence, we can view any set of products as either a recommendation set or query (or choice) set. Given a set, one can ask: what is the value of the set viewed as recommendation set; or what is its value as a query?

Recently, Viappiani and Boutilier [16, 15] showed how these two problems are connected to each other, under both a Bayesian framework or when one assumes *minimax regret* as a criterion. In the following we show the same connection when minimax utility is used as the decision criterion.

3.1 Setwise Maximin Utility

Suppose we have a slate of k options to present to the user and want to quantify the minimum utility obtained by restricting the user's decision to options in that slate. Intuitively, the user may select any of the k options as being "optimal." An adversary wanting to minimize utility should do so assuming that any such choice is possible, as we allow the user to pick *any* of the k options. Formally, we choose the set of k options first, but delay the specific choice from the slate until *after* the adversary has chosen a utility function $\mathbf{w}$. The maximin utility is the utility of the best option w.r.t. $\mathbf{w}$ in the slate. (To keep notation to a minimum, we assume $\mathbf{Z}$ is restricted to suitable subsets of $\mathbf{X}$ (e.g., of cardinality k) without making this explicit.)

Definition 3 *Let W be a feasible utility set, $\mathbf{Z} \subseteq \mathbf{X}$. Define:*

$$\begin{aligned} SMU(\mathbf{Z}, W) &= \min_{w\in W} \max_{\mathbf{x}\in\mathbf{Z}} u(\mathbf{x}; w) \\ SMMU(W) &= \max_{\mathbf{Z}\subseteq\mathbf{X}} \min_{w\in W} \max_{\mathbf{x}\in\mathbf{Z}} u(\mathbf{x}; w) \\ \mathbf{Z}^*_W &= \arg\max_{\mathbf{Z}\subseteq\mathbf{X}} \min_{w\in W} \max_{\mathbf{x}\in\mathbf{Z}} u(\mathbf{x}; w) \end{aligned}$$

The *setwise minimum utility(SMU)* of a set $\mathbf{Z}$ of k options reflects the intuitions above. *Setwise maximin utility (SMMU)* is SMU of the minimax optimal set $\mathbf{Z}^*_W$, i.e., the set that maximizes $SMU(\mathbf{Z}, W)$.

Setwise maximin utility has some intuitive properties. First, adding new items to a recommendation set cannot decrease SMU:

Observation 1 $SMU(\mathbf{A} \cup \mathbf{B}, W) \geq SMU(\mathbf{A}, W)$.

Incorporating options that are known to be dominated given W does not change setwise maximin utility:

Observation 2 *If $u(\mathbf{a}, w) > u(\mathbf{b}, w)$ for some $\mathbf{a} \in \mathbf{Z}$ and all $\mathbf{w} \in W$, then $SMU(\mathbf{Z} \cup \{\mathbf{b}\}, W) = SMU(\mathbf{Z}, W)$.*

Observation 3 *MU and SMU can be explicitly expressed as the minimization over different utility spaces*

$$MU(\mathbf{A}; W_1 \cup W_2) = \min\{MU(\mathbf{A}; W_1), MU(\mathbf{A}; W_2)\}$$
$$SMU(\mathbf{A}; W_1 \cup W_2) = \min\{SMU(\mathbf{A}; W_1), SMU(\mathbf{A}; W_2)\}$$

The choice of $x \in \mathbf{Z}$ for SMU is dictated by which x has the highest utility with respect to the chosen $\mathbf{w} \in W$. Due to this, the different choices of $x \in \mathbf{Z}$ define a partition of the utility space, where a partition with respect to a given x is the region of W where the utility of x is higher than any other option in $\mathbf{Z}$. We make this partition explicit:

$$W[\mathbf{Z} \to \mathbf{x}_i] = \{\mathbf{w} \in W : u(\mathbf{x}_i; w) > u(\mathbf{x}_j; w)\ \forall j \neq i, 1 \leq j \leq k\}$$

(i.e., the region of $\mathbf{w}$ where $\mathbf{x}_i$ has greater utility than any other option in $\mathbf{Z}$). The regions $W[\mathbf{Z} \to \mathbf{x}_i]$, $\mathbf{x}_i \in \mathbf{Z}$, partition W (we ignore ties over full-dimensional subsets of W, which are easily dealt with, but complicate the presentation). We call this the $\mathbf{Z}$-*partition of* W. Using the $\mathbf{Z}$-partition, we can rewrite SMU:

Observation 4 *Let $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$. Then*

$$\begin{aligned} SMU(\mathbf{Z}, W) &= \min_{\mathbf{x} \in \mathbf{Z}} \min_{w \in W[\mathbf{Z} \to \mathbf{x}]} u(\mathbf{x}, w) \\ &= \min_{i=1 \leq \ldots \leq k} MU(\mathbf{x}_i, W[\mathbf{Z} \to \mathbf{x}_i]) \end{aligned}$$

We use a similar notation to express the combination of two partitions: $W[\mathbf{Z}_1 \to \mathbf{x}_i, \mathbf{Z}_2 \to \mathbf{x}_j] = W[\mathbf{Z}_1 \to \mathbf{x}_i] \cap W[\mathbf{Z}_2 \to \mathbf{x}_j]$. Using this notation, we observe the following inequality for all i, j: (the proof is straightforward from the definition)

Observation 5 *For all $i, j \in \{1 \ldots k\}$:*

$$MU(\mathbf{x}_i, W[\mathbf{Z} \to \mathbf{x}_i, \mathbf{Z} \to \mathbf{x}_j] \geq MU(\mathbf{x}_i, W[\mathbf{Z} \to \mathbf{x}_i])$$

3.2 Optimal Myopic Elicitation

Usually, utility information is not readily available, but must be acquired through an elicitation process. Since elicitation can be costly, it is important to ask queries that elicit the most information. Our setwise maximin utility criterion can be used directly for this purpose, implementing a form of preference-based diversity. This stands in contrast to "product diversity" typically considered in recommender systems based on critiquing. And unlike recent work in polyhedral conjoint analysis [14], which emphasizes volume reduction of the utility polytope W, our maximin utility-based criterion is sensitive to the range of feasible products and does not reduce utility uncertainty for its own sake.

Any set $\mathbf{Z}$ can be interpreted as a query (or system-generated dynamic compound critique): We simply allow the user to state which of the k elements $\mathbf{x}_i \in \mathbf{Z}$ she prefers. We refer to $\mathbf{Z}$ interchangeably as a *query* or a *choice set*. The choice of some $\mathbf{x}_i \in \mathbf{Z}$ refines the set of feasible utility functions W by imposing the $k-1$ linear constraints $u(\mathbf{x}_i; \mathbf{w}) > u(\mathbf{x}_j; \mathbf{w}), j \neq i$.

When treating $\mathbf{Z}$ as a choice set (as opposed to a recommendation set), we are not interested in its maximin utility, but rather in *how much a query response will reduce maximin utility*. In our distribution-free setting, the most appropriate measure is *posterior maximin utility*, a measure of the value of information of a query. Generalizing the pairwise measure of [2], we define:

Definition 4 *The* worst case posterior maximin utility (WP) *of $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ is*

$$WP(\mathbf{Z}, W) = \min[MMU(W[\mathbf{Z} \to \mathbf{x}_1]), \ldots, MMU(W[\mathbf{Z} \to \mathbf{x}_k])]$$

which can be rewritten as:

$$WP(\mathbf{Z}, W) = \min_{\mathbf{x} \in \mathbf{Z}} \max_{\mathbf{x}' \in \mathbf{X}} \min_{\mathbf{w} \in W[\mathbf{Z} \to \mathbf{x}]} u(\mathbf{x}', \mathbf{w})$$

An optimal choice set $OptQuery(W)$ *is any $\mathbf{Z}$ that maximizes worst case posterior maximin utility $MaxWP(W)$:*

$$MaxWP(W) = \max_{\mathbf{Z} \subseteq \mathbf{X}} WP(\mathbf{Z}, W)$$

Intuitively, each possible response $\mathbf{x}_i$ to the query $\mathbf{Z}$ gives rise to updated beliefs about the user's utility function. We use the worst-case response to measure the quality of the query (i.e., the response that leads to the updated W with lowest maximin utility). The optimal query is that which maximizes this value. We observe:

Observation 6 $WP(\mathbf{Z}, W) \geq SMU(\mathbf{Z}, W)$.

Proof If we consider the definition of $WP(\mathbf{Z}, W)$ and the equation for $SMU(\mathbf{Z}, W)$ in observation 4, we see that they are the same except that $WP(\mathbf{Z}, W)$ picks a maximizing $\mathbf{x}' \in \mathbf{X}$ after $\mathbf{x} \in \mathbf{Z}$ has been picked. Since $\mathbf{X}$ includes all options, $\mathbf{x}'$ can at worst be equal to $\mathbf{x}$. ■

Using this fact, we introduce a transformation that modifies a given recommendation set $\mathbf{Z}$ in such a way that SMU cannot decrease and usually increases. This will be used both for proving the optimality of SMU as a choice set, and as a heuristic for efficiently generating choice sets. Define the transformation T to be a mapping that updates a given recommendation set $\mathbf{Z}$ in the following way: (a) First we construct the $\mathbf{Z}$ partition of W; (b) we then compute the *single recommendation* that has maximin utility in each region of the partition of W; (c) finally, we let $T(\mathbf{Z})$ be the new recommendation set consisting of these new recommendations.

Definition 5 *Let $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$. We define*

$$T(\mathbf{Z}) = \{\mathbf{x}^*_{W[\mathbf{Z} \to \mathbf{x}_1]}, \ldots \mathbf{x}^*_{W[\mathbf{Z} \to \mathbf{x}_k]}\}$$

Using Observation 3 and Observation 4, we prove the following.

Observation 7 *Let $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$. Let be $W^1, ..., W^l$ be any partition of W.*

$$\begin{aligned} WP(\mathbf{Z}, W) &= \min_i MMU(W[\mathbf{Z} \to \mathbf{x}_i]) \\ &= \min_i MU(\mathbf{x}^*_{W[\mathbf{Z} \to \mathbf{x}_i]}, W[\mathbf{Z} \to \mathbf{x}_i]) \\ &= \min_{i,j}\{MU(\mathbf{x}^*_{W[\mathbf{Z} \to \mathbf{x}_i]}, W[\mathbf{Z} \to \mathbf{x}_i] \cap W^j)\} \end{aligned}$$

In particular, if we consider $T(\mathbf{Z}) = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_k\}$ *where* $\mathbf{x}'_i = \mathbf{x}^*_{W[\mathbf{Z}\rightarrow\mathbf{x}_i]}$ *and its induced partition on* W*, the exapression above become the following.*

$$WP(\mathbf{Z}, W) = \min_{i,j}\{MU(\mathbf{x}^*_{W[\mathbf{Z}\rightarrow\mathbf{x}_i]}, W[\mathbf{Z} \rightarrow \mathbf{x}_i; T(\mathbf{Z}) \rightarrow \mathbf{x}'_i]\}$$

Using this, we can now prove the following lemma:

Lemma 1 $SMU(T(\mathbf{Z}), W) \geq WP(\mathbf{Z}, W)$

Proof Let $T(\mathbf{Z}) = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_k\}$ where $\mathbf{x}'_i = \mathbf{x}^*_{W[\mathbf{Z}\rightarrow\mathbf{x}_i]}$. The previous observations allow to write WP and SMU compactly

$$WP(\mathbf{Z}, W) = \min_{i,j}[MU(\mathbf{x}'_i, W[\mathbf{Z} \rightarrow \mathbf{x}_i, T(\mathbf{Z}) \rightarrow \mathbf{x}'_j])] \quad (2)$$

$$SMU(T(\mathbf{Z}), W) = \min_{i,j}[MU(\mathbf{x}'_j, W[\mathbf{Z} \rightarrow \mathbf{x}_i, T(\mathbf{Z}) \rightarrow \mathbf{x}'_j])] \quad (3)$$

We now compare the two expressions componentwise. Consider the utility space $W[\mathbf{Z} \rightarrow \mathbf{x}_i, T(\mathbf{Z}) \rightarrow \mathbf{x}'_j]$: if $i = j$ then the two MU components are the same. If $i \neq j$, consider any $w \in W[\mathbf{Z} \rightarrow \mathbf{x}_i, T(\mathbf{Z}) \rightarrow \mathbf{x}'_j]$. Since $w \in W[T(\mathbf{Z}) \rightarrow \mathbf{x}'_j]$, we must have $u(\mathbf{x}'_j; w) > u(\mathbf{x}'_i; w)$. Therefore $MU(\mathbf{x}'_j, W[\mathbf{Z} \rightarrow \mathbf{x}_i, T(\mathbf{Z}) \rightarrow \mathbf{x}'_j]) \geq MU(\mathbf{x}'_i, W[\mathbf{Z} \rightarrow \mathbf{x}_i, T(\mathbf{Z}) \rightarrow \mathbf{x}'_j])$. In the expression of $SMU(T(\mathbf{Z}))$ (Eq. 3), each element is no less than its correspondent in the $WP(\mathbf{Z})$ expression (Eq. 2). Thus $SMU(T(\mathbf{Z}), W) \geq WP(\mathbf{Z}, W)$. ∎

From observation 6 and lemma 1 it follows that $SMU(T(\mathbf{Z}), W) \geq SMU(\mathbf{Z}, W)$.

Theorem 1 *Let* $\mathbf{Z}^*_W$ *be a maximin optimal recommendation set. Then* $\mathbf{Z}^*_W$ *is an optimal choice set:* $WP(\mathbf{Z}^*_W, W) = MaxWP(W)$.

Proof Suppose $\mathbf{Z}^*_W$ is not an optimal choice set, i.e., there is some $\mathbf{Z}'$ such that $WP(\mathbf{Z}', W) > WP(\mathbf{Z}^*_W, W)$. If we apply transformation T to $\mathbf{Z}'$ we obtain a set $T(\mathbf{Z}')$, and by the results above we have: $SMU(T(\mathbf{Z}', W)) \geq WP(\mathbf{Z}', W) > WP(\mathbf{Z}^*, W) \geq SMR(\mathbf{Z}^*_W, W)$. This contradicts the (setwise) maximin optimality of $\mathbf{Z}^*_W$. ∎

4 Maximin Utility Optimization

In this section we formalize the problem of generating recommendations (both single recommendations and setwise recommendations) using mathematical programming techniques (linear programming models and mixed integer programming models).

In the following we assume the utility to be linear in w: $u(\mathbf{x}; w) = w \cdot \mathbf{x}$. In this case W is convex polytope effectively represented by a set of constraints. Whenever the user answer a query, new constraints are added. We denote with *Constraints(W)* the set of constraints that represent the space of feasible utility functions (consistent with the user's answers).

MU(x, W) Given a configuration $\mathbf{x}$ and a space of possible utility functions W (encoded by linear constraints), the minimum utility of x can be found by solving the following linear problem ($w_i^\perp$ and $w_i^\top$ are a lower and upper bound on the values of the utility parameters w_i; this can be used to encode a non-probabilistic "prior" on the utility parameters).

$$\min \quad \mathbf{w} \cdot \mathbf{x} = \sum_{1 \leq i \leq n} x_i \cdot w_i$$

$$\text{s.t. } Constraints(W) \quad (4)$$

$$w_i^\perp \leq \mathbf{w}_i \leq w_i^\top \qquad \forall i \in \{1 \ldots n\} \quad (5)$$

Decision variables: $\mathbf{w}$ (vector of size n)

MMU(W) Given a space of possible utility functions W (encoded by linear constraints), the problem is to find the configuration $\mathbf{x}^*_W$ that is associated with maximin utility. In order to "break" the maximin optimization, we make use of Benders decomposition:

$$\max \quad \delta$$

$$\text{s.t. } \delta \leq \mathbf{w} \cdot \mathbf{x} \qquad \forall \mathbf{w} \in GEN \quad (6)$$

Decision variables: $\mathbf{x}$, δ

In this model, δ corresponds to the *maximin utility* of the optimal recommendation $\mathbf{x}^*_W$. Constraint 6 ensures that δ is less than the utility of choice $\mathbf{x}$ for each $\mathbf{w}$. The optimization is exact when $GEN = W$ in constraint 6. However, all the constraints over W need not be expressed for each of the (continuously many) $\mathbf{w} \in W$. Since maximin utility is optimal at some vertex of W, we only need to apply constraints for all vertices of W, which we denote $Vert(W)$. However, the number of vertices in W can still be potentially exponential. We apply constraint generation in order to make solving the MIP much more efficient, as very few of the vertices are usually needed. This procedure works by solving a relaxed version of the problem above—the *master problem*— using only the constraints corresponding to a small subset $GEN \subset Vert(W)$. We then test whether any constraints are violated in the current solution. This is accomplished by computing the *minimum utility* of the returned solution. If MU is lower than what was found in the master problem, a constraint was violated. The vertex for this constraint (corresponding to the choice w^a of the adversary) is added to the master problem, tightening the MIP relaxation. The new relaxation is computed, and this process is repeated until no violated constraints exist.

Now we provide LP and MIP formulations that extend these optimization to sets.

SMU(Z, W) Given a set Z and a space of possible utility functions W the setwise minimum utility of Z can be found by solving k (k being the cardinality of Z) optimization problems, in virtue of Observation 4. Considering the Z-partition of W, we compute $MU(\mathbf{x}, W[\mathbf{Z} \rightarrow \mathbf{x}])$ for each $\mathbf{x} \in \mathbf{Z}$, using the LP model shown above. We then take the (arithmetic) minimum of the results: $\min_{x \in \mathbf{Z}} MU(\mathbf{x}, W[\mathbf{Z} \rightarrow \mathbf{x}])$.

SMMU(W) Given utility space W, we can compute the maximin optimal set (of cardinality k) using the following MIP.

$$\max \quad \delta$$

$$\text{s.t. } \delta \leq \sum_{1 \leq j \leq k} v^j_w \qquad \forall \mathbf{w} \in GEN \quad (7)$$

$$v^j_w \leq \mathbf{w} \cdot \mathbf{x}^j \qquad \forall j \leq k, w \in GEN \quad (8)$$

$$v^j_w \leq w^\top I^j_w \qquad \forall j \leq k, w \in GEN \quad (9)$$

$$\sum_{1 \leq j \leq k} I^j_{\mathbf{w}} = 1 \qquad \forall \mathbf{w} \in GEN \quad (10)$$

$$I^j_{\mathbf{w}} \in \{0, 1\} \qquad \forall j \leq k, \mathbf{w} \in GEN$$

Decision variables: $\mathbf{x}^j$, δ, $\mathbf{I}_{\mathbf{w}}$, $\mathbf{v}_w$

In this model, δ corresponds to the *setwise maximin utility* of the optimal set $\mathbf{Z}^*_W$. M is an arbitrary large number; $w^\top$ is some upper bound on the values taken by the weight parameters. Constraints 7, 8 and 9 ensures that δ is less than the utility of the best option in $\{\mathbf{x}^1, ..., \mathbf{x}^k\}$ for each $\mathbf{w}$, by introducing a variable v (for each w and each element of the set) to represent the value of minimum utility for the item selected, and indicators $\mathbf{I}_w$ to represent the selection. Only one $\mathbf{v}_w$ will be different from zero for each w, and since the objective function is maximized, the optimization will set $v^j_w = \mathbf{w} \cdot \mathbf{x}^j$ for the j such that $I^j_w = 1$; constraint 9 enforces 0 in the other cases.

Constraint 10 ensures that only one of the k items is selected for each utility function w.

We employ constraint generation in a way analogous to the single item case. At each step of the optimization, we compute the *setwise minimum utility*, solved using a series of LPs (as discussed above).

Alternative Heuristics Setwise optimization requires solving a large number of MIPs using constraint generation strategies. We also present a number of heuristic strategies that are computationally less demanding.

- The *current solution strategy* (CSS) proceeds as follows. Consider $\mathbf{w}^a$, the adversary's utility minimizing the utility of $\mathbf{x}^*_W$, the current maximin optimal recommendation; $u(\mathbf{x}^*_W; \mathbf{w}^a) = MU(\mathbf{x}^*_W; W)$. Let's further consider $\mathbf{x}^a = \arg\max_{\mathbf{x}\in\mathbf{X}} u(\mathbf{x}; w^a)$. CSS will return the set $\mathbf{Z}_{CSS} = \{\mathbf{x}^*_W, \mathbf{x}^a\}$. We extend this to sets with cardinality greater than two. Considering a set $\mathbf{Z}$, define $\mathbf{w}^a(\mathbf{Z}) = \arg\min_{\mathbf{w}\in W} \max_{\mathbf{x}\in\mathbf{Z}} u(\mathbf{x}; w)$ and be $\mathbf{x}^a(\mathbf{Z}) = \arg\max_{\mathbf{x}\in\mathbf{X}} u(\mathbf{x}; w^a(\mathbf{Z}))$. The *chain of adversaries* strategy constructs a set of size k starting by initializing $\mathbf{Z}$ to be $\mathbf{Z}_{CSS}$, the set of size two returned by the current solution strategy, and then iteratively add one element ($k-2$ times) by setting $\mathbf{Z} := \mathbf{Z} \cup \mathbf{x}^a(\mathbf{Z})$.
- The *query iteration strategy* (QIS) directly applies the T operator until a fixed point is reached. A fixed point is such that $SMU(T(\mathbf{Z}); W) = SMU(\mathbf{Z}; W)$.

5 Experiments

Using randomly generated elicitation data we ran a number of experiments using the algorithms described above. For all experiments, we generated constraints on the possible options using random binary constraints of the form $\neg f_1 \vee \neg f_2$ where f_1 and f_2 are features. We also assume some prior knowledge of user preferences, represented by random utility constraints of the form $\mathbf{w} \cdot \mathbf{x_k} \geq \mathbf{w} \cdot \mathbf{x_l}$, where $\mathbf{x_k}$ and $\mathbf{x_l}$ are random assignments $\in [0,1]^m$ (not necessarily feasible options) sampled with uniform probability over all possible assignments. The user's preference values $w_1 \ldots w_n$ are random and normalized such that $\sum_{w\in W} w = 1$. Finally, for all experiments we use recommendation/query sets of size (k) 3.

First, we ran experiments to determine how the runtime of the algorithms are affected by increasing instance sizes. This was done by running the algorithms on instances ranging from 10 to 15 features, with 30 experiments performed on each size. The average runtimes for these experiments can be seen in figure 1. As seen in the figure, runtime of exact SMMU computation becomes rapidly higher, and we were unable to perform experiments with more than 15 features, as several of the 30 experiments per size would time out with 16 features. In contrast to this we see that the runtime of the CSS and QIS algorithms do not rise significantly as the number of features increase. Due to this, we focus on the performance of CSS and QIS in the following experiments, as SMMU computation is too slow for practical use.

Using the CSS and QIS algorithms, we ran experiments to determine how the MMU optimal recommendation improves as more queries are asked. These were performed using larger instances, with 30 features per instance, 40 binary feature constraints and 40 utility constraints. In figure 2 we present the utility loss from recommending the MMU optimal recommendation as opposed to the optimal recommendation according to the user's preferences, as a function of

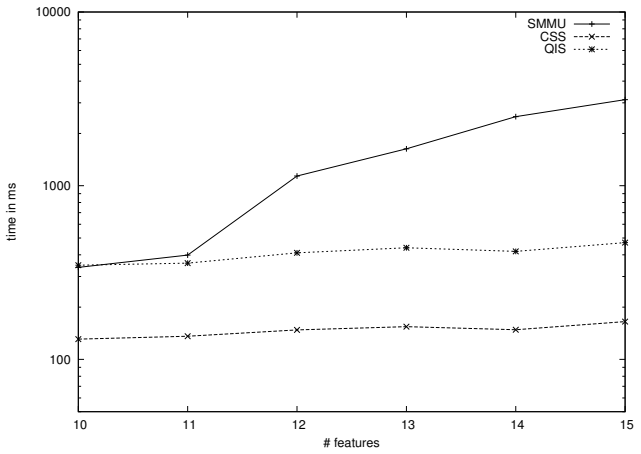


Figure 1. Average runtime of query computation for an increasing number of features. Averaged over 30 instances per size, with k = 3

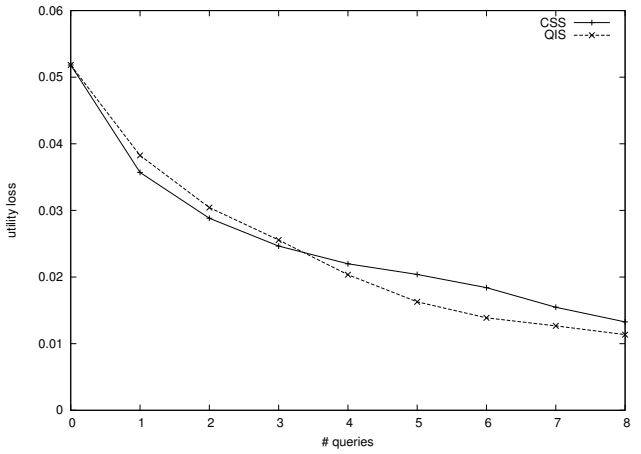


Figure 2. Average utility loss of the optimal recommendation as a function of the number of queries, using the CSS and QIS algorithms. Averaged over 30 instances, with k = 3.

the number of queries. The CSS and QIS algorithms have comparable performance, both improving utility loss by a small margin.

In figure 3 we show the minimum utility guarantee from the MMU recommendation as it increases with more queries asked. It quickly increases with the first 4-5 queries, but after that there is little improvement. While our theoretical results show that there is a connection between the problem of generating recommendations and queries, our results show that the pessimistic maximin decision criterion is generally not able to effetively elicit user preferences beyond the first few queries. In this case, it might be useful to adopt a non-myopic approach, or an alternative decision criterion.

Further investigation is required to determine in which settings our framework can be used effectively in interactive elicitation, and how to avoid stalling.

We also note that it is of course possible to use maximin as a decision criterion, while resorting to other strategies (perhaps based on regret or on probabilistic methods) to decide the next query.

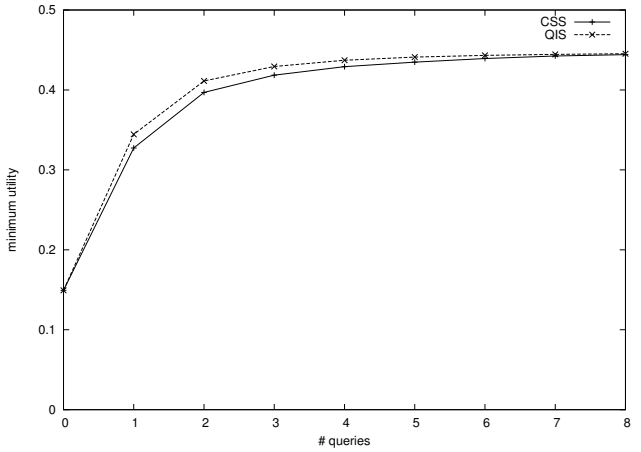


Figure 3. Average minimum utility as a function of the number of queries using the CSS and QIS algorithms. Averaged over 30 instances, with k = 3.

6 Discussion

In this paper we have developed a novel formalization for decision-making under utility uncertainty (making recommendations) using the maximin utility criterion. This approach allows for the highest degree of robustness, as the option recommended is guaranteed to ensure highest utility in the worst case. We formulated the problem of generating recommendation sets and introduced a new decision criteria. We developed computational MIP methods for optimal recommendation sets, as well as tractable approximations.

Moreover, following analogous models available for the minimax regret and Bayesian frameworks, we showed the connection between the problem of generating optimal recommendation sets and myopically optimal elicitation queries. This shows that our setwise maximin criterion, a natural extension of maximin to sets, in addition to providing robust recommendation sets, also serves as a means of generating myopically optimal choice queries (asking the user to pick his most preferred option in a slate).

Finally, we provided preliminary experimental results, showing performance of our approach on randomly generated data. We showed that maximin as an elicitation framework can provide good initial queries, but in an interactive setting it often stalls before finding the optimal recommendation.

We conclude with a remark about the choice of the decision criterion. A common criticism about maximin is that it can be overly pessimistic. Indeed expected utility (assuming a prior is available) or minimax regret may yield better recommendations in many cases. However, when a decision maker wishes guarantees on the worstcase performance (perhaps in critical decisions with high stakes), she must be willing to sacrifice "average" utility for such guarantee. This is the price to pay for the (strong) worstcase guarantees of maximin! We argue that the question of what criterion to use is almost philosophical, as there is no "right" or "wrong" decision criterion (each one might be better suited to different decision contexts).

REFERENCES

[1] Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Proc. of AAAI-02*, pages 239–246, Edmonton, 2002.

[2] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artifical Intelligence*, 170(8–9):686–713, 2006.

[3] Craig Boutilier, Richard S. Zemel, and Benjamin Marlin. Active collaborative filtering. In *Proc. of UAI-03*, pages 98–106, Acapulco, 2003.

[4] Darius Braziunas and Craig Boutilier. Minimax regret-based elicitation of generalized additive utilities. In *Proc. of UAI-07*, pages 25–32, Vancouver, 2007.

[5] Darius Braziunas and Craig Boutilier. Elicitation of factored utilities. *AI Magazine*, 29(4):79–92, 2008.

[6] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Proc. of AAAI-2000*, pages 363–369, Austin, TX, 2000.

[7] Peter C. Fishburn. Interdependence and additivity in multivariate, unidimensional expected utility theory. *International Economic Review*, 8:335–342, 1967.

[8] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, New York, 1976.

[9] Jordan J. Louviere, David A. Hensher, and Joffre D. Swait. *Stated Choice Methods: Analysis and Application*. Cambridge University Press, Cambridge, 2000.

[10] Bart Peintner, Paolo Viappiani, and Neil Yorke-Smith. Preferences in interactive systems: Technical challenges and case studies. *AI Magazine*, 29(4):13–24, 2008.

[11] Robert Price and Paul R. Messinger. Optimal recommendation sets: Covering uncertainty over user preferences. In *Proc. of AAAI-05*, pages 541–548, 2005.

[12] James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. Incremental critiquing. *Knowledge-Based Systems*, 18(4–5):143–151, 2005.

[13] Ahti Salo and Raimo P. Hämäläinen. Preference ratios in multiattribute evaluation (PRIME)–elicitation and decision procedures under incomplete information. *IEEE Trans. on Systems, Man and Cybernetics*, 31(6):533–545, 2001.

[14] Olivier Toubia, John Hauser, and Duncan Simester. Polyhedral methods for adaptive choice-based conjoint analysis. (4285-03), 2003.

[15] Paolo Viappiani and Craig Boutilier. Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys09)*, pages 101–108, New York, 2009.

[16] Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems 23 (NIPS)*, Vancouver, 2010.

Explaining Qualitative Preference Models

Wietske Visser and **Koen V. Hindriks** and **Catholijn M. Jonker**[1]

Abstract. We propose an explanation facility for a qualitative preference representation framework. We show how an explanation can be provided for qualitative, multi-criteria preferences based on the criteria that are used to decide preferences between outcomes. Such a facility provides an important tool for a user to understand how preferences are determined. We show that this facility can also be used by a user to inform the system about its preferences. Such a user-provided explanation can be used for updating and improving a preference model maintained by the system.

1 INTRODUCTION

A preference representation framework provides a tool for determining preferences between outcomes. That is, for any two outcomes it can determine whether one is strictly preferred to the other, both are equally preferred, or they are incomparable. In this paper, we discuss an additional facility, namely the *explanation* of preferences maintained by such a system. Explanation of preferences is useful and important in many cases, such as situations where a decision maker has to explain his decision to other actors; where a decision support system that is elicited from an expert has to explain its list of recommended options to a non-expert user; or where agents may give each other feedback on offers in negotiation, without revealing all their preferences [6]. Another reason to use explanation is to improve users' confidence in a system, since lack of confidence is an obstacle to acceptance and practical use of the system [7]. In these cases, it is not satisfactory to just present the preference model. Although this model does contain all information on which the preference is based, the format is not suitable for presentation to a user. First, the model is too technical for the average human user to interpret. Second, even experts may have trouble interpreting the model since it may be quite large, and hence it would be hard to quickly find the reason behind the preference.

Besides explaining someone's preferences to another party, explanation may also be used 'in reverse' during preference elicitation and updating. Here the idea is as follows. The user is not only asked to state his preference between two given outcomes, but also to explain this preference. This explanation can then be used to update the preference model in such a way that the explanation for the user's preference that would be generated by the updated model coincides with the explanation given by the user.

In this paper we propose an approach to generate explanations from a preference model and to use explanations to update a preference model. The preference models we consider are expressed in a particular preference representation framework called Qualitative Preference Systems (QPS) [8, 9]. QPS is a general framework for the representation of qualitative, multi-criteria preferences. In Section 2, we give a summary of the QPS framework. In Section 3 we propose a way to explain qualitative preferences by the deciding criteria, and discuss in particular how this can be implemented for QPS models. In Section 4 we discuss how such explanations, if given by the user of a system, can be used to update the system's current model of the user's preferences. We give detailed interaction diagrams that indicate when and how a QPS preference model should be altered. Section 5 concludes the paper.

[1] Interactive Intelligence Group, Delft University of Technology, The Netherlands, email: {Wietske.Visser, K.V.Hindriks, C.M.Jonker}@tudelft.nl

2 QUALITATIVE PREFERENCE SYSTEMS

The main aim of the Qualitative Preference System (QPS) framework [8, 9] is to determine preferences between *outcomes* in a purely *qualitative* way. Outcomes are defined as variable assignments that respect the constraints in a *knowledge base*. The preferences between outcomes are based on multiple *criteria*. Every criterion can be seen as a *reason* for preference, or as a preference from one particular *perspective*. We distinguish between simple and compound criteria. Simple criteria are based on a single variable. Multiple (simple) criteria can be combined in a compound criterion to determine an overall preference. QPS distinguishes between two kinds of compound criteria: cardinality criteria and lexicographic criteria. The subcriteria of a cardinality criterion all have equal priority, and preference is determined by a kind of voting mechanism that counts the number of subcriteria that support a certain preference and those that do not. In a lexicographic criterion, the subcriteria are ordered by priority and preference is determined by the subcriteria with the highest priority; lower priority subcriteria only influence the preference if the higher priority subcriteria are indifferent.

Definition 1. (Qualitative Preference System [8]) A *Qualitative Preference System (QPS)* is a tuple $\langle Var, Dom, K, \mathcal{C} \rangle$. *Var* is a finite set of *variables*. Every variable $X \in Var$ has a domain $Dom(X)$ of possible values. K (a *knowledge base*) is a set of constraints on the assignments of values to the variables in *Var*. An *outcome* α is an assignment of a value $x \in Dom(X)$ to every variable $X \in Var$, such that no constraints in K are violated. Ω denotes the set of all outcomes: $\Omega \subseteq \prod_{X \in Var} Dom(X)$. α_X denotes the value of variable X in outcome α. $\mathcal{C}$ is a finite rooted tree of criteria, where leaf nodes are simple criteria and other nodes are compound criteria. Child nodes of a compound criterion are called its subcriteria. The root of the tree is called the top criterion. Weak preference between outcomes by a criterion c is denoted by the relation $\succeq_c$. $\succ_c$ denotes the strict subrelation, $\approx_c$ the indifference subrelation. $\alpha \lambda_c \beta$ denotes that $\alpha \not\preceq_c \beta$ and $\beta \not\preceq_c \alpha$.

Definition 2. (Simple criterion [8]) A *simple criterion* c is a tuple $\langle X_c, \geq_c \rangle$, where $X_c \in Var$ is a variable, and $\geq_c$, a preference relation on the possible values of X_c, is a preorder on $Dom(X_c)$. $>_c$ is the strict

Table 1. Explanations

	lexicographic criterion c	goal-based cardinality criterion c
$\alpha \succ_c \beta$	any subcriterion $s \in C_c$ such that $\alpha \succ_s \beta$ and for all $s' \in C_c$: if $s' \rhd s$ then $\alpha \approx_{s'} \beta$ and if $s' \triangle s$ then $\alpha \succeq_{s'} \beta$ or there is a $s'' \in C_c (s'' \rhd_c s'$ and $\alpha \not\approx_{s''} \beta)$	the set of subgoals $g \in C_c$ such that $\alpha \succ_g \beta$
$\alpha \approx_c \beta$	for all subcriteria $s \in C_c$: $\alpha \approx_s \beta$	the set of subgoals $g \in C_c$ such that $\alpha \succ_g \beta$ plus the set of subgoals $g \in C_c$ such that $\beta \succ_g \alpha$
$\alpha \wedge_c \beta$	1: any subcriterion $s \in C_c$ such that $\alpha \wedge_s \beta$ and for all $s' \in C_c$: if $s' \rhd s$ then $\alpha \approx_{s'} \beta$ 2: any pair of subcriteria (s_1, s_2) where $s_1, s_2 \in C_c$ such that $\alpha \succ_{s_1} \beta$ and $\beta \succ_{s_2} \alpha$ and $s_1 \triangle_c s_2$ and for all $s' \in C_c$: if $s' \rhd_c s_1$ or $s' \rhd_c s_2$ then $\alpha \approx_{s'} \beta$	n/a

subrelation, $\doteq_c$ is the indifference subrelation. A simple criterion $c = \langle X_c, \geq_c \rangle$ *weakly prefers* an outcome α over an outcome β, denoted $\alpha \geq_c \beta$, iff $\alpha_{X_c} \geq_c \beta_{X_c}$.

Definition 3. (Goal [9]) A QPS *goal* is a simple criterion $\langle X, \geq \rangle$, where $X \in Var$ is a Boolean variable ($Dom(X) = \{\top, \bot\}$), and $\top \rhd \bot$.

Definition 4. (Goal-based cardinality criterion [9]) A *goal-based cardinality criterion* c is a tuple $\langle C_c \rangle$ where C_c is a nonempty set of goals (the *subcriteria* or *subgoals* of c). A goal-based cardinality criterion $c = \langle C_c \rangle$ *weakly prefers* an outcome α over an outcome β, denoted $\alpha \geq_c \beta$, iff $|\{s \in C_c \mid \alpha \succ_s \beta\}| \geq |\{s \in C_c \mid \alpha \not\prec_s \beta\}|$, or equivalently, iff $|\{s \in C_c \mid \alpha_{X_s} = \top\}| \geq |\{s \in C_c \mid \beta_{X_s} = \top\}|$.

Note that a goal-based cardinality criterion can only have goals as subcriteria. This is to guarantee transitivity of the preference relation induced by a cardinality criterion [8].

Definition 5. (Lexicographic criterion [8]) A *lexicographic criterion* c is a tuple $\langle C_c, \rhd_c \rangle$, where C_c is a nonempty set of criteria (the *subcriteria* of c) and $\rhd_c$, a *priority relation* among subcriteria, is a strict partial order (a transitive and asymmetric relation) on C_c. $s \triangle_c s'$ denotes that $s \not\rhd_c s'$ and $s' \not\rhd_c s$. A lexicographic criterion $c = \langle C_c, \rhd_c \rangle$ *weakly prefers* an outcome α over an outcome β, denoted $\alpha \geq_c \beta$, iff $\forall s \in C_c (\alpha \geq_s \beta \vee \exists s' \in C_c (\alpha \succ_{s'} \beta \wedge s' \rhd_c s))$.

3 EXPLAINING PREFERENCES

Ideally, any explanation given to a human user should be easily understandable by that user. Therefore, both the content and the format of the explanation matter. [6] distinguishes between two steps in explanation generation. First, the content of the explanation has to be selected. Next, a natural language explanation has to be generated. Like [6], we focus on the first step and only look at the content of an explanation. An example of natural language generation for evaluative arguments such as explanations can be found in [2].

We are not aware of any work on the explanation of preferences represented in a qualitative framework, but some work has been done on the explanation of (decisions based on) quantitative preferences. Klein and Shortliffe [5] presented strategies for automatically explaining decisions based on Multiattribute Value Theory (a quantitative preference representation framework). The explanations are based on the compellingness of objectives. Labreuche [6] presents a general framework for explaining the results of a multi-attribute preference model. He takes a quantitative approach where the utilities of the combined criteria are weighted and summed to obtain an overall utility. He develops a formal framework that justifies the selection of arguments (criteria) to be presented as explanation of a preference.

One of the main differences between quantitative and qualitative approaches to multi-criteria preference modelling is that quantitative approaches are compensatory, whereas their qualitative counterparts are not. In quantitative approaches, a low score on one criterion can be compensated by high scores on other criteria, even if the other criteria are less important, as long as the scores are high enough. In qualitative approaches, this is not possible. For example, if one outcome is preferred to another according to the highest priority subcriterion of a lexicographic criterion, it will also be preferred according to this lexicographic criterion, no matter what the preferences of the other subcriteria are. This allows us to precisely identify the criteria that are 'responsible' or 'deciding' for the overall preference. It is our intuition that these criteria also provide a natural explanation for the overall preference.

Explanations for preferences by QPS criteria

We now turn to the question how a preference between two outcomes by a QPS criterion can be explained. The answer to this question depends on the kind of criterion that is considered. Preferences by simple criteria (including goals) are self-explanatory, since they follow immediately from the specification of the simple criterion or goal. For example, a simple criterion c strictly prefers an outcome α to an outcome β because α's value of X_c is better than β's value of X_c. Similarly, a goal c strictly prefers an outcome α to an outcome β because α satisfies c but β does not. Of course, these facts may in turn require explanation. But since this would be explanation of knowledge (factual information about outcomes) rather than preferences, we do not discuss this topic here.

Preferences by compound criteria can be explained by the subcriteria that are *deciding* in the overall preference. Which subcriteria are deciding depends both on the kind of compound criterion (lexicographic or goal-based cardinality criterion) and on the kind of preference (strict, equal or incomparable). The deciding factor may be a single subcriterion, a pair, or even a set of multiple subcriteria that together determine the overall preference. In the following, we discuss the deciding subcriteria (and hence the explanations) for both kinds of compound criteria and for all kinds of preferences. An overview is given in Table 1.

Lexicographic criteria

Strict preference Suppose a lexicographic criterion c strictly prefers an outcome α over an outcome β ($\alpha \succ_c \beta$). The explanation of this preference is given by a subcriterion s that strictly prefers α to β ($\alpha \succ_s \beta$). But not just any subcriterion that strictly prefers α to β will do. First, every subcriterion s' with a higher priority than s ($s' \rhd_c s$) has to be indifferent: $\alpha \approx_{s'} \beta$, otherwise s would

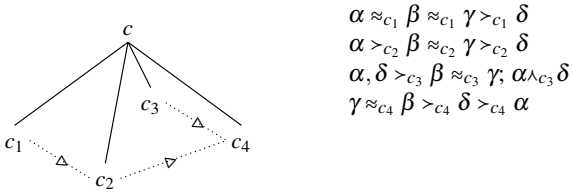


Figure 1. Example lexicographic criterion

have been overruled by s'. Second, every subcriterion s' whose priority is incomparable to that of s ($s' \triangle_c s$) and which is not overruled ($\forall s'' \rhd_c s' : \alpha \approx_{s''} \beta$) has to agree with s or be indifferent ($\alpha \succeq_{s'} \beta$), otherwise s would not have decided the preference by c.

Example 1. Consider the lexicographic criterion c displayed in Figure 1. It has four subcriteria c_1, c_2, c_3, c_4 such that $c_1 \rhd_c c_2 \rhd_c c_4$ and $c_3 \rhd_c c_4$. The nature of the subcriteria is unspecified, but their preferences regarding four outcomes $\alpha, \beta, \gamma, \delta$ are given. The criterion c strictly prefers α over β: $\alpha \succ_c \beta$. The subcriteria that can explain this preference are c_2 and c_3. c_3 strictly prefers α over β, and is undominated. c_2 also strictly prefers α over β, and is dominated only by an indifferent criterion (c_1). Neither is 'contradicted' by a criterion with incomparable priority.

Equal preference A lexicographic criterion c is only indifferent between two outcomes α and β ($\alpha \approx_c \beta$) if all its subcriteria are indifferent between α and β. No single subcriterion is deciding in the overall preference, but all subcriteria contribute equally (note that priority does not matter, since indifferent criteria do not overrule lower priority criteria). This means that the explanation of the indifference is given by the fact that all subcriteria are indifferent.

Example 2. Consider again the lexicographic criterion c in Figure 1. c is indifferent between β and γ, because all subcriteria are indifferent between β and γ.

Incomparability If a lexicographic criterion c cannot compare between two outcomes α and β ($\alpha \wedge_c \beta$), this incomparability can have two possible reasons. First, the incomparability may result from a subcriterion s that cannot compare between α and β ($\alpha \wedge_s \beta$). Like in the case of strict preference, every subcriterion s' with a higher priority than s ($s' \rhd_c s$) has to be indifferent: $\alpha \approx_{s'} \beta$, otherwise s would have been overruled by s'.

Example 3. Consider again the lexicographic criterion c in Figure 1. c cannot compare between α and δ. This is due to subcriterion c_3, which cannot compare between α and δ, and which is not overruled by any other subcriterion. Therefore c_3 explains c's incomparability between α and δ.

Second, the incomparability may result from two conflicting subcriteria that do not overrule each other. That is, there is one subcriterion s_1 that strictly prefers α to β ($\alpha \succ_{s_1} \beta$), and all higher priority subcriteria are indifferent. There is also another subcriterion s_2 that strictly prefers β to α ($\beta \succ_{s_2} \alpha$), and all higher priority subcriteria are indifferent. Note that this also means that s_1 and s_2 have incomparable priorities, which means that neither overrules the other, so no preference can be determined. In this case, the subcriteria s_1 and s_2 together explain the incomparability.

Example 4. Consider again the lexicographic criterion c in Figure 1. c cannot compare between γ and δ. Subcriterion c_3 strictly prefers δ over γ, the other three subcriteria stricly prefer γ over δ. Not all subcriteria are suitable to explain the incomparability. c_4 is discarded because c_3 has higher priority. But also c_2 should not be used, even though it is incomparable in priority with c_3. This is because c_1 has higher priority and is not indifferent. This makes c_1 and c_3 the deciding criteria that are used as explanation.

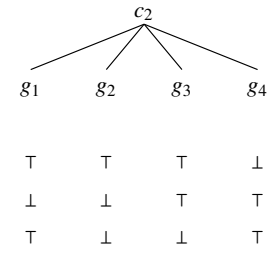


Figure 2. Example goal-based cardinality criterion

Goal-based cardinality criteria

Strict preference Suppose a goal-based cardinality criterion c strictly prefers an outcome α over an outcome β ($\alpha \succ_c \beta$). Then this is because the subgoals that α satisfies outnumber the subgoals that β satisfies. There may be subgoals that are satisfied by both α and β. They are counted on both sides, but do not influence the overall preference between α and β. Therefore, as an explanation of c's preference of α over β we only consider the subgoals g that α satisfies but β does not ($\alpha_{X_g} = \top$ and $\beta_{X_g} = \bot$, or equivalently, $\alpha \succ_g \beta$).

Example 5. Consider the goal-based cardinality criterion c_2 displayed in Figure 2. It has four goals g_1, g_2, g_3, g_4 as subcriteria. For three outcomes α, β, γ it is given whether they satisfy each of the four goals. The criterion c_2 strictly prefers α to β. The explanation of this preference is given by the goals g_1 and g_2 that α satisfies but β does not. Although α also satisfies goal g_3, this goal is not used in the explanation since it is also satisfied by β and hence is not deciding in the overall preference. Similarly, c_2's preference of α over γ can be explained by the goals g_2 and g_3.

Equal preference If a goal-based cardinality criterion c equally prefers two outcomes α and β ($\alpha \approx_c \beta$), this means that both outcomes satisfy the same number of subgoals of c. However, it does not necessarily mean that both outcomes satisfy the same goals. As explanation, we take the goals that α satisfies but β does not, and the set of goals that β satisfies but α does not. Both (disjoint) sets contain the same number of goals, which compensate for each other. This explains the indifference between the two outcomes.

Example 6. Consider again the goal-based cardinality criterion c_2 in Figure 2. c_2 is indifferent between β and γ. Both outcomes satisfy two goals, but one goal (g_4) is satisfied by both outcomes. Therefore the explanation of the indifference is given by g_3 (which is satisfied by β but not by γ) and g_1 (which is satisfied by γ but not by β).

4 USING EXPLANATION TO UPDATE A PREFERENCE MODEL

Before a preference model can be used in practice in a system, it has to be constructed or instantiated. Preference elicitation is likely to be an iterative process, and for this reason an existing preference model

should also be updateable. There are several ways of constructing and updating a preference model. In this paper we focus on the approach of guiding preference elicitation by asking the user particular questions and updating the preference model according to the answers. The advantages of this approach are that it provides an intuitive interaction with non-expert users and that preferences can be discovered during the process. In particular, we consider the case in which the user is asked not only to give his preference between two outcomes, but also to provide an explanation for this preference. This explanation can then be used to update the current preference model. If the user just provides his preference between outcomes, there may be many different ways in which the model could be updated to reflect this preference. The added value of additionally obtaining an explanation from the user is that it provides clues on how exactly the model should be updated, possibly after some further interaction involving targeted follow-up questions.

Updating a QPS model with explanations

We investigate how a system's current model of the user's preferences can be updated by engaging in a conversation with the user. Using explanations of preferences given by a user, the system can find out whether its current representation is accurate, and if not, where it has to be changed. Our approach allows for an initial model to be present that can be adapted by the user. The user can add preference information on his own initiative, or alternatively the system can ask the user to provide specific preferences (for example between two outcomes that are incomparable in its current model). In any case, if the preference given by the user does not match the preference that follows from the system's current model, the user is asked to provide an explanation. We assume that the user's explanation of his preference coincides with one of the explanations listed in Table 1. Depending on the user's answer and the nature of the top criterion (lexicographic or goal-based cardinality), the system can proceed by asking follow-up questions or updating its preference model in a particular way.

In the following, we discuss every situation in detail and provide interaction diagrams for each. We assume that the user has stated a preference between two outcomes that is not supported by the system's current preference model. It is important to distinguish between the current preference model maintained by the system, and the statements of the user. Since the interaction is designed to identify the elements of the model that need to be updated, the user's statements typically disagree with the current model. The interaction diagrams start with the system asking for an explanation for the given preference. The system's possible responses depend on the explanation given and the current preference model. More than one response may be applicable. In that case, the system should keep the interaction going until the preference model induces the given preference. When the process is finished, the updated preference model should not only model the preference given by the user, but also generate the same explanation for it.

Lexicographic criteria

Strict preference The interaction diagram for updating a preference model with a strict preference of an outcome α over an outcome β by a lexicographic criterion c is given in Figure 3. The explanation of such a preference is given by a subcriterion s of c that, according to the user, strictly prefers α to β. There can be different reasons why this subcriterion does not decide c's preference in the current preference model S.

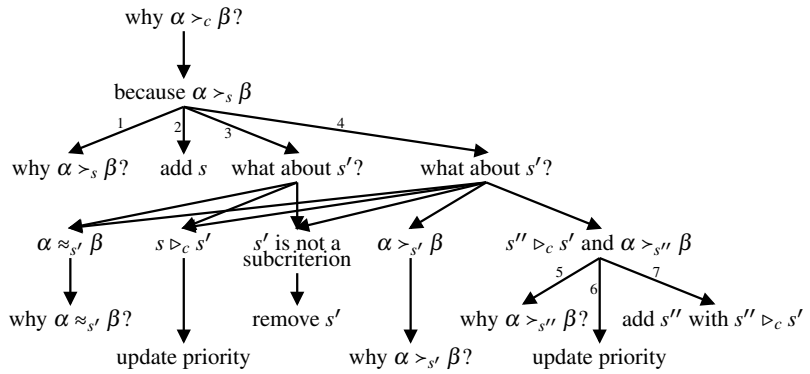
- First, s may not strictly prefer α to β according to S. In this case, the user is asked to explain this preference.
- Second, s may not be listed as a subcriterion of c in S. In this case, the system adds s to the set of subcriteria C_c.
- Third, according to S there may be another subcriterion s' that overrules s, i.e. that has higher priority but is not indifferent between α and β. In this case, the user is asked to clarify this issue, and may respond in several ways. (i) If the user states that s' actually is indifferent, he is asked for an explanation. (ii) If the user states that s actually has higher priority than s', the system updates the priority relation accordingly. (iii) If the user states that s' is not actually a subcriterion, the system removes s' from C_c.
- Fourth, according to S there may be another subcriterion s' that is not comparable in priority to s, does not weakly prefer α to β, and is not overruled. In this case, the user is asked to clarify this issue. The same responses by the user as in the previous case are possible, plus two more. (iv) If the user states that s' actually strictly prefers α to β, he is asked to give an explanation. (v) If the user states that there actually is another subcriterion s'' with higher priority that strictly prefers α to β, there are three options. If the preference does not follow from S, then the user is asked for an explanation. If s'' does not have higher priority than s' in S, the system updates the priority relation. And if s'' was not listed as a subcriterion of c, the system adds it with the right priority.

Equal preference The interaction diagram for updating a preference model with an equal preference between two outcomes α and β by a lexicographic criterion c is given in Figure 4. Such a preference is explained by the fact that, according to the user, all subcriteria are indifferent. There can only be one reason that the indifference does not follow from the current preference model S.

- There must be a subcriterion s in S that is not indifferent. In this case, the user is asked to clarify this issue. He can do so in two different ways. (i) If the user states that s is actually indifferent, he is asked to give an explanation. (ii) If the user states that s is not actually a subcriterion of c, then the system removes s from the set of subcriteria C_c.

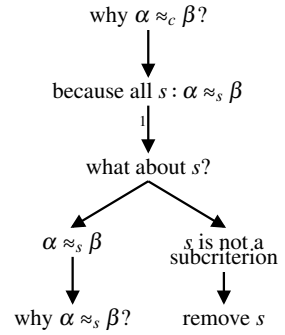
Incomparability The interaction diagram for updating a preference model with an incomparability between two outcomes α and β by a lexicographic criterion c is given in Figure 5. Since there are two kinds of explanation of such an incomparability, the interaction tree splits into two branches. If the incomparability is explained by a subcriterion that cannot compare between α and β according to the user, the possible responses are very similar to the case of strict preference. Therefore we do not discuss this case here but refer to the lefthand branch in Figure 5 for the details. If the incomparability is explained by two contradicting subcriteria s_1 and s_2, where $\alpha \succ_{s_1} \beta$ and $\beta \succ_{s_2} \alpha$ according to the user, there can be different reasons why these subcriteria do not decide c's preference in the current preference model S.

- First, it may be that $\alpha \not\succ_{s_1} \beta$ or $\beta \not\succ_{s_2} \alpha$ according to the current preference model S. In this case, the user is asked to explain that preference.
- Second, s_1 or s_2 may not be listed as a subcriterion of c in S. In this case, the system adds it to the set of subcriteria C_c.
- Third, according to S there may be another subcriterion s_1' that overrules s_1. In this case, the user can reply in different ways. (i) If the user states that s_1' is actually indifferent between α and β, he is asked for an explanation. (ii) If the user states that s_1' does not actually have higher priority than s_1, the system updates the priority relation ac-



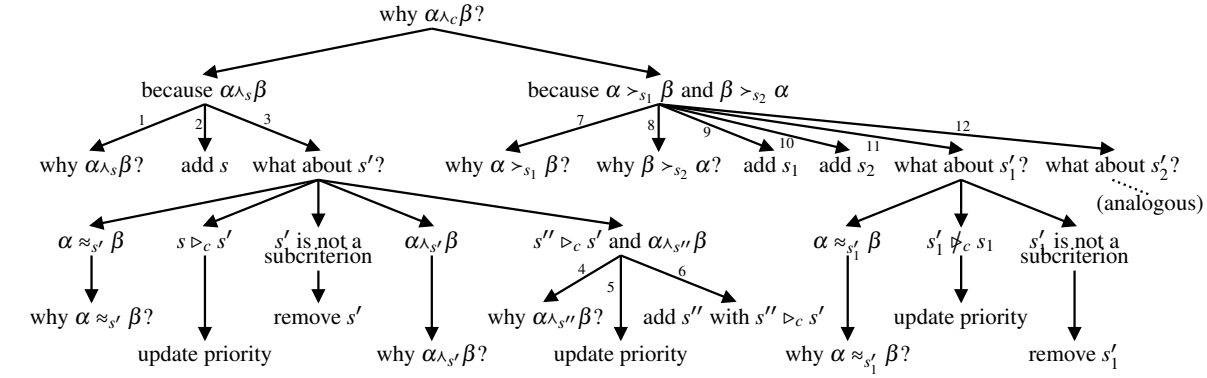
1: if $\alpha \not\succ_s \beta$
2: if $s \notin C_c$
3: for any $s' \in C_c$ such that $s' \rhd_c s$ and $\alpha \not\approx_{s'} \beta$
4: for any $s' \in C_c$ such that $s' \triangle s$ and $\alpha \not\succeq_{s'} \beta$ and s' is not overruled
5: if $\alpha \not\succ_{s''} \beta$
6: if $s'' \not\rhd_c s$
7: if $s'' \notin C_c$

Figure 3. Updating with a strict preference by a lexicographic criterion



1: for any $s \in C_c$ such that $\alpha \not\approx_s \beta$

Figure 4. Updating with an equal preference by a lexicographic criterion



1: if $\alpha \not\wedge_s \beta$
2: if $s \notin C_c$
3: for any $s' \in C_c$ such that $s' \rhd_c s$ and $\alpha \succ_{s'} \beta$ or $\beta \succ_{s'} \alpha$
4: if $\alpha \not\wedge_{s''} \beta$
5: if $s'' \not\rhd_c s'$
6: if $s'' \notin C_c$
7: if $\alpha \not\succ_{s_1} \beta$
8: if $\beta \not\succ_{s_2} \alpha$
9: if $s_1 \notin C_c$
10: if $s_2 \notin C_c$
11: for any $s'_1 \in C_c$ such that $s'_1 \rhd_c s_1$ and $\alpha \not\approx_{s'_1} \beta$
12: for any $s'_2 \in C_c$ such that $s'_2 \rhd_c s_2$ and $\alpha \not\approx_{s'_2} \beta$

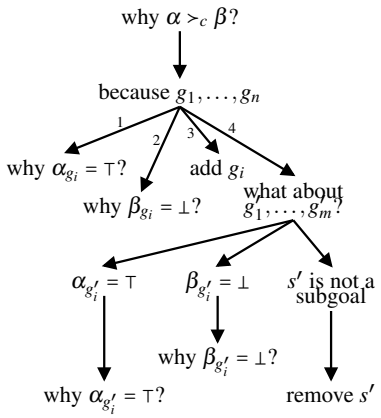
Figure 5. Updating with an incomparability by a lexicographic criterion

cordingly. (iii) If the user states that s'_1 is not actually a subcriterion of c, then the system removes s'_1 from the set of subcriteria C_c.
• Fourth, according to S there may be another subcriterion s'_2 that overrules s_2. This case is handled analogously to the third case.

Goal-based cardinality criteria

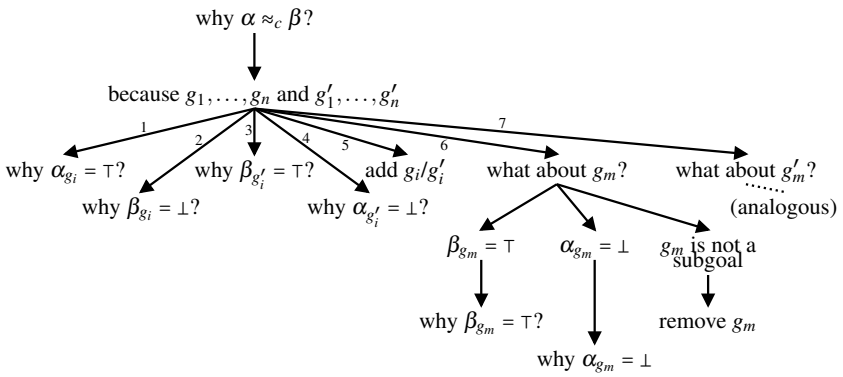
Strict preference The interaction diagram for updating a preference model with a strict preference of an outcome α over an outcome β by a goal-based cardinality criterion c is given in Figure 6. The explanation of such a preference is given by a set of subgoals $g_1, \ldots, g_n$ that are all satisfied by α but not by β according to the user. There can be different reasons why this set of goals does not decide c's preference in the current preference model S.
• First, one of the goals may not be satisfied by α in S. In this case, the user is asked to explain this fact.
• Second, one of the goals may be satisfied by β in S. In this case, the user is also asked to give an explanation.
• Third, one of the goals may not be listed as a subgoal of c in S. In this case, the system adds it to the set of subgoals C_c.
• Fourth, there may be a set of goals $g'_1, \ldots, g'_m$ that are all satisfied by β but not by α according to S, which contains at least as many goals as $g_1, \ldots, g_n$. In this case, the user is asked to clarify this issue, and may respond in several ways. (i) If the user states that one of the goals is actually satisfied by α or (ii) not satisfied by β, he is asked to for an explanation. (iii) If the user states that one of the goals is actually not a subgoal of c, then the system removes this goal from the set of subgoals C_c.

Equal preference The interaction diagram for updating a preference model with an equal preference between two outcomes α and β by a goal-based cardinality criterion c is given in Figure 7. The explanation of such a preference is given by two equally sized sets of subgoals: $g_1, \ldots, g_n$ that are all satisfied by α but not by β, and $g'_1, \ldots, g'_n$ that are all satisfied by β but not by α according to the user. Again, there can be different reasons why these sets of goals do not decide c's preference in the current preference model S.
• First, according to S, α may not satisfy some g_i, β may satisfy some g_i, β may not satisfy some g'_i, or α may satisfy some g'_i. In this case, the user is asked to give an explanation.



1: if $\alpha_{g_i} \neq \top$
2: if $\beta_{g_i} \neq \bot$
3: if $g_i \notin C_c$
4: for any $g'_1, \ldots, g'_m$ such that $m \geq n$ and for all $g'_i : \beta \succ_{g'_i} \alpha$

Figure 6. Updating with a strict preference by a goal-based cardinality criterion



1: if $\alpha_{g_i} \neq \top$
2: if $\beta_{g_i} \neq \bot$
3: if $\beta_{g'_i} \neq \top$
4: if $\alpha_{g'_i} \neq \bot$
5: if $g_i/g'_i \notin C_c$
6: for any $g_m \in C_c$ such that $g_m \notin g_1, \ldots, g_n$ and $\alpha \succ_{g_m} \beta$
7: for any $g'_m \in C_c$ such that $g'_m \notin g'_1, \ldots, g'_n$ and $\beta \succ_{g'_m} \alpha$

Figure 7. Updating with an equal preference by a goal-based cardinality criterion

• Second, any g_i or g'_i may not be listed as a subgoal of c in S. In this case, the system adds it to the set of subgoals C_c.
• Third, according to S there may be a goal g_m in C_c that is not in $g_1, \ldots, g_n$ and is satisfied by α but not by β. In this case, the user is asked to clarify this issue and may respond in several ways. (i) If the user states that β actually satisfies g_m, or (ii) α actually does not satisfy g_m, he is asked to explain this fact. (iii) If the user states that g_m is actually not a subgoal of c, then the system removes g_m from the set of subgoals C_c.
• Fourth, according to S there may be a goal g'_m in C_c that is not in $g'_1, \ldots, g'_n$ and is satisfied by β but not by α. This case is handled analogously to the third case.

5 CONCLUSION

Qualitative Preference Systems (QPS) [8, 9] provide a general framework for the representation of qualitative, multi-criteria preferences. We have shown that the composite tree structure of multiple criteria, combined with the non-compensatoriness of a qualitative approach provides a basis for the generation of explanations for the preferences that follow from a preference model represented in the QPS framework. The explanation strategy that we proposed is based on the intuition that preferences between outcomes can be explained by the criteria that are deciding in the overall preference. We identified the explanations that can be given for different preferences by different kinds of criteria. We then showed that the same explanations can also be useful when updating a preference model, because they provide information on how exactly the model should be updated.

Some interesting issues remain for future work. First, in some instances it may be necessary to explain facts about the outcomes involved in a preferential comparison, e.g. to explain why they do or do not satisfy a particular goal. Explanation of knowledge and reasoning is a separate field of study that may provide solutions to this issue. Second, when the system updates the priority relation between two subcriteria of a lexicographic criterion, this relation has to remain a partial order. Moreover, as the system iteratively engages in an interaction with the user as described here, it has to ensure that the previous preferences and explanations expressed by the user remain valid. It is important to investigate how such consistencies can be ensured. Third, the explanation of preferences may be part of a larger picture, for example in recommendation, decision making or planning. We would like to investigate how the explanation mechanism presented here can be embedded in other explanation mechanisms, such as the one presented in [1], where a tree structure of goals and beliefs is used to explain actions. Besides these theoretical considerations, we would like to take a more practical approach and implement the QPS framework together with the proposed explanation mechanism and update mechanism. We can then experimentally test the validity of our intuitions. This is related to the work of [3], who tested the predictive performance of the Take the Best (TTB) heuristic [4], which is a simplified instantiation of the lexicographic rule.

Acknowledgements 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. It is part of the Pocket Negotiator project with grant number VICI-project 08075.

REFERENCES

[1] J. Broekens, M. Harbers, K. Hindriks, K. van den Bosch, C. Jonker, and J.-J. Meyer, 'Do you get it? User-evaluated explainable BDI agents', in *Multiagent System Technologies*, *LNCS* 6251, 28–39, (2010).
[2] G. Carenini and J.D. Moore, 'Generating and evaluating evaluative arguments', *Artificial Intelligence*, **170**(11), 925–952, (2006).
[3] A. Dieckmann, K. Dippold, and H. Dietrich, 'Compensatory versus non-compensatory models for predicting consumer preferences', *Judgment and Decision Making*, **4**(3), 200–213, (2009).
[4] G. Gigerenzer, P.M. Todd, and ABC Group, *Simple heuristics that make us smart*, Oxford University Press, 1999.
[5] D.A. Klein and E.H. Shortliffe, 'A framework for explaining decision-theoretic advice', *Artificial Intelligence*, **67**(2), 201–243, (1994).
[6] C. Labreuche, 'A general framework for explaining the results of a multi-attribute preference model', *Artificial Intelligence*, **175**(7-8), 1410–1448, (2011).
[7] B. Moulin, H. Irandoust, M. Bélanger, and G. Desbordes, 'Explanation and argumentation capabilities:towards the creation of more persuasive agents', *Artificial Intelligence Review*, **17**(3), 169–222, (2002).
[8] W. Visser, R. Aydoğan, K.V. Hindriks, and C.M. Jonker, 'A framework for qualitative multi-criteria preferences', in *Proc. ICAART*, pp. 243–248, (2012).
[9] W. Visser, K.V. Hindriks, and C.M. Jonker, 'Goal-based qualitative preference systems', in *Proc. DALT*, (2012).

On Balancing Occupants' Comfort in Shared Spaces

Nic Wilson
Cork Constraint Computation Centre
Department of Computer Science,
University College Cork, Ireland
email: n.wilson@4c.ucc.ie

Abstract. Maintaining comfortable thermal conditions in an office environment is very important, as it can affect the quality of life of the occupants, their work productivity, and improve energy efficiency. One significant aspect of this task is how to balance the preferences of a number of occupants sharing the same space. We analyse and suggest some approaches to this problem, both for the case of optimising for a single time period, and for the problem of optimising over multiple different time periods.

1 INTRODUCTION

Maintaining comfortable thermal conditions in an office environment is clearly very important; apart from the quality of life of the occupants, it can also affect work productivity, and is very relevant for energy efficiency, since it can save considerable wastage through, for instance, overheating a room [15, 17]. One significant aspect of this task is how to balance the preferences of a number of occupants sharing the same space.

For a single occupant, one approach is, for each potential vector of environmental conditions, to produce a prediction of how they are likely to feel (e.g., a little warm), based on a scale for thermal sensation or comfort (see Section 2). We can then use this to generate a predicted dissatisfaction value. For example, if we are just using temperature to predict occupant responses, then each potential value of temperature is assigned a predicted dissatisfaction level, with a value of zero meaning completely comfortable. This therefore gives an objective function which we can aim to minimise. (This minimisation process can be complex, but is not the focus of the current paper.)

In Section 3 we consider the problem of evaluating overall occupant dissatisfaction for a single occupant over a number of (potentially temporally distant) time periods. Perhaps the main motivation for considering this situation is as a special case of the multiple occupants over time case. We argue that transforming the occupant dissatisfaction levels to a linear scale is very desirable.

We next (Section 4.1) consider the problem of evaluating overall occupant dissatisfaction for several occupants at a single timepoint (or time period). For multiple occupants, the notion of fairness comes into play. In Section 4.2 we consider the problem of evaluating overall occupant dissatisfaction for several occupants over a number of time-periods. The purpose of considering multiple timepoints is that it can be much easier to balance the occupant preferences over time, rather than for a single timepoint: we may be able to balance the mild discomfort felt by an occupant in one kind of scenario by what happens at other times. Section 5 concludes.

2 THERMAL COMFORT

Scales for Thermal Comfort: We want to predict how dissatisfied occupants are regarding thermal comfort in different environments. The standard ASHRAE seven-point thermal sensation scale [1] has integers -3 to $+3$ representing cold, cool, slightly cool, neutral, slightly warm, warm and hot, respectively. However, we are interested in occupant thermal comfort rather than thermal sensation, and they are not quite the same thing [2]. Because of this it is perhaps preferable to use labels that express thermal comfort more directly, such as: "Too cold", "slightly too cold", "a little cool", "comfortable", "a little warm", "too warm", "too hot". To allow expression of more extreme situations, we can add further points to the ends of the scale such as e.g., +4 for "much too hot", and +5 for "almost unbearably hot".

Predicting Thermal Comfort: Our approach relies on us being able to predict the thermal comfort of occupants based on different environmental conditions. Fanger's model [4] predicts the degree of a thermal sensation of an individual based on four physical variables and two personal variables: air temperature, air velocity, mean radiant temperature, relative humidity, clothing insulation and activity level. This mostly performs reasonably well [2, 7]. An alternative approach we have used is based on what the occupants have told us previously about how they felt in different situations. It appears that, at least in some cases, this can predict thermal sensation/comfort well, even if one only uses the air temperature data [14, 9].

3 OPTIMISING FOR A SINGLE OCCUPANT

Although our primary focus is multi-occupant rooms, it is helpful to first analyse the case of a single occupant.

3.1 Case of a Single Occupant for a Single Timepoint

Let us assume, at least for now, that we know, for any potential thermal conditions, the degree of thermal comfort for any occupant. The thermal conditions are measured by a collections of sensor readings, for example, including air temperature at different locations, humidity and so on.

Let $\vec{\theta}$ represent a vector of sensor readings. We are assuming we have a real-valued function H on all such vectors $\vec{\theta}$ (or at least on all feasible ones), that correctly predicts the degree of thermal comfort $H(\vec{\theta})$ for the occupant in situation $\vec{\theta}$. Thus if, for example, $H(\vec{\theta}) = 1$

then it is predicting that the occupant will be a little warm in the situation represented by $\vec{\theta}$.

Clearly, what we'd like to do is to control the system so as to generate conditions $\vec{\theta}$ that make $H(\vec{\theta})$ as close to 0 (fully comfortable) as possible.[1] However, we need to decide more precisely what it means for $H(\vec{\theta})$ to be close to 0. We want some function L which maps values of thermal comfort to degrees of dissatisfaction. For example, if we'd like the function to be symmetric about 0 (the y-axis) (i.e., an even function), we might choose $L(x) = x^2$ or $L(x) = |x|$, in which cases we will aim to optimise $H(\vec{\theta})^2$ or $|H(\vec{\theta})|$, respectively. In contrast, some occupants may dislike more being cold than being warm, leading to a function L that is not symmetric about the y-axis.

3.2 Case of Single Occupant over Time

It can be important to consider a longer term view than just optimising for a single timepoint (or time period, where we assume that each time period is the same length). We consider here the case of evaluating a collection of comfort scores (or, alternatively, dissatisfaction scores) for a single occupant at different timepoints. For example, suppose a particular control policy leads to thermal comfort/sensation levels scores 3 at 9am, 0 at 2pm and 0 at 4pm. A different control policy leads to 1 at 9am, 1 at 2pm and 1 at 4pm. Which of these two is better? They each have the same average comfort value, so at first sight one might consider them to be equivalent. However, the score of 3 may indicate the occupant being much too hot, possibly being very uncomfortable. Because of this, one can argue that a score of 3 is really much worse than three times as bad as a score of 1, with the latter indicating at most mild discomfort.

More generally, suppose that the occupant registers ASHRAE-scale (or thermal comfort) values over a number $x_1, \ldots, x_N$ of timepoints; we want to evaluate these to give an overall cost value. We want some function L that takes these sequence of thermal comfort values and generates a dissatisfaction score, which is on an additive scale—i.e., the combination of a collection of dissatisfaction scores is their sum. Thus the overall evaluation (dissatisfaction) of a sequence of comfort values $x_1, \ldots, x_N$ will then be defined to be $\sum_{i=1}^{N} L(x_i)$, so that for two sequences $(x_1, \ldots, x_N)$ and $(y_1, \ldots, y_N)$ if $x_1 + \cdots + x_N = y_1 + \cdots + y_N$ then the two sequences are equally adequate. We call L, a "linearising function".

We may have $L(-x_i) = L(x_i)$ but we might not. Also, the cost (negative utility) of, for example, "almost unbearably hot" can be very different for different occupants in different situations; in particular, it makes a major difference if the occupant can acceptably leave the room and find somewhere more comfortable.

3.2.1 Generating Linearising Function L

Standard elicitation procedures for utility [13] can be adapted to elicit values of the function L. For example we could ask queries such as: *For which value of* x *is* $(0, 0, x)$ *equivalent to* $1, 1, 1$*?* This could be used to specify $L^{-1}(3)$, given that we've defined $L(0)$ to be 0 and $L(1)$ to be 1.

Note that positive linear transformations of the scores make no essential difference, so that if L is an adequate linearising function, then so is $DL + E$, given by $(DL + E)(x) = DL(x) + E(x)$, where D and E are strictly positive real numbers. This allows us to normalise in some way the function L if we wish, by transforming L using such a linear transformation in order to ensure certain conditions are respected. Firstly, we can normalise L to ensure $L(0) = 0$. We could also, if we wished normalise to ensure that e.g., $L(1) = 1$; or e.g., $L(1.5) = 1$.

Candidate Functions

If we do not the have the opportunity to elicit values for L, a standard function can be used. A simple candidate function L is given by $L(x) = x^2$, with e.g., $L(2) = 4$, so a comfort value of 2 is judged to be four times as bad as a comfort value of 1. One possible family of functions, which can bias much more to the more extreme points of the scale, is that of the form: $L(x) = (a^{x^2} - 1)/(a - 1)$ for some $a \geq 0$ with $a \neq 1$, where we set $L(1) = 1$. L is continuous and $L(0) = 0$.For example, with $a = 1.2$, $L(2)$ is around 5.4 and $L(3) \approx 20.8$. For a close to 1, $L(x)$ tends to x^2. For $a > 1$, $L(x)$ grows faster than x^2; for $a < 1$, $L(x)$ grows slower than x^2.

3.2.2 Further Interpretation of Linear Scale

In summary, we should attempt to map user discomfort values to a linear scale. A further advantage of using a linear scale is when we are summing up previous data (or uncertainty about discomfort values) with an expected value. The latter doesn't make so much sense if the scale is not a linear one.

The transformed values, on the linear discomfort scale, can be considered as negative utility (in the sense of expected utility). One interpretation of such values is as the financial gain they would require in order to suffer this degree of discomfort (this relates somewhat to "Willingness to Pay", Section 3.8 of [13]). This financial interpretation might seem a little far fetched at first sight: there is unlikely to be any differential compensation paid to occupants according to their degrees of comfort. However, thinking about a home situation, the relevant kinds of comparisons are being made implicitly. If I am working at home on my own on a cold winter's day I will tend to adjust the heating so that I am slightly but not very cold. I am thus implicitly expressing a tradeoff between my comfort level and the cost of fuel: I am paying for a rate of fuel usage that transforms my thermal comfort level from too cold to slightly cool; but I'm implicitly not being prepared to pay for the extra amount of fuel to move my thermal comfort level to completely comfortable.

4 MULTIPLE OCCUPANTS

In this section, we are focusing on multiple occupants who share the same space, such as in the same room, or perhaps in adjoining rooms with thin partitions separating the rooms. This does not mean that the different occupants are experiencing the same thermal conditions; for example, the air temperature around one occupant may be 22^oC, but 20^oC for another occupant in the same large office, for example, if the former is receiving direct sunlight.[2] (Our approaches would apply also for multiple occupants in different rooms that are far apart; however, the method is not so relevant there, since for the latter situation the control system could treat the two occupants independently, with an action relevant for one of the occupants, such as turning on a heater in her room, not relevant for the other.)

[1] We may well also want to involve energy efficiency or energy usage in the objective function, which can be done by adding a separate term. We do not focus further on this issue in this paper.

[2] Collaborators on the ITOBO project [8] are currently performing experiments within two multi-occupant rooms in the Environmental Research Institute building at University College Cork, where various parameters including air temperature and lux (light) level are measured at different points in the room.

For each occupant in each situation let us again assume that we know what their dissatisfaction level is (we consider the case of where there is uncertainty briefly later in Section 4.4). We also assume that these dissatisfaction levels are on a linear scale, as described in Section 3.

4.1 Single Decision For Multiple Occupants

First we consider a situation where we are interested in optimising at a single timepoint (time period), for multiple occupants sharing the same space.

We would like to treat the occupants fairly. Firstly, we would like to regard them each as having equal importance (although later, in Section 4.3, we consider allowing different grades of importance). Secondly, we would like to bias towards having a more equitable range of degrees of discomfort. For example, in a two occupant office, we would prefer a situation where both occupants have degrees of dissatisfaction of 1 (on a linear scale) than one having degree of dissatisfaction of 0, and the other degree of dissatisfaction of 2.

4.1.1 Relative Scaling of Occupants for Equal Occupant Importance

The linearising functions L described in Section 3 were, for the purpose of optimising for a single occupant, non-unique, in that one can multiply the function L by a strictly positive real scalar, and get a function that performs equivalently. For multiple occupants, we will have such a scaling function L_i for each occupant i. We are considering a situation where the occupants are assumed to be of equal importance. We therefore need to scale the different functions L_i for each occupant so that they reflect equal importance.

We are assuming, as described above, that for any vector $\vec{\theta}$, representing the environmental conditions, a degree of discomfort/dissatisfaction $J_i(\vec{\theta})$, which is a non-negative real number. In particular, J_i is based on a function L_i that maps a thermal comfort level to a non-negative real number. In order to respect the requirement that the occupants have equal importance, we need a way of calibrating the functions L_i for different occupants. As mentioned above, we can ensure that $L_i(0) = 0$, i.e., that thermally neutral (comfortable) corresponds with a zero degree of dissatisfaction. We want to rescale the functions L_i to reflect equal importance.

One approach is to normalise the functions L_i in some way; for instance to ensure $L_i(1) = 1$; or alternatively, $L_i(2) = 1$; or $L_i(1.5) = 1$.

Alternatively, if the functions L_i can be given a financial interpretation, as the monetary value needed to compensate for the thermal discomfort, then this already gives a relative calibration/scaling of the functions L_i.

One might perhaps also try to take into account, in this relative scaling step, the relative "choosiness" of different occupants: some occupants will tend to give more extreme inputs, which will tend to have more impact on the objective function. One could decide to correct for this, effectively lessening the importance of such occupants, so that their inputs will tend to dominate less.

After the rescaling process, for each vector of environmental conditions we have a vector $(s_1, \ldots, s_K)$ of dissatisfaction values, one for each of the K occupants in the currently considered space.

4.1.2 Properties of Aggregation Operators

Our task is to sum the vector $(s_1, \ldots, s_K)$ of dissatisfaction values (one for each occupant) into a single non-negative real number that somehow represents the overall degree of (thermal) dissatisfaction for the group. We thus want to define some real-valued function G that sums up these K numbers into a real value $G(s_1, \ldots, s_K)$. Such an aggregation function G can be a kind of average or a generalised summation.

The topic of aggregation operators and their properties has been studied for a long time, at least since Cauchy in 1821 and Kolmogoroff and Nagumo in 1930 [12, 10]. We give some properties that are arguably natural for our problem.

The first property implies that the result is not affected by the identity of the occupant. It also implies that the occupants are equally important.

Symmetric: if $(t_1, \ldots, t_K)$ is a permutation of $(s_1, \ldots, s_K)$ then $G(t_1, \ldots, t_K) = G(s_1, \ldots, s_K)$.

The next two properties are very natural: if one increases the degree of dissatisfaction of any user then the overall dissatisfaction is increased (or at least not decreased, for the former property).

Increasing: $G(s_1, \ldots, s_K)$ is an increasing function of its arguments, i.e., if for all $i = 1, \ldots, K$, $s_i \leq t_i$ then $G(s_1, \ldots, s_K) \leq G(t_1, \ldots, t_K)$.

Strictly Increasing: $G(s_1, \ldots, s_K)$ is a strictly increasing function of its arguments, i.e., if for all $i = 1, \ldots, K$, $s_i \leq t_i$ and for some i, $s_i < t_i$, then $G(s_1, \ldots, s_K) < G(t_1, \ldots, t_K)$.

A small increase in input dissatisfactions should not cause a large jump (continuity). The second property is less clearly essential, but reflects the smoothness of the change in output as the input changes.

Continuous: G is a continuous function;

Continuously Differentiable: G is continuously differentiable

The following property from [6] represents the idea that if we bring the values s_i closer together without changing their sum then we improve the overall evaluation.

Transfer Principle: If $s_i < s_j$ and $\epsilon < s_j - s_i$ then $G(s_1, \ldots, s_K) > G(t_1, \ldots, t_K)$ where $t_i = s_i + \epsilon$, and $t_j = s_j - \epsilon$, and $t_k = s_k$ for $k \neq i, j$.

The following properties relate to the fact that the choice of the linearising functions L_i (see Section 3.2.1) would often be non-unique, in that a positive linear transformation of the same function could also do. The second is perhaps less important, since we are arranging that $L_i(0) = 0$. The two properties together are known as being *stable for positive linear transformations* [12].

Scaling: For $C > 0$,

$$G(Cs_1, \ldots, Cs_K) = C \times G(s_1, \ldots, s_K).$$

Uniform translation: For $D > 0$,

$$G(s_1 + D, \ldots, s_K + D) = G(s_1, \ldots, s_K) + D.$$

The following seems natural for an averaging operator:

Idempotence: $G(s, \ldots, s) = s$ [only for an averaging operator].

The next property relates to sum-like operators, and is a convenient associativity property:

Decomposable as binary operator: [only for sum operators] G can be decomposed as an associative binary operation $\oplus$: there exists associative binary operation $\oplus$ such that $G(s_1, \ldots, s_K) = s_1 \oplus \cdots \oplus s_K$.

Ordinary Summation and Mean

We can define the sum operator $G(s_1, \ldots, s_K) = s_1 + \cdots + s_K$, or the mean operator $G(s_1, \ldots, s_K) = \frac{1}{K}(s_1 + \cdots + s_K)$. This satisfies all the above properties except the Transfer Principle. However, for the sake of fairness, we want to ensure that the Transfer Principle is satisfied.

We go on the suggest three families of aggregation operators that might be used in this context, and briefly discuss their properties.

4.1.3 Ordered Weighted Average

One idea is to use a weighted average, where more weight is attributed to give extra weight to the more uncomfortable occupants. This kind of average is known as an ordering weighted averaging operator [16, 5]. If there are K occupants present then one uses positive decreasing weights $w_1, \ldots, w_K$ that sum to 1. The overall cost function is then equal to $w_1 s_{(1)} + \cdots + w_K s_{(K)}$, where $(s_{(1)}, \ldots, s_{(K)})$ is a permutation of $s_1, \ldots, s_K$ such that $s_{(1)} \geq \cdots \geq s_{(K)}$.

Since there may not be constant number of occupants in the same space, we need a sequence[3] $w_1, \ldots, w_K$ of such weights for each K.

We can generate a corresponding Sum operator, by multiplying through by K:

$$G(s_1, \ldots, s_K) = K w_1 s_{(1)} + \cdots + K w_K s_{(K)}$$

If we set $w_1 = 1$ and otherwise $w_i = 0$ we obtain the max operator. Also if we set w_i proportional to ϵ^i for some small positive number ϵ we get an operator that orders vectors in a similar way to the leximin ordering [3]. However, both of these give what might be considered as excessive weight to the most uncomfortable occupant.

For any weights vector, the corresponding aggregation operators are symmetric, increasing and continuous, and satisfy both Scaling and Uniform Translation. Idempotence is also satisfied for the averaging version. The operators are strictly increasing when all the weights are non-zero, and satisfies the Transfer Principle whenever all the weights are different.

It is typically not decomposable as a binary operator. The only other one of the above properties not satisfied by ordered weighted averages is being continuously differentiable. The derivative with respect to s_i at points when $s_i \neq s_j$ for all $j \neq i$, is one of the weights (specifically w_j such that $s_i = s_{(j)}$). The derivative is thus discontinuous (except if all the weights are equal).

4.1.4 Skewing of Linear Scale

Let λ be a continuously differentiable (strictly) increasing bijection on the non-negative reals.

We can define a Sum operator G_λ by

$$G_\lambda(s_1, \ldots, s_K) = \lambda^{-1}\Big(\sum_{i=1}^{K} \lambda(s_i)\Big).$$

An averaging operator can be defined similarly:

$$\lambda^{-1}\Big(\frac{1}{K}\sum_{i=1}^{K} \lambda(s_i)\Big).$$

This is known as a quasi-arithmetic mean [11].

[3] For different K, one would expect the sequences to be related in some way, so one might consider coherence conditions that relate to the weights as K varies; however, we do not consider this issue further in the current paper.

We are interested especially in cases where λ has a strictly increasing derivative, i.e., a positive second derivative (except possibly at 0) since then it satisfies the Transfer Principle. The smoothness properties are satisfied given suitably smooth λ (e.g., strictly positive derivative, except possibly at 0). G_λ doesn't generally satisfy Scaling and Uniform Translation. However, if we use $\lambda(x) = x^a$ for some $a > 1$ then Scaling is satisfied. The other properties are satisfied.

4.1.5 Mean-Plus-Spread Approaches

Another approach is to consider the value of vector $(s_1, \ldots, s_K)$ as consisting of two components: the first being the mean $\frac{1}{K}(s_1 + \cdots + s_K)$, and the second relating to the spread of the s_i's around the mean. For example, we can consider functions G of the form

$$G(s_1, \ldots, s_K) = \mu + R\sigma,$$

where R is a non-negative real number (which may depend on the number of occupants K), $\mu = \frac{1}{K}(s_1 + \cdots + s_K)$ is the mean of the K values, and σ is their standard deviation, so that $\sigma^2 = \frac{1}{K}\sum_{i=1}^{K}(s_i - \mu)^2$, which equals $\frac{1}{K}\sum_{i=1}^{K}(s_i)^2 - \mu^2$.

G is strictly increasing if we choose R such that $R < 1/\sqrt{K-1}$. It is not decomposable as a binary operator, but satisfies the other properties.

4.2 Multiple Occupants With Multiple Time Periods

We now consider a situation where we have multiple occupants in the shared space, for multiple time periods. We would like to generate an objective function that gives an overall cost (overall degree of dissatisfaction/undesirability).

We assume that the thermal sensation input for each occupant has been mapped to a linear scale of dissatisfaction, as in Section 3. We can therefore sum these degrees of dissatisfaction to get an overall degree of dissatisfaction for each occupant. Different occupants may be present different numbers of time periods, and it can be natural sometimes to explicitly take this into account. and it is important to take this into account, For each occupant we therefore then have a pair (s_i, N_i) representing the summed degree of dissatisfaction s_i and the number N_i of time periods in which they were present. Define s_i^* to be s_i/N_i, the mean degree of dissatisfaction for occupant i.

We therefore would like to generate a function F that takes as input a sequence

$$\Big((s_1, N_1), \ldots, (s_K, N_K)\Big),$$

representing the summarised inputs for the K occupants.

The three families of aggregation operators of Section 4.1 can be adapted for this task. A simple way to optimise for the overall function is, at each timepoint, to control for the environmental conditions so as to minimise $F\Big((s'_1, N'_1), \ldots, (s'_K, N'_K)\Big)$, where $\Big((s'_1, N'_1), \ldots, (s'_K, N'_K)\Big)$ corresponds to the inputs received so far. On the other hand, if we were to have information about the expected occupancy and dissatisfaction scores in future periods, then the optimisation technique could take these into account.

4.2.1 *Ordered Weighted Sum Approach*

The idea here is again to use a weighted sum/average, where higher weights are attributed to more dissatisfied occupants. So, we choose decreasing weights $w_1, \ldots, w_K$. We define the overall cost function F by

$$F\Big((s_1, N_1), \ldots, (s_K, N_K)\Big) = Kw_1 s_{(1)} + \cdots + Kw_K s_{(K)},$$

where $(s_{(1)}, \ldots, s_{(K)})$ is a permutation of $s_1, \ldots, s_K$ such that $s^*_{(1)} \geq \cdots \geq s^*_{(K)}$. Note that the weight assigned to the ith occupant is based on their mean degree of dissatisfaction, $s^*_i = s_i/N_i$, but the overall cost is based on their total degree of dissatisfaction s_i.

4.2.2 *Skewing of Linear Scale*

The approach from Section 4.1.4 can be applied directly. Again let λ be a function with the properties given in Section 4.1.4 (e.g., continuously differentiable strictly increasing bijection on the non-negative reals, which has a strictly increasing derivative). We define the overall cost function F_λ by

$$F_\lambda\Big((s_1, N_1), \ldots, (s_K, N_K)\Big) = \lambda^{-1}\Big(\sum_{i=1}^{K} \lambda(s_i)\Big).$$

Here the values N_i do not come into the definition.

As before, we can use, for example, λ of the form $\lambda(x) = x^a$ where $a > 1$.

4.2.3 *Mean-Plus-Spread Approaches*

The approach described in Section 4.1.5 can be adapted easily. We can consider a random variable which, for $i = 1, \ldots, K$, takes value s^*_i with chance $\frac{N_i}{N}$ where $N = \sum_{i=1}^{K} N_i$. Let μ be the mean of this random variable and σ^2 be the variance, so that

$$\mu = \sum_{i=1}^{K} \frac{N_i}{N} s^*_i = \frac{1}{N} \sum_{i=1}^{K} s_i,$$

and

$$\sigma^2 = \sum_{i=1}^{K} \frac{N_i}{N} (s^*_i - \mu)^2.$$

Again we can define the overall cost function to be $\mu + R\sigma$, choosing positive real R (which may depend on N).

4.3 Incorporating Importance

The approaches in Section 4.2 can easily be adapted to take an importance weight $v_i > 0$ into account for each occupant i. Each time period that occupant i spends in the space is treated as v_i time periods to give more emphasis to occupants with higher importance weight. We replace s_i by $v_i s_i$ and N_i by $v_i N_i$, and apply the equations in Sections 4.2.1 4.2.2 and 4.2.3 to obtain objective functions that bias according to the importance weights.

4.4 Taking Uncertainty into Account

There are many potential sources of uncertainty in our application. For instance, uncertainty about:

- what the current environmental conditions are, because of inaccuracy in sensing;
- the thermal sensation value the occupant will feel in any given conditions; this is true if we use a learning algorithm based on past inputs, or a PMV-based approach;
- each particular occupant's mapping from the thermal sensation scale to degrees of dissatisfaction;
- which occupants will be present.

One common and natural approach to dealing with this uncertainty is to use some kind of expected value, specifically for dissatisfaction given particular environmental conditions. This gives another reason to use a linear scale for dissatisfaction; if we compute expected value on a non-linear scale then the value will not necessarily adequately sum up the distribution.

Alternatively, we can generate probability distributions over the dissatisfaction levels of each occupant, giving a random variable for each occupant's dissatisfaction. The different methods described above for computing the values of an objective (cost) function can be extended for this probabilistic case, using, for example, a Monte-Carlo algorithm to estimate expected cost, if we assume the occupants' random variables are mutually independent.

5 CONCLUSION

The paper addresses the issue of how one evaluates a collection of thermal comfort inputs from multiple occupants and over time. This is important for defining an objective function for control, based on predicted responses of occupant under various environmental conditions.

We argue that it is important firstly to map the degrees of thermal comfort onto a linear scale of dissatisfaction, so that the values can be summed for the combination of several values for a single occupant. We have suggested three families of approaches for aggregating the dissatisfaction scores of several occupants, with different strengths and weaknesses. We have shown how this may be applied for the case of multiple occupants over many time periods. Although all three families seem quite natural, there are, of course, other approaches that should be explored.

There are other issues that could be considered for more sophisticated approaches. For instance, we assumed that the ordering of the sequential inputs was unimportant; however, for consecutive time-points this could make a difference to the overall dissatisfaction of an occupant, where, for example, starting off cold and slowly increasing heat would presumably be better than a more random ordering.

ACKNOWLEDGEMENTS

This material is based upon works supported by the Science Foundation Ireland under Grant No. 08/PI/I1912, and the ITOBO Strategic Research Cluster. I am grateful to the reviewers for their comments, and for discussions with many collaborators on the ITOBO project including Ken Brown, Anika Schumann, Conor Ryan, Damien Fay, Massimo Manna and Mateo Burillo.

REFERENCES

[1] ANSI/ASHRAE Standard 55-2004, 'Thermal environmental conditions of human occupancy', (2004).

[2] K.E. Charles, 'Fanger's thermal comfort and draught models', Technical Report IRC-RR-162, National Research Council of Canada: Institute for Research in Construction, (2003).

[3] D. Dubois, H. Fargier, and H. Prade, 'Refinements of the maximin approach to decision-making in fuzzy environment', *Fuzzy Sets and Systems*, **81**, pp. 103–122, (1996).

[4] P.O. Fanger, *Thermal Comfort: Analysis and Applications in Environmental Engineering*, McGraw-Hill, NY, 1972.

[5] J. Fodor, J.-L. Marichal, and M. Roubens, 'Characterization of the ordered weighted averaging operators', *IEEE Transactions on Fuzzy Systems*, **3**(2), 236–240, (1995).

[6] Christophe Gonzales, Patrice Perny, and Jean-Philippe Dubus, 'Decision making with multiple objectives using gai networks', *Artif. Intell.*, **175**(7-8), 1153–1179, (2011).

[7] M. A. Humphreys and J. F. Nicol, 'The validity of ISO-PMV for predicting comfort votes in every-day thermal environments', *Energy and Buildings*, **34**(6), 667–684, (2002).

[8] ITOBO, 'Information & communication technology for sustainable and optimised building operation (http://zuse.ucc.ie/itobo/)', *Cork, Ireland*, (2007-2012).

[9] C. Manna, N. Wilson, and K. Brown, 'Learning individual thermal comfort by robust locally weighted regression with adaptive bandwidth', in *ECAI-2012 Workshop: AI Problems and Approaches for Intelligent Environments*, (2012).

[10] J.-L. Marichal, *Aggregation Operators For Multicriteria Decision Aid*, Ph.D. dissertation, University of Liège Liège, Belgium, 1998/1999.

[11] J.-L. Marichal, 'On an axiomatization of the quasi-arithmetic mean values without the symmetry axiom', *Aequationes Mathematicae*, **59**(1-2), 74–83, (2000).

[12] J.-L. Marichal, P. Mathonet, and E. Tousset, 'Characterization of some aggregation functions stable for positive linear transformations', *Fuzzy Sets and Systems*, **102**(2), 293–314, (1999).

[13] R.L. Keeney H. Raifa, *Decisions with Multiple Objectives: Preferences & Value Tradeoffs*, Cambridge University Press, 1993.

[14] A. Schumann, M. Burillo, and N. Wilson, 'Predicting the desired thermal comfort conditions for shared offices', in *Proceedings of the International Conference on Computing in Civil and Building Engineering (ICCCBE-10)*, pp. 95–96, (2010).

[15] A. Wagner, E. Gossauer, C. Moosmann, Th. Gropp, and R. Leonhart, 'Thermal comfort and workplace occupant satisfaction–results of field studies in german low energy office buildings', *Energy and Buildings*, **39**(7), 758–769, (2007).

[16] Ronald R. Yager, 'On ordered weighted averaging aggregation operators in multicriteria decisionmaking', *IEEE Trans. Syst. Man Cybern.*, **18**(1), 183–190, (January 1988).

[17] R. Yao, B. Li, and J. Liu, 'The impact of office environments on employee performance: The design of the workplace as a strategy for productivity enhancement', *Building and Environment*, **44**, 2089–2096, (2009).