Upside-down Preference Reversal: How to Override Ceteris Paribus Preferences?

Ulrich Junker
ILOG S.A.
1681, route des Dolines
F-06560 Valbonne
ujunker@ilog.fr

Abstract

Specific preference statements may reverse general preference statements, thus constituting a change of attitude in particular situations. We define a semantics of preference reversal by relaxing the popular ceteris-paribus principle. We characterize preference reversal as default reasoning and we link it to prioritized Pareto-optimization, which permits a natural computation of preferred solutions. The resulting method simplifies elicitation, representation, and utilization of complex preference relations and may thus enable a more realistic preference handling in personalized decision support systems and in preference-based intelligent systems.

Introduction

As most real-world problems studied in Artificial Intelligence involve large solution spaces, preference knowledge is needed to guide a problem solver towards the preferred choices. Recent research on preference handling in AI has shown that preference knowledge can be represented in a declarative way as any other knowledge while adding new possibilities to knowledge representation and problem solving (cf. e.g. (Brafman *et al.* 2005)). Preference handling is important for multi-agent systems, combinatorial auctions, recommender systems, configuration, planning, and other AI tasks. In this paper, we consider preferences handling for combinatorial problems with multiple criteria. As it is the case for many real-world problems, we suppose that solutions satisfying the problem constraints can easily be enumerated and that the main difficulty is to address preferences formulated on a combinatorial criteria space.

Preferences on combinatorial domains have intensively been studied in multi-criteria decision making and multi-criteria decision analysis, but most approaches make strong assumptions about the user's preferences. The classic assumption is that the preference relation is a complete preorder, which permits a presentation of the user's preferences in terms of numeric utility functions. An even stronger assumption is preferential independence meaning that the preferences on a subset of the criteria are independent of the values of the other criteria. The popular multi-attribute utility theory supposes preferential independence for all subsets of the criteria and uses weighted sums of the criteria's utilities. Other methods such as Generalized Additive Independence use a relaxed form of preferential independence, but lead to utility functions that are more difficult to elicit (cf. e.g. (Boutilier *et al.* 2006)).

An alternative to utility functions is to directly represent and to use qualitative preference statements elicited from the user. Recent work in AI started a study of compact representations of preference relations over a combinatorial domain. Doyle et. al. have introduced ceteris-paribus preferences between propositional formulas (Doyle, Shoham, & Wellman 1991; McGeachie & Doyle 2004). Boutilier and Brafman et. al. have studied networks of conditional preference statements (Boutilier *et al.* 1999; 2004). There are numerous examples for conditional preferences. For example, we may prefer pop music to classic music when being at home and classic music to pop music when being at a reception.

Although conditional preferences exhibit one kind of preferential dependency, they do not capture all surprising phenomena in human choice behaviour. In this paper, we study another, may be less frequent, but even more surprising phenomenon, namely that of preference reversal. Preference reversal has been studied by psychologists in situations where trade-offs between two criteria need to be found (Tversky & Thaler 1990). Suppose that the user of a vacation recommender system prefers swimming to hiking and flat coasts to rocky coasts. When asked for a choice between swimming at a rocky coast and hiking at a flat coast, she may prefer the first choice. When asked which coast type would make her switch from swimming to hiking she may answer the flat coast. Hence, a change of the preference elicitation method causes a change of the trade-off.

In this paper, we investigate a more extreme form of preference reversal. We suppose that more specific preference statements can reverse more general statements. For example, we may prefer swimming to hiking and flat coasts to rocky coasts. The ceteris paribus principle then tells us that we prefer swimming at flat coasts to hiking at rocky coasts. However, we tell the system that we prefer hiking at rocky coast to swimming at flat coasts. This upside-down reversal of the ceteris-paribus preferences constitutes a change of attitude in a particular situation that presents new opportu-

nities. Hiking at rocky coasts, for example, offers an exciting panoramic view. More specific preference statements can thus override more general preference statements. A classic ceteris-paribus preferences between a and b imposes $ax \succ bx$ for all contexts x. We introduce reversible ceteris paribus preferences that impose the preference between ax and bx by default for all contexts x. However, if a more specific preference imposes $by \succ ay$ for some context y then the preference between a and b is overridden in this case.

We elaborate a precise semantics for preference reversal which defines the conditions in which a reversible ceteris paribus preference can be overridden. We can thus represent preference relations that do not obey the strong monotonicity property mentioned above and that capture more complex interactions between multiple criteria. As the ceteris paribus principle is used as a default rule, only exceptions need to be stated by the user, which permits a compact representation of a nonmonotonic preference relation. This reduces not only the burden of preference elicitation, but also improves the efficiency of optimization algorithms. We will show that the number of optimization steps for computing a single preferred solution is proportional to the number of preference statements.

We first introduce combinatorial problems with multiple criteria. We then review ceteris paribus preferences, before introducing the preference reversal semantics. We relate this new preference relation to prioritized Pareto-optimization in order to derive an optimization algorithm.

Decision Space and Criteria Space

We consider problems where a decision has to be chosen from a combinatorial space. Two decisions can be compared in terms of multiple criteria. We thus encounter a combinatorial decision space which is subject of constraints and a combinatorial criteria space which is subject of a preference relation. The criteria define a function from the decision space to the criteria space.

The decision space is specified in form of a finite set of constraints $\mathcal{C}$ that are formulated over a finite set of variables $\mathcal{X}$. Each variable x in $\mathcal{X}$ has a domain $D(x)$, which defines the possible value assignments to this variable. The domain $D(X)$ of a set X of variables is defined by the union of the domains $D(x)$ of all the variables x in X. An assignment to X is a function σ from X to $D(X)$ that maps each variable x to a value v from $D(x)$. The set of all assignments to X is called the *problem space* of X and we denote it by $\mathcal{S}(X)$. We often write an assignment in relational form $\{(x_1, v_1), \ldots, (x_n, v_n)\}$ or in logical form $x_1 = v_1 \wedge \ldots \wedge x_n = v_n$ to facilitate definitions and readability. As assignments are relations, they can easily be combined in terms of set operations.

A constraint c formulates restrictions on a subspace which is identified by a subset X_c of the variables. The restriction is expressed by a (finite) set R_c of the admissible assignments to the variables X_c. We can project an assignment σ over $\mathcal{X}$ to a subset X_c by defining $\sigma[X_c] := \{(x, v) \in \sigma \mid x \in X_c\}$. We say that σ satisfies the constraint c iff $\sigma[X_c]$ is an element of R_c. We say that an assignment in the problem space $\mathcal{S}(\mathcal{X})$ is a solution of $\mathcal{C}$ iff it satisfies all the constraints in $\mathcal{C}$. The solutions of this constraint satisfaction problem constitute the *decision space*.

The criteria space is specified by a finite set of criteria $\mathcal{Z}$. Each criterion z has an outcome domain $\Omega(z)$ and is a function from the problem space $\mathcal{S}(\mathcal{X})$ to $\Omega(z)$. We define the outcome domain $\Omega(Z)$ of a set of criteria Z as the union of the outcome domains of those criteria. The function can be formulated in terms of numerical or symbolic expressions over the variables x_j. Examples are linear expressions, piecewise linear expressions, minimum, and maximum. Given a solution σ of the constraints $\mathcal{C}$, we denote the value of the criterion z for this decision by $z(\sigma)$. We can also view the criteria as variables and say that a solution σ generates an assignment to $\mathcal{Z}$ which associates each element z of $\mathcal{Z}$ with its value $z(\sigma)$ under σ, i.e. $\mathcal{Z}(\sigma) := \{(z, z(\sigma)) \mid z \in \mathcal{Z}\}$. The *criteria space* is the set of all the assignments to $\mathcal{Z}$ which are generated by solutions. If there is no ambiguity, we also call these generated assignments solutions.

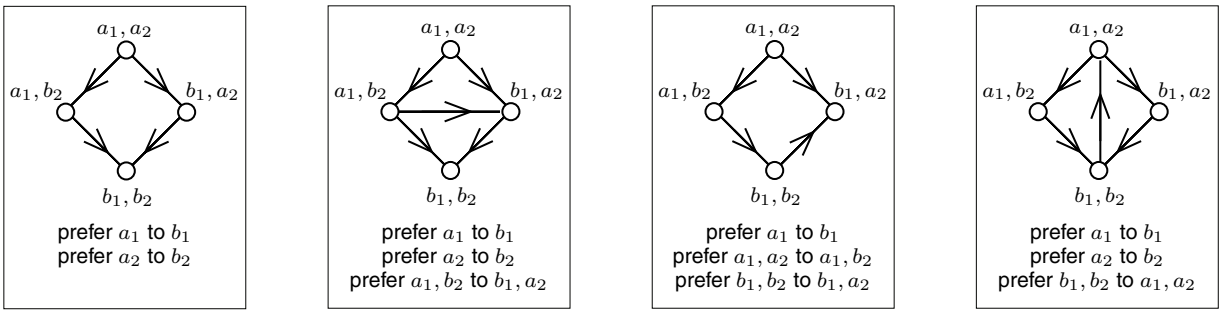
The user can compare two decisions in terms of the criteria values. We model the resulting preference order by a preorder on the *outcome space* $\mathcal{S}(\mathcal{Z})$ which contains all the assignments to $\mathcal{Z}$. A preorder $\succsim$ is a reflexive and transitive binary relation. If $\omega_1 \succsim \omega_2$ holds for two outcomes $\omega_1, \omega_2 \in \mathcal{S}(\mathcal{Z})$, then this means that the outcome ω_1 is at least as preferred as ω_2. The strict part $\succ$ of a preorder $\succsim$ contains all tuples (ω_1, ω_2) that satisfy $\omega_1 \succsim \omega_2$, but not $\omega_2 \succsim \omega_1$. We do not require that the preference order $\succsim$ is complete. It is thus possible that neither $\omega_1 \succsim \omega_2$, nor $\omega_2 \succsim \omega_1$ hold. This addresses cases where ω_1 and ω_2 are incomparable and cases where a preference between ω_1 and ω_2 is not known. In the second case, the considered preference relation may be refined when additional information is elicited.

A decision σ^* dominates a decision σ w.r.t. a preference order $\succsim$ iff $\mathcal{Z}(\sigma^*) \succsim \mathcal{Z}(\sigma)$ holds, but not $\mathcal{Z}(\sigma) \succsim \mathcal{Z}(\sigma^*)$, i.e. $\mathcal{Z}(\sigma^*)$ is better than $\mathcal{Z}(\sigma)$ w.r.t. the strict order $\succ$. A decision σ is a *preferred solution* w.r.t. a preference order $\succsim$ iff it is not dominated by any other decision.

Although the representation of an arbitrary preference order over a combinatorial space is doubly exponential, many interesting preference orders have a specific structure allowing a compact representation. This paper tries to identify one of those structures and thus addresses an emerging research topic (cf. e.g. (Lang 2004)).

Conjunctive Preference Statements

Although preference representations in form of numerical utility functions are ubiquitous in combinatorial optimization, there is a general agreement that users express their preferences in a qualitative form. For example, consider a (web-based) recommender system that proposes vacation packages to the user. It is a usual assumption that a decision support system can elicit the user's preferences by asking questions such as 'do you prefer vacation package A at least as much as vacation package B?'. If this question is posed for each pair A and B of possible decisions, then a complete preference order on the decision space can be acquired in this way. However, the choice of a vacation package may require a combination of more basic choices such



a) Preferences on values. b) Trade-off preference. c) Conditional preferences. d) Annihilation by reversal.

Figure 1: Ceteris Paribus Preference Orders.

as the vacation destination, the hotel chain, the time period, and different extras. As a consequence, the recommender system has to deal with a combinatorial decision space and the exhaustive elicitation is not practical in this case. Hence, the system will usually elicit an incomplete preference order which may be refined during further interactions.

Furthermore, as the decisions are complex objects and consist of many details, the user will not compare the decisions in their full extent, but base the comparison on criteria which are attributes of the decisions. For example, a user may compare two vacation packages in terms of the possible vacation activities, such as swimming, hiking, windsurfing, and the kind of the landscape, such as flat coast, rocky coast, and mountain. For example, the user may prefer a swimming vacation as least as much as a hiking vacation supposing that all other criteria are equal for the two vacations. Here, the user compares two possible values for one criterion and states a ceteris paribus preference between those values. We can write this statement as a preference between two assignments to the same criterion z:

$$\text{prefer } z = v \text{ to } z = w$$

Each such ceteris paribus preference statement defines a preference order on the outcome space. We simply complete the preference by an assignment to the other criteria. Then $\{(z, v)\} \cup \xi$ is preferred to $\{(z, w)\} \cup \xi$ and this for all assignments ξ to the criteria $\mathcal{Z} - \{z\}$. If multiple ceteris paribus statements are given, we take the union of their respective preference orders. Figure 1a shows the preference order for two ceteris paribus preference statements, namely a preference of swimming (a_1) to hiking (b_1) and a preference of flat costs (a_2) to rocky coasts (b_2). Since this order is generated for single-criterion preferences, it corresponds to the well-known notion of Pareto-dominance and the preferred solutions correspond to Pareto-optimal solutions.

Usually the set of Pareto-optimal solutions is quite large. It can be reduced by stating trade-off preferences between several criteria. For example, the user may state that she prefers swimming at rocky coasts (a_1, b_2) to hiking at flat coasts (b_1, a_2). Here, we consider two assignments which combine the best value for one criterion with the worst value of the other criterion. Such a trade-off preference refines the previous preference relation and also refines its strict part as illustrated by Figure 1b. As such, each preferred solution of the refined order is also a preferred solution of the original order. As a trade-off preference compares two assignments to the same criteria $z_1, \ldots, z_k$, we obtain preference statements that compare two conjunctions:

$$\text{prefer } z_1 = v_1 \wedge \ldots \wedge z_k = v_k \text{ to } z_1 = w_1 \wedge \ldots \wedge z_k = w_k$$

Let us now define a (finite) set $\mathcal{P}$ of ceteris paribus preference statements as follows. Each preference in $\mathcal{P}$ has the form (α, β) where α and β are two assignments to the same criteria set $Z \subseteq \mathcal{Z}$ and expresses a (weak) preference of α to β. The criteria set Z is called the *scope* of the preference. As we compare the same criteria, the all-else-equal condition thus concerns exactly the criteria in the complement $\mathcal{Z} - Z$ of the scope. Doyle and Wellman also consider more general forms of preference statements between propositional formulas. An example are importance preferences. If we prefer $z_1 = v_1$ to $z_2 = v_2$, then a decision supporting $z_1 = v_1$ is preferred to a decision supporting $z_2 = v_2$ and this independent of the value of z_2 in the first decision and the value of z_1 in the second decision. The investigation of importance preferences is beyond the scope of this paper.

It is important to note that the preferences in $\mathcal{P}$ can differ in their scope. We synthesize the local preferences in $\mathcal{P}$ to a global preference order over all the criteria. The standard way of doing this is to interpret the statements as ceteris paribus statements:

Definition 1 *(ceteris paribus) A preorder $\succsim$ on $\mathcal{S}(\mathcal{Z})$ is a* ceteris paribus order *for $\mathcal{P}$ iff $\alpha \cup \xi \succsim \beta \cup \xi$ holds for all statements $(\alpha, \beta) \in \mathcal{P}$ and for all assignments ξ to the complement of the scope of (α, β).*

The intersection of two ceteris paribus orders for $\mathcal{P}$ is also a ceteris paribus order for $\mathcal{P}$. Hence, there is a unique minimal ceteris paribus orders for $\mathcal{P}$. It is known that this minimal order can be generated from the ceteris paribus statements in the following way:

Proposition 1 *Let $\succsim_{cp}$ be the minimal ceteris paribus order and let $\alpha \succsim_{cp} \beta$. There exists an $n \geq 1$ and $\gamma_1, \ldots, \gamma_n$ such that $\alpha = \gamma_1$, $\beta = \gamma_n$, and for each $i = 1, \ldots, n-1$ there is a $(\alpha_i, \beta_i) \in \mathcal{P}$ and an assignment ξ_i to the complement of the scope of (α_i, β_i) s.t.*

$$\gamma_i = \alpha_i \cup \xi_i \succsim_{cp} \beta_i \cup \xi_i = \gamma_{i+1} \tag{1}$$

As observed in (McGeachie & Doyle 2004), conjunctive preferences are sufficiently expressive to represent conditional preferences (Boutilier *et al.* 1999) of the form:

$$\text{if } z_1 = v \text{ then prefer } z_2 = v_2 \text{ to } z_2 = w_2$$

It can be reformulated into a conjunctive preference between two assignments that agree in the value of z_1:

$$\text{prefer } z_1 = v \wedge z_2 = v_2 \text{ to } z_1 = v \wedge z_2 = w_2$$

Figure 1c shows the preference order for two conditional preferences for z_2 and a single-criterion preference for z_1.

The conjunctive ceteris paribus statements are more expressive than CP-nets. Indeed, any preorder over a combinatorial space can be represented by a (doubly exponential number of) conjunctive ceteris paribus statements.

However, conjunctive preference statements suffer from two problems which are due to overlapping scopes and preference reversal. Firstly, we consider two statements that have overlapping scopes:

$$\text{prefer } z_1 = v_1 \wedge z_2 = v_2 \text{ to } z_1 = w_1 \wedge z_2 = w_2$$
$$\text{prefer } z_2 = v_2 \wedge z_3 = v_3 \text{ to } z_2 = w_2 \wedge z_3 = w_3$$

We expect that $z_1 = v_1 \wedge z_2 = v_2 \wedge z_3 = v_3$ is preferred to $z_1 = w_1 \wedge z_2 = w_2 \wedge z_3 = w_3$ as the first assignment consists of the desired choices of the two statements and the other assignment consists of the non-desired choices of the two statements. To establish this, we need to find a sequence of 'improvement flips' that transforms the second assignment into the first assignment while applying a single preference in each flip. The first statement transforms $z_1 = w_1 \wedge z_2 = w_2 \wedge z_3 = w_3$ into $z_1 = v_1 \wedge z_2 = v_2 \wedge z_3 = w_3$, thus improving the values of z_1 and z_2. But now we cannot apply the second statement since z_2 has already the value v_2. Similarly, the first statement gets blocked when we first apply the second statement. If the two assignments $z_1 = v_1 \wedge z_2 = v_2 \wedge z_3 = w_3$ and $z_1 = w_1 \wedge z_2 = v_2 \wedge z_3 = v_3$ do not belong to the criteria space, then the ceteris paribus semantics will propose $z_1 = w_1 \wedge z_2 = w_2 \wedge z_3 = w_3$, which consists of the non-desired choices, as a preferred solution in addition to $z_1 = v_1 \wedge z_2 = v_2 \wedge z_3 = v_3$, which consists of the desired choices. As this result is counterintuitive, we advocate for a parallel application of several preference statements.

A parallel application of several statements is possible if these statements are compatible. We say that two assignments α_1 and α_2 are compatible iff $\alpha_1 \cup \alpha_2$ is a function, i.e. $(z, v_1) \in \alpha_1$ and $(z, v_2) \in \alpha_2$ implies $v_1 = v_2$. Two preference statements (α_1, β_1) and (α_2, β_2) in $\mathcal{P}$ are compatible iff α_1 and α_2 are compatible and β_1 and β_2 are compatible. A set P of preference statements is compatible iff any two preferences p_1 and p_2 from P are compatible.

Definition 2 *(parallel ceteris paribus) A preorder $\succsim$ on $\mathcal{S}(\mathcal{Z})$ is a* parallel ceteris paribus order *for $\mathcal{P}$ iff each compatible subset P of $\mathcal{P}$ satisfies*

$$\bigcup_{(\alpha,\beta)\in P} \alpha \cup \xi \succsim \bigcup_{(\alpha,\beta)\in P} \beta \cup \xi \qquad (2)$$

for each assignment ξ to the complement of the scope of P.

The intersection of two parallel ceteris paribus orders for $\mathcal{P}$ is also a parallel ceteris paribus orders for $\mathcal{P}$. Hence, there is a unique minimal parallel ceteris paribus order for $\mathcal{P}$. If an assignment is better than another one under the minimal parallel ceteris paribus order, then we find a sequence of parallel ceteris paribus flips:

Proposition 2 *Let $\succsim_{pcp}$ be the minimal parallel ceteris paribus order and let $\alpha \succsim_{pcp} \beta$. There exists an $n \geq 1$ and $\gamma_1, \ldots, \gamma_n$ such that $\alpha = \gamma_1$, $\beta = \gamma_n$, and for each $i = 1, \ldots, n-1$ there is a $P_i \subseteq \mathcal{P}$ and an assignment ξ_i to the complement of the scope of P_i s.t.*

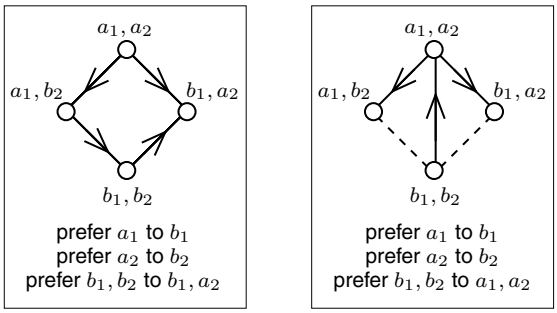
$$\gamma_i = \bigcup_{(\alpha,\beta)\in P_i} \alpha \cup \xi_i \succsim_{pcp} \bigcup_{(\alpha,\beta)\in P_i} \beta \cup \xi_i = \gamma_{i+1} \qquad (3)$$

Preference reversal is obtained if the (parallel) ceteris paribus order has a cycle. This means there are two different assignment ω_1 and ω_2 such that $\omega_1 \succsim \omega_2$ and $\omega_2 \succsim \omega_1$. The first preference is thus reversed by the second one. As a consequence, there is no strict preference between ω_1 and ω_2, meaning that both assignments may be optimal. Hence, the preference statements that generated $\omega_1 \succsim \omega_2$ and $\omega_2 \succsim \omega_1$ get annihilated. Annihilation can also occur in the extended semantics of CP-nets (Brafman & Dimopoulos 2004). We distinguish two cases of preference reversal. In the first case, we suppose that the cycle is caused by preferences of same scope. Hence, the cycle is already present in the statements given by the user. For example (Tversky & Thaler 1990) reports a reversal of trade-off preferences. Consider the preferences in Figure 1b and suppose that another elicitation method adds a preference of b_1, a_2 to a_1, b_2. Hence, the trade-off preferences annihilate themselves and we return to the strict order of Figure 1a. This case of preference reversal can only be addressed by further user interaction.

The other case of preference reversal is obtained if the statements differ in their scope. For example, suppose that the user prefers hiking at rocky coasts to swimming at flat coasts, although she prefers swimming to hiking in general and flat coasts to rocky coasts in general. The reason might be a nice panoramic view, which could be expressed by a third criterion if it were present. As a consequence the Pareto-dominance order is turned upside-down when comparing the best Pareto-optimal choice with the worst Pareto-optimal choice. This upside-down reversal leads to an annihilation of all stated preferences, meaning that all choices become optimal (cf. 1d). Indeed, the ceteris-paribus semantics expresses a strong monotonicity principle stating that a_1a_2 is preferred to b_1b_2 whenever a_1 is preferred to b_1 and a_2 is preferred to b_2. This monotonicity principle is a central axiom in multi-attribute utility theory (Le Huédé *et al.* 2006). Although its violation may be rare and represents a change of attitude in a particular situation, it is nevertheless legitimate from the stand-point of Artificial Intelligence, which seeks cognitive adequateness. Therefore, we explore a semantics for this upside-down preference reversal.

Preference Reversal

We now elaborate a semantics of preference reversal. If a cycle in the preference relation is caused by an interaction between preference statements of different scope, then we will



a) Conditional preference. b) Upside down reversal.

Figure 2: Reversible Ceteris Paribus Preference Orders.

give priority to the more specific statements. Since the user has explicitly stated that she prefers hiking at rocky coasts to swimming at flat coasts, we suppose that this statement overrides the inverse preference which is derived from the general preference for swimming and the general preference for flat coasts via the ceteris paribus principle. If we relax this principle and apply it only as a default rule, then we can avoid the preference annihilation shown in Figure 1d.

However, the ceteris-paribus principle should only be relaxed in certain cases. Consider a preference which expresses that $z_1 = w_1 \wedge z_2 = w_2$ is the desired choice:

$$\text{prefer } z_1 = w_1 \wedge z_2 = w_2 \text{ to } z_1 = v_1 \wedge z_2 = v_2$$

This desired choice can be put into question by two general preferences stating that v_1 is preferred to w_1, all else equal, and that v_2 is preferred to w_2, all else equal:

$$\text{prefer } z_1 = v_1 \text{ to } z_1 = w_1$$
$$\text{prefer } z_2 = v_2 \text{ to } z_2 = w_2$$

We protect the preferred choice of the specific preference against improvement flips based on less specific preferences. More generally, if an assignment γ is the desired choice of a preference of a scope Z, then only a preference of a scope that is a superset of Z may be used to flip γ. In all other cases, we block the second preference:

Definition 3 *(blocking) An assignment ω to $\mathcal{Z}$* blocks *a preference (α, β) in $\mathcal{P}$ iff there is a preference (γ, δ) in $\mathcal{P}$ s.t. β and γ are subsets of ω, β and γ have a non-empty intersection and β is not a superset of γ, i.e. $\gamma \not\subseteq \beta$.*

In our example, the general preference of a_1 to b_1 and of a_2 to b_2 are blocked in the state $b_1 b_2$ as shown in Figure 2b. However, we do not block the general preferences in other situations such as $a_1 b_2$ and $b_1 a_2$. When a preference statement (α, β) is blocked in an assignment $\beta \cup \xi$, then we do not prefer $\alpha \cup \xi$ to $\beta \cup \xi$. We use this relaxation to define reversible parallel ceteris paribus orders:

Definition 4 *(reversible parallel ceteris paribus) A preorder $\succsim$ on $\mathcal{S}(\mathcal{Z})$ is a* reversible parallel ceteris paribus order *for $\mathcal{P}$ iff each compatible subset P of $\mathcal{P}$ satisfies*

$$\bigcup_{(\alpha,\beta)\in P} \alpha \cup \xi \succsim \bigcup_{(\alpha,\beta)\in P} \beta \cup \xi \qquad (4)$$

for each assignment ξ to the complement of the scope of P such that no $(\alpha, \beta) \in P$ is blocked in $\bigcup_{(\alpha,\beta)\in P} \beta \cup \xi$.

The intersection of two reversible parallel ceteris paribus orders for $\mathcal{P}$ is also a reversible parallel ceteris paribus orders for $\mathcal{P}$. Hence, there is a unique minimal reversible parallel ceteris paribus order for $\mathcal{P}$. If an assignment is better than another one under this minimal order, then we find a sequence of parallel ceteris paribus flips where more specific statements precede more general statements:

Proposition 3 *Let $\succsim_{rcp}$ be the minimal reversible parallel ceteris paribus order and let $\alpha \succsim_{rcp} \beta$. There exists an $n \geq 1$ and $\gamma_1, \ldots, \gamma_n$ such that $\alpha = \gamma_1$, $\beta = \gamma_n$, and for each $i = 1, \ldots, n-1$ there is a subset P_i of the strict part of the reflexive transitive closure $\mathcal{P}^*$ of $\mathcal{P}$ and an assignment ξ_i to the complement of the scope of P_i s.t.*

$$\gamma_i = \bigcup_{(\alpha,\beta)\in P_i} \alpha \cup \xi_i \succsim_{rcp} \bigcup_{(\alpha,\beta)\in P_i} \beta \cup \xi_i = \gamma_{i+1} \qquad (5)$$

and for each $(\alpha_1, \beta_1) \in P_i$ s.t. $i < n-1$ there is a $(\alpha_2, \beta_2) \in P_{i+1}$ s,t. $\alpha_2 \subset \beta_1$.

The reversible ceteris paribus order agrees with the traditional order if all preference statements have disjoint scopes. Furthermore, it does not impact trade-off preferences. However, preference reversal impacts conditional preferences as it permits to simplify the representation of conditional preferences. One of the conditional preferences can be turned into a single-criterion preference which becomes the default (see Figure 2a). Conditional preferences then represent reversals of standard single-criterion preferences, which are the core of multi-attribute utility theory.

Preference Reversal and Pareto-Optimality

We now establish the correspondence between preference reversal and a prioritized form of Pareto-optimization (Junker 2002) by reformulating the problem. We can then compute a preferred solution by applying an existing optimization algorithm to the reformulated problem.

We add new dimensions to the criteria space. For each preference $(\alpha, \beta) \in \mathcal{P}$, we introduce a binary criterion y_α which has the outcome domain $\{0, 1\}$ and we define $\mathcal{Y}$ as the set of all those criteria. The meaning of these new criteria is as follows: whenever y_α has the value 1 in an assignment ω to $\mathcal{Z}^* := \mathcal{Y} \cup \mathcal{Z}$, then ω must contain α. However, we do not require the converse and permit that y_α has the value 0 when ω contains α. We consider only those assignments ω to $\mathcal{Z}^*$ that respect these constraints and we call those assignments *legal*. Hence, the new criteria space is defined by an extended set of criteria which we denote by $\mathcal{Z}^*$ and an extended set of constraints which we denote by $\mathcal{C}^*$. Furthermore, we associate the criterion y_α with a preference order $\succ_{y_\alpha}$ on the domain $\{0, 1\}$. As α represents a desired choice, we try to make it true whenever possible and therefore prefer the value 1 to the value 0 by defining $v \succ_{y_\alpha} w$ iff $v > w$. Hence, the binary criterion y_α behaves like a default rule $\frac{:\alpha}{\alpha}$. We define $\succ_z$ as the empty ordering for the criteria z in $\mathcal{Z}$.

Furthermore, we define an importance ordering $\succ_{\mathcal{Z}^*}$ on the criteria in $\mathcal{Z}^*$. We say that $y_\alpha \in \mathcal{Y}$ is more important than $y_\beta \in \mathcal{Y}$, i.e. $y_\alpha \succ_{\mathcal{Z}^*} y_\beta$, iff there is a preference (α, γ) in the strict part of the reflexive transitive closure $\mathcal{P}^*$ of $\mathcal{P}$

Algorithm PREFERENCEREVERSER($\mathcal{C}$, $\mathcal{Z}$, $\mathcal{P}$)

1. solve $\mathcal{C}$ and let σ be the result;
2. **if** $\sigma = \bot$ **then** return $\bot$;
3. let $Y := \{y_\alpha \mid (\alpha, \beta) \in \mathcal{P}\}$;
4. add legality constraints for Y to $\mathcal{C}$;
5. let $\mathcal{O}$ be the strict part of the
6. reflexive transitive closure of $\mathcal{P}$;
7. let $I := \{(y_\alpha, y_\beta) \in Y^2 \mid \exists(\alpha, \gamma) \in \mathcal{O} : \beta \subseteq \gamma\}$;
8. let $y_1, \ldots, y_n$ be the nodes of (Y, I)
9. in a topological order;
10. **for each** $i = 1, \ldots, n$ **do**
11. solve $\mathcal{C} \cup \{y_i = 1\}$ and let σ' be the result;
12. **if** $\sigma' \neq \bot$ **then** $\mathcal{C} := \mathcal{C} \cup \{y_i = 1\}$; $\sigma := \sigma'$;
13. return σ;

Figure 3: Computing a solution by preference reversal.

s.t. $\beta \subseteq \gamma$, and we say that $y_\alpha \in \mathcal{Y}$ is more important than $z \in \mathcal{Z}$, i.e. $y_\alpha \succ_{\mathcal{Z}^*} z$, iff z is in the scope of α. We now define a prioritized Pareto-order $\succ_{pp}$ on legal assignments:

Definition 5 *(Prioritized Pareto-ordering) Let σ_1 and σ_2 be two legal assignments to $\mathcal{Z}^*$. Then σ_1 is better than σ_2 w.r.t. the prioritized Pareto-order, i.e. $\sigma_1 \succ_{pp} \sigma_2$, iff $z(\sigma_1) \neq z(\sigma_2)$ for some $z \in \mathcal{Z}^*$ and for each $z \in \mathcal{Z}^*$ with $z(\sigma_1) \neq z(\sigma_2)$ one of the following condition holds:*

1. *$z(\sigma_1) \succ_z z(\sigma_2)$ or*
2. *there exists a $z^* \in \mathcal{Z}^*$ s.t. $z^* \succ_{\mathcal{Z}^*} z$ and $z^*(\sigma_1) \neq z^*(\sigma_2)$.*

According to the traditional Pareto-optimality, a solution can dominate a second solution only if all criteria have values in the dominating solution that are better than or equal to their values in the dominated solution. Prioritized Pareto-optimality permits that certain criteria may have worse values in the dominating solution iff there is a more important criterion which has a better value in the dominating solution. For example, $y_{z_1=v_1 \wedge z_2=v_2} = 1 \wedge y_{z_1=w_1} = 0$ dominates $y_{z_1=v_1 \wedge z_2=v_2} = 0 \wedge y_{z_1=w_1} = 1$ if the criterion $y_{z_1=v_1 \wedge z_2=v_2}$ is more important than $y_{z_1=w_1}$.

We now project the prioritized Pareto-dominance ordering $\succ_{pp}$ which is defined on the outcome space $\mathcal{S}(\mathcal{Z}^*)$ onto the original outcome space $\mathcal{S}(\mathcal{Z})$. We define $\omega_1 \succ_{pp}^{\mathcal{Z}} \omega_2$ for two assignments ω_1, ω_2 to $\mathcal{Z}$ if and only if there are legal assignments ω_1^* and ω_2^* to $\mathcal{Z}^*$ such that ω_1^* is a superset of ω_1, ω_2^* is a superset of ω_2, and ω_1^* dominates ω_2^* in the prioritized Pareto-ordering $\succ_{pp}$. We now state our main result:

Theorem 1 *The projection $\succ_{pp}^{\mathcal{Z}}$ of the prioritized Pareto-dominance ordering $\succ^{\mathcal{Z}}$ is equal to the strict part of the minimal reversible parallel ceteris paribus order $\succsim_{rcp}$.*

As a corollary, the preferred decisions of both orderings coincide. The proof of this theorem (which is omitted for the lack of space) exploits the particularities of the concepts that we have introduced. It makes use of the parallel application of ceteris paribus preferences. It relates the blocking notion to that of prioritization under the importance order.

Furthermore, the theorem leads to an immediate algorithm for computing one preferred solution under the reversible ceteris paribus ordering. As explained in (Junker 2006), we can find a Pareto-optimal solution by choosing a lexicographical ordering that extends the Pareto-dominance. We can find such an ordering even for prioritized Pareto-optimality since it suffices to consider a linearization of the importance ordering $\succ_{\mathcal{Z}^*}$ (which can be done by a topological sort) and to do a lexicographic optimization based on this linearization. The optimization steps for a binary criterion y just consist in checking whether the assignment $y = 1$ is consistent w.r.t. the other constraints. As the original criteria z have an empty preference order $\succ_z$, we need not perform an optimization step for them. The resulting algorithm is displayed in Figure 3. It returns $\bot$ if the constraints $\mathcal{C}$ have no solution. Otherwise, it returns a preferred solution under $\succ_{pp}$. Given n preference statements, it needs n calls of a constraint solver and there is much room for optimizing the algorithm in future work.

Conclusion

Generalized forms of ceteris-paribus preferences as studied in AI permit the representation of any preference relation over a combinatorial criteria space. However, the computation of this preference relation appears to be a difficult task according to the experiences reported in (McGeachie & Doyle 2004) and (Boutilier *et al.* 2004). Moreover, we have shown in this paper that the preference relation may lead to counterintuitive results if preferences overlap or if specific preferences reverse more general ones and annihilate all the preferences. We elaborated a new semantics that permits preference reversal. More specific preference statements can override more general preference statements. Preference reversal does not only model interesting phenomena in the interaction of multiple preferences, but also simplifies the representation of the well-studied conditional preference statements. However, the most surprising property is that preference reversal can be mapped to a prioritized form of Pareto-optimization and bears a natural lexicographical optimization algorithm. This algorithm does not require a computation of the preference relation, but applies the preferred choices of the preference statements starting with the most specific ones. This approach can easily be implemented on top of the system described in (Junker & Mailharro 2003). Hence, preference reversal simplifies the elicitation, representation, and utilization of complex form of preferences and thus promises a more realistic preference handling in personalized decision support systems such as recommender systems and preference-based intelligent systems.

Appendix: Reversing Search Preferences

In this appendix, we illustrate preference reversal by an example of reasoning as it occurs in AI research. We consider systematic search methods for solving combinatorial problems. The current wisdom prefers dynamic decision orderings over static decision orderings. Furthermore, constraint-based branching methods which either add a constraint (such as a variable-value assignment) or its negation to the original problem are preferred over 'fancy branching' methods that set up subproblems without adding a specific constraint to the solver. Forthcoming work (Junker 2007) shows

that these preferences may be reversed in certain circumstances, thus preferring fancy branching with static orderings to constraint-based branching with dynamic orderings. We summarize the essential ideas of this work.

Let C be a conjunction of propositional clauses formulated over the propositional variables $x_1, \ldots, x_n$. We investigate the problem of deciding the satisfiability of C. The approach in (Junker 2007) introduces *selection operators* on the set of clauses in order to decompose the satisfiability problem into disjoint subproblems. The *positive subproblem* of C is denoted by $\rho_i^+(C)$ and is the conjunction of all clauses from C that contain the literal $\neg x_i$. The *negative subproblem* of C is denoted by $\rho_i^-(C)$ and is the conjunction of all clauses from C that contain the literal x_i. The *neutral subproblem* of C is denoted by $\rho_i^0(C)$ and is the conjunction of all clauses in C that do not contain any of x_i and $\neg x_i$. The original problem C is equivalent to the conjunction of these subproblems and each clause occurs in exactly one subproblem. This decomposition is not obtained by adding a literal to the original problem and is thus 'fancy' as discussed above. If a subproblem is empty then it is the conjunction $\bot$ and we call it a *failing leaf problem*. A *projection operator* $\pi_{-i}(P)$ reduces each subproblem P by deleting the literals x_i and $\neg x_i$ from all clauses in P. The neutral subproblem $\rho_i^0(C)$ is equal to $\pi_{-i}(\rho_i^0(C))$. The positive subproblem $\rho_i^+(C)$ is equivalent to $x_i \Rightarrow \pi_{-i}(\rho_i^+(C))$ and the negative subproblem is equivalent to $\neg x_i \Rightarrow \pi_{-i}(\rho_i^-(C))$. If a subproblem contains the unit clause x_i or $\neg x_i$, the projection operator π_{-i} reduces this problem to the tautology $\top$. We call such a subproblem a *successful leaf problem*.

Furthermore, (Junker 2007) defines a family of anti-lexicographical orderings on vectors of truth values $\mathtt{T}, \mathtt{F}$. A value v is better than a value w iff v is equal to $\mathtt{T}$ and w is equal to $\mathtt{F}$. The lexicographic orderings are comparing the vectors starting from the end until they find an index with different truth values in the two vectors: $(v_1, \ldots, v_n) \succ_{1,\ldots,n}^{antilex} (w_1, \ldots, w_n)$ iff there is a k s.t. $v_k > w_k$ and $v_j = w_j$ for $j = k+1, \ldots, n$. In (Junker 2007), the lexicographical orderings are used to impose multiple lexicographical bound constraints

$$\bigwedge_{j=1}^{i} (u_{j,1}, \ldots, u_{j,j}) \succeq_{1,\ldots,j}^{antilex} (x_1, \ldots, x_j) \qquad (6)$$

in addition to the constraints of C. These bound constraints conserve intermediate results obtained from exploring the subproblems. Bound constraints on partial vectors $(x_1, \ldots, x_i)$ conserve the results of neutral subproblems. The bounds cannot be chosen arbitrarily. The idea is that bounds on larger vectors must be at least as tight as those on their subvectors. (Junker 2007) defines a *bound matrix* of size i as a tuple $u := (u_1, \ldots, u_i)$ of vectors $u_j := (u_{j,1}, \ldots, u_{j,j})$ that satisfy $(u_{j-1,1} \ldots, u_{j-1,j-1}) \succeq_{1,\ldots,j-1}^{antilex} (u_{j,1}, \ldots, u_{j,j-1})$ for $j = 2, \ldots, i$. A bound matrix can also be the inconsistency $\bot$. A particular bound matrix is the trivial bound matrix $u^{\mathtt{T}}$ which entirely consists of $\mathtt{T}$-bounds. Given a bound matrix u, we define $B(u)$ as the constraint (6) applied to u if the bound matrix is different to $\bot$. Otherwise, $B(u)$ is equal to $\bot$.

Algorithm $\mathtt{QuickLex}(C, i, u)$

```
1.  if u^C = ⊥ or C = ⊥ then return ⊥;
2.  if C = ⊤ then return u;
3.  u := min(u, u^C);
4.  while u ≠ u^C do
5.      u^C := u
6.      if u = ⊥ then return ⊥;
7.      u^0 := QuickLex(π_{-i}(ρ^0(C)), i − 1, π_{-i}(u))
8.      if u^0 = ⊥ then u^C := ⊥; return u^C;
9.      u := update(u, u^0)
10.     u^+ := ⊥;
11.     if u_{i,i} = T then
12.         u^+ := QuickLex(π_{-i}(ρ^+(C)), i − 1, π^+_{-i}(u))
13.     u^- := QuickLex(π_{-i}(ρ^-(C)), i − 1, π_{-i}(u))
14.     u := update(u, max(u^+, u^-));
15.     if u^+ ≠ ⊥ then u_i := (u^+_{i-1,1}, ..., u^+_{i-1,i-1}, T)
16.     else if u^- ≠ ⊥ then u_i := (u^-_{i-1,1}, ..., u^-_{i-1,i-1}, F)
17.     else u^C := u; return ⊥;
18. return u^C;
```

Figure 4:

(Junker 2007) introduces an algorithm called $\mathtt{QuickLex}$ that is supplied with a conjunction C of clauses over variables $x_1, \ldots, x_i$, the index i, and a bound matrix u of size i. If $C \wedge B(u)$ is unsatisfiable, then the algorithm returns $\bot$. Otherwise, it returns a bound matrix u' of size i, from which it is possible to extract a model of C.

$\mathtt{QuickLex}$ performs a set of operations on bound matrices. The maximum $\mathtt{max}(v, v')$ of two vectors v, v' of size i is the vector that is greater than or equal to the other one in the anti-lexicographical ordering $\succeq_{1,\ldots,i}^{antilex}$. Similarly, the minimum $\mathtt{min}(v, v')$ of two vectors v, v' of size i is the vector that is smaller than or equal to the other one in the anti-lexicographical ordering $\succeq_{1,\ldots,i}^{antilex}$. The maximum $\mathtt{max}(u, u')$ of two bound matrices $u := (u_1, \ldots, u_i)$ and $u' := (u'_1, \ldots, u'_i)$ is $(\mathtt{max}(u_1, u'_1), \ldots, \mathtt{max}(u_i, u'_i))$. The minimum $\mathtt{min}(u, u')$ is defined similarly. The maximum of a bound matrix u and $\bot$ is u. The minimum of a bound matrix and $\bot$ is $\bot$.

If $v := (v_1, \ldots, v_i)$ is a vector, then $\pi_{-i}(v) := (v_1, \ldots v_{i-1})$ is the vector obtained by removing the last element of v. If $u := (u_1, \ldots, u_i)$ is a bound matrix, then $\pi_{-i}(u) := (u_1, \ldots, u_{i-1})$ is the bound matrix obtained by removing the last vector from u. A variant is $\pi^+_{-i}(u) := (u_1, \ldots, u_{i-2}, \pi_{-i}(u_i))$ which removes the second last vector and adapts the size of the last vector. We define the update $\mathtt{update}(v, v')$ of a vector $v := (v_1, \ldots, v_i)$ of size i by a vector v' of size $i-1$ by taking the minimum $\mathtt{min}(v', (v_1, \ldots, v_{i-1})) = (w_1, \ldots, w_{i-1})$ and by adding v_i, i.e. $(w_1, \ldots, w_{i-1}, v_i)$. We define the update $\mathtt{update}(u, u')$ of a bound matrix $u := (u_1, \ldots, u_i)$ of size i by a bound matrix $u' := (u'_1, \ldots, u'_{i-1})$ of size $i-1$ as $(u'_1, \ldots, u'_{i-1}, \mathtt{update}(u_i, u'_{i-1}))$.

The algorithm $\mathtt{QuickLex}$ is initially called with the original problem $\mathcal{C}$, the number n of variables, and the trivial bound matrix $u^{\mathtt{T}}$ of size n. It then recursively decomposes

the problem into subproblems for x_n, x_{n-1}, and so on until reaching the leaf problems. We thus obtain a hierarchical decomposition tree. A node of this tree is identified by a conjunction C of clauses. There is at most one failing leaf for each constraint of the original problem. Hence, the number of failing leaves is bounded by the number m of constraints. Consequently, the number of inner nodes is bounded similarly. The algorithm maintains a bound matrix u^C for each inner node, thus avoiding that a failing leaf is explored twice. This bound matrix is initialized by a trivial bound matrix of adequate size. The space complexity for those bound matrices is $O(n^2 \cdot m)$.

Given a bound matrix u, the algorithm seeks to make it tighter. As failing leaves are not satisfiable, the algorithm returns $\bot$ in this case (line 1). Similarly, $\bot$ is returned if the bound matrix of the subproblem is $\bot$. As successful leaves are satisfied by any solution, the given bound cannot be tightened in such a case and is returned unchanged (line 2). In all other cases, the bound matrix of the subproblem is used to tighten the bound matrix u (line 3). If the result is equal to the bound matrix of the subproblem (line 4), then the subproblem will not reduce u further and the algorithm returns the result. Otherwise, the bound matrix u is tighter now than u^C, which is updated (line 5). If bound matrix u is equal to $\bot$ it cannot be tightened anymore (line 6). Otherwise, `QuickLex` is applied to the neutral subproblem in order to reduce $\pi_{-i}(u)$. If the neutral subproblem has no solution, the current problem has no solution either (line 8). Otherwise, its result is used to update u (line 9). If x_i may have a positive value (line 11), then the positive subproblem is solved under a reduced form of u (line 12). As the last element of u is valid if x_i is chosen, it is kept and the second last vector of u is removed when solving the positive subproblem. The negative subproblem is solved in all cases (line 13), but here the last vector of u is removed as it is not valid for the negative subproblem. The maximum of the results of both subproblems is used to update u (line 14). Furthermore, the last vector of u is updated by the positive subproblem (line 15) if it was successful or by the negative subproblem otherwise (line 16). If none of the subproblems was successful, $\bot$ is returned (line 17). The whole process is continued until the bound of the neutral subproblem agrees with the bound of the positive or of the negative subproblems.

The algorithm has the following properties. Let u' be the result of $\texttt{QuickLex}(C, i, u)$. Then $B(u')$ is a logical consequence of $B(u) \wedge C$. If u' is equal to $\bot$, this means that $B(u) \wedge C$ is unsatisfiable. If u' is different to $\bot$, let σ be an assignment of the variables to the truth values such that $\sigma(x_i) = u_{i,i}$ if $\pi_{-i}(u_i) = u_{i-1}$ and $\sigma(x_i) = \texttt{F}$ otherwise. Then σ is a model of $B(u) \wedge C$, meaning that $B(u) \wedge C$ is satisfiable. As the initial bound matrix is trivially satisfied, this means that algorithm `QuickLex` can decide the satisfiability of C.

Algorithm `QuickLex` clearly uses a problem decomposition based on a static ordering. Furthermore, it uses 'fancy branching' as the neutral subproblem neither imposes x_i, nor $\neg x_i$ for some literal x_i. Nevertheless, the complexity analysis in (Junker 2007) suggests that this combination of a static ordering and fancy branching is preferable to a dynamic ordering and constrained-based branching. Hence, the standard search preferences are reversed by this specific and unconventional combination of methods.

References

Boutilier, C.; Brafman, R.; Geib, C.; and Poole, D. 1999. Reasoning with conditional ceteris paribus preference statements. In *UAI*, 55–64.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. Preference-based constrained optimization with CP-nets. *Computational Intelligence* 20:137–157.

Boutilier, C.; Patrascu, R.; Poupart, P.; and Schuurmans, D. 2006. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence* 170(8-9):686–713.

Brafman, R., and Dimopoulos, Y. 2004. Extended semantics and optimization algorithms for CP-networks. *Computational Intelligence* 20:218–245.

Brafman, R.; Doyle, J.; Junker, U.; and Pu, P. 2005. Preference models and applications. IJCAI-2005 tutorials, IJCAI.

Doyle, J.; Shoham, Y.; and Wellman, M. P. 1991. A logic of relative desire (preliminary report). In Ras, Z., ed., *Sixth International Symposium on Methodologies for Intelligent Systems*, LNCS, 16–31. Berlin: Springer-Verlag.

Junker, U., and Mailharro, D. 2003. Preference programming: Advanced problem solving for configuration. *AI-EDAM* 17(1):13–29.

Junker, U. 2002. Preference-based search and multi-criteria optimization. In *AAAI*, 34–40.

Junker, U. 2006. Outer branching: How to optimize under partial orders? In *ECAI-06 Workshop on Advances in Preference Handling*, 58–64.

Junker, U. 2007. QUICKLEX(SAT): hierarchical lexicographic bound reduction for propositional satisfiability. Forthcoming.

Lang, J. 2004. Logical preference representation and combinatorial vote. *Ann. Math. Artif. Intell.* 42(1-3):37–71.

Le Huédé, F.; Grabisch, M.; Labreuche, C.; and Savéant, P. 2006. Integration and propagation of a multi-criteria decision making model in constraint programming. *Journal of Heuristics* 12(4-5):329–346.

McGeachie, M., and Doyle, J. 2004. Utility functions for ceteris paribus preferences. In *Computational Intelligence*, volume 20, 158–217. Blackwell Publishing.

Tversky, A., and Thaler, R. H. 1990. Anomalies: Preference reversals. *Journal of Economic Perspectives* 4:201–211.