Sorted-Pareto Dominance: an extension to Pareto Dominance and its application in Soft Constraints

Conor O'Mahony
Cork Constraint Computation Centre
University College Cork, Ireland
Email: c.omahony@4c.ucc.ie

Nic Wilson
Cork Constraint Computation Centre
University College Cork, Ireland
Email: n.wilson@4c.ucc.ie

***Abstract*—The Pareto dominance relation compares decisions with each other over multiple aspects, and any decision that is not dominated by another is called Pareto optimal, which is a desirable property in decision making. However, the Pareto dominance relation is not very discerning, and often leads to a large number of non-dominated or Pareto optimal decisions. By strengthening the relation, we can narrow down this non-dominated set of decisions to a smaller set, e.g., for presenting a smaller number of more interesting decisions to a decision maker. In this paper, we look at a particular strengthening of the Pareto dominance called Sorted-Pareto dominance, giving some properties that characterise the relation, and giving a semantics in the context of decision making under uncertainty. We then examine the use of the relation in a Soft Constraints setting, and explore some algorithms for generating Sorted-Pareto optimal solutions to Soft Constraints problems.**

I. Introduction

The notion of Pareto optimality originated in social welfare and economic theory, and the Pareto dominance relation is widely used in that field and many other related decision making fields. In a general decision-making context, a decision Pareto dominates another if it is strictly preferred in at least one aspect of the decision and at least as good in all other aspects [22, Ch. 2] (where an aspect could be: a voter in collective decision making, a state of the world in decision making under uncertainty [6], or a criterion in multi-criteria decision-making [14]). However, a problem with Pareto dominance is its lack of discriminatory power, as many comparisons between pairs of decisions do not result in dominance, which in turn leads to a large number of non-dominated or Pareto optimal solutions. Therefore, it is desirable to look at relations that extend Pareto dominance, where the extending relation has more power when comparing decisions, thus leading to a smaller set of non-dominated decisions that are all still Pareto optimal.

In this paper we look at an extension to the Pareto dominance relation, called Sorted-Pareto dominance. This relation is extended by the Leximin/Leximax relation [7], [20], which compares two decisions by lexicographically comparing the worst evaluations of the decisions. However, Leximin/Leximax places excessive emphasis on the worst evaluations (as it ignores the better evaluations when comparing two decisions, except if the decisions have the same worst evaluation) whereas Sorted-Pareto compares decisions considering all evaluations. Also, Sorted-Pareto assumes only a qualitative or ordinal scale, and therefore does not rely on quantitative information to compare decisions (as in, for example, the sum of weights approach, where decision evaluations are aggregated via summation and then compared).

The remainder of the paper is outlined as follows. Section II describes a simple decision making framework and the notion of Pareto dominance. Section III defines the Sorted-Pareto dominance relation, giving some properties that characterise the relation, and a semantics in terms of Weighted Constraint Satisfaction Problems in situations where there is uncertainty surrounding the weights. Section IV examines Sorted-Pareto dominance in the context of a general Soft Constraints problem framework, and describes a backtracking search algorithm and depth first branch and bound algorithms where (as well as a lower bound) an upper bound is used. Section V details our implementation of the Sorted-Pareto framework and the algorithms in our Soft Constraints solver, and discusses results obtained when solving particular instances of Sorted-Pareto problems using this implementation, where the use of an upper bound is shown to be helpful in some cases. Section VI briefly describe some extensions to the Sorted-Pareto dominance relation, with a view to motivating possible future work. Section VII highlights some similar work in this area, and Section VIII concludes with some discussion.

II. Preliminaries

In what follows, we describe a decision making setup where the objective is to minimise (e.g., costs), however, as usual, an alternative representation can be formulated where the objective is to maximise (e.g., utility). Let $\mathcal{A}$ represent a set of decisions, and let $\mathcal{S} = \{1, \ldots, m\}$, represent a finite set of decision aspects. Let α_i represent an evaluation, on a totally ordered scale T, of decision α in aspect i, where the scale T is ordered by $\leq$. This induces a relation $\preccurlyeq_i$ on $\mathcal{A}$ defined as, for all $\alpha, \beta \in \mathcal{A}$, $\alpha \preccurlyeq_i \beta$, if and only if, $\alpha_i \leq \beta_i$, i.e., α is as least as good as β in aspect i. Let $\prec_i$ represent the strict part of $\preccurlyeq_i$, (i.e., $\alpha \prec_i \beta$, if and only if, $\alpha \preccurlyeq_i \beta$ and $\beta \not\preccurlyeq_i \alpha$). Let $\equiv_i$ represent the indifference part of $\preccurlyeq_i$, (i.e., $\alpha \equiv_i \beta$, if and only if, $\alpha \preccurlyeq_i \beta$ and $\beta \preccurlyeq_i \alpha$). Therefore for each decision $\alpha \in \mathcal{A}$, we have a vector of evaluations over the m aspects, $(\alpha_1, \ldots, \alpha_m)$. Each decision is characterised by its vector of evaluations, and so for simplicity, we refer

to a decision α as meaning some $\alpha \in \mathcal{A}$ (i.e., some decision in the set of decisions) or as meaning a vector of evaluations corresponding to a decision (i.e., $\alpha = (\alpha_1, \ldots, \alpha_m)$).

For all $\alpha, \beta \in \mathcal{A}$, α Pareto dominates β, written as $\alpha \prec_P \beta$, if and only if (a) for all $i \in \{1, \ldots, m\}$, $\alpha \preccurlyeq_i \beta$, and (b) there exists $j \in \mathcal{S}$ such that $\alpha \prec_j \beta$ [22]. (For the weak version of the relation, $\preccurlyeq_P$, only condition (a) is required). A decision $\alpha \in \mathcal{A}$ is Pareto non-dominated if and only if, it is not Pareto dominated by any other decision, i.e., there is no $\beta \in \mathcal{A}$ such that $\beta \prec_P \alpha$. The set of these decisions, we will denote as $\text{OPT}_P(\mathcal{A})$.

III. Sorted-Pareto Dominance

In this section, we define the Sorted-Pareto dominance relation, which is an *extension* to the Pareto dominance relation, and based on an ordering defined in [15]. An extension $\preccurlyeq_*$ to some relation $\preccurlyeq$, is a relation such that, if α is preferred to β according to the original relation, then it is still preferred according to the extension, i.e., $\alpha \preccurlyeq \beta \Rightarrow \alpha \preccurlyeq_* \beta$. For these definitions, we assume decision making situations where the evaluations of the aspects are using the same scale, for example in the situation where the aspects correspond to different voters or experts, or even different criteria that use the same scale. However, even in situations where evaluations of aspects may not be on the same scale, then there are methods for modifying or normalising the different scales, e.g., using [15], so that the scales are commensurate. We also assume that each of the aspects are of equal importance, or in other terms, that the the ordering of the evaluation vector is irrelevant. This naturally occurs in decision making situations such as, group decision making where all experts are considered equal, or in decision making under uncertainty where there is no information on which state will occur.

To faciliate the definitions we introduce some additional notation: A *sorted permutation* $\alpha^{\uparrow}$ of a decision α is a reordering of the evaluations of the decision in ascending order, i.e., $\alpha^{\uparrow} = (\alpha_{(1)}, \ldots, \alpha_{(m)})$, such that, $\alpha_{(1)} \leq \ldots \leq \alpha_{(m)}$.

Definition 1. *For all $\alpha, \beta \in \mathcal{A}$, α Weak Sorted-Pareto dominates β, written as $\alpha \preccurlyeq_S \beta$, if and only if, $\alpha^{\uparrow} \preccurlyeq_P \beta^{\uparrow}$, i.e., $\alpha_{(i)} \leq \beta_{(i)}$, for all $i \in \{1, \ldots, m\}$. For all $\alpha, \beta \in \mathcal{A}$, α Sorted-Pareto dominates β, written as $\alpha \prec_S \beta$, if and only if, $\alpha^{\uparrow} \prec_P \beta^{\uparrow}$, i.e. $\alpha_{(i)} \leq \beta_{(i)}$, for all $i \in \{1, \ldots, m\}$ and there exists $j \in \{1, \ldots, m\}$ such that $\alpha_{(j)} < \beta_{(j)}$. Equivalently, Sorted-Pareto Dominance can be defined in terms of $\preccurlyeq_S$, as, $\alpha \prec_S \beta$, if and only if, $\alpha \preccurlyeq_S \beta$ and $\beta \not\preccurlyeq_S \alpha$.*

The following proposition gives an alternative characterisation of Weak Sorted-Pareto dominance. This relates the relation to the orderings defined in Definition 2.1 of [4] and Definition 11 of [19].

Proposition 1. *For all $\alpha, \beta \in \mathcal{A}$, α Weak Sorted-Pareto dominates β, if and only if, there exists a permutation σ of α, such that, $\sigma(\alpha) \preccurlyeq_P \beta$.*

A decision $\alpha \in \mathcal{A}$ is Sorted-Pareto non-dominated, if and only if, it is not Sorted-Pareto dominated by any other decision, i.e., there is no $\beta \in \mathcal{A}$ such that $\beta \prec_S \alpha$. We will denote this set as $\text{OPT}_S(\mathcal{A})$. Proposition 2(iii) below implies that the set of Sorted-Pareto non-dominated decisions $\text{OPT}_S(\mathcal{A})$ is a subset of the Pareto non-dominated decision $\text{OPT}_P(\mathcal{A})$, therefore we have a smaller set of solutions that are still Pareto optimal. Let us look at an example.

Example 1. *Consider a trivial group decision making problem, where the evaluation occurs of two decisions, $\mathcal{A} = \{\alpha, \beta\}$, by three different decision makers, $\mathcal{S} = \{1, 2, 3\}$, on the ordered scale $T = \{low, med, high\}$ (where of course $low \leq med \leq high$). Suppose decision α is evaluated as $(low, high, med)$ and decision β is evaluated as $(low, low, high)$. According to Pareto dominance, we see that $\alpha \not\prec_P \beta$, and $\beta \not\prec_P \alpha$, i.e., neither decision dominates the other, and therefore $\text{OPT}_P(\mathcal{A}) = \{\alpha, \beta\}$ (so either could be chosen as the actual decision). Now, if we use Sorted-Pareto dominance instead of Pareto dominance, then the sorted permutations of the decisions are $\alpha^{\uparrow} = (low, med, high)$ and $\beta^{\uparrow} = (low, low, high)$, and we can see that $\beta^{\uparrow} \prec_P \alpha^{\uparrow}$, i.e., $\beta \prec_S \alpha$, i.e., β dominates α using the Sorted-Pareto relation. Therefore, in this instance, β is the only non-dominated decision, i.e., $\text{OPT}_S(\mathcal{A}) = \{\beta\}$, and therefore it could make sense to choose decision β over α.*

A. Properties of Sorted-Pareto dominance

We now look at some general properties of the Weak Sorted-Pareto and Sorted-Pareto dominance relations, followed by properties of the relations when viewed as a ordering on multisets.

Proposition 2.

(i) $\preccurlyeq_S$ is reflexive (i.e., for all $\alpha \in \mathcal{A}$, $\alpha \preccurlyeq_S \alpha$) and transitive (i.e., for all $\alpha, \beta, \gamma \in \mathcal{A}$, if $\alpha \preccurlyeq_S \beta$ and $\beta \preccurlyeq_S \gamma$, then $\alpha \preccurlyeq_S \gamma$), therefore it is a preorder.

(ii) $\prec_S$ is irreflexive (i.e., for all $\alpha \in \mathcal{A}$, $\alpha \not\prec_S \alpha$) and transitive.

(iii) $\preccurlyeq_S$ (resp., $\prec_S$) extends the weak Pareto (resp., Pareto) ordering, i.e., for all $\alpha, \beta \in \mathcal{A}$, if $\alpha \preccurlyeq_P \beta$ (resp., $\alpha \prec_P \beta$), then $\alpha \preccurlyeq_S \beta$ (resp., $\alpha \prec_S \beta$).

(iv) Let $\preccurlyeq_R$ be some ordering on $\mathcal{A}$ that extends the weak Pareto ordering $\preccurlyeq_P$. Define relation $\preccurlyeq$ on $\mathcal{A}$ by, for all $\alpha, \beta \in \mathcal{A}$, $\alpha \preccurlyeq \beta \iff \alpha^{\uparrow} \preccurlyeq_R \beta^{\uparrow}$. Then $\preccurlyeq$ extends $\preccurlyeq_S$.

Part (iv) above, which follows immediately from the definitions, implies that Sorted-Pareto dominance is extended by the Leximin and Leximax [7], [20] total preorders. Sorted-Pareto is also extended by generalized Lorenz dominance, and by Ordered Weighted Averages [11], which are other refinements of Pareto dominance, however unlike these relations Sorted-Pareto can be used for purely ordinal or qualitative information aggregation.

Sorted-Pareto on multisets and its properties: We show how the Sorted-Pareto ordering can be characterised, in particular, when viewed as an ordering on multisets.

Let us say that an ordering $\preccurlyeq$ on $\mathcal{A}$ is *unaffected by permutations* if, for any two decisions α and β, and for any

two permutations σ and σ' of the decision aspects, $\alpha \preccurlyeq \beta \iff \sigma(\alpha) \preccurlyeq \sigma'(\beta)$. In this case, $\preccurlyeq$ can be represented as an ordering on M^T, the set of multisets of T, rather than vectors of T. Orderings that are unaffected by permutations are important, for instance, in the context of soft constraints, where the ordering of the soft constraints is taken to be irrelevant (so we can view the input as being a multiset of soft constraints). Its definition immediately implies that the Sorted-Pareto is unaffected by permutations. In fact, it can be easily seen that it is the (setwise) smallest such relation that extends the Pareto ordering.

In the remainder of Section III-A we consider the induced Sorted-Pareto ordering on M^T. If A is a multiset of T then we say that vector α, of k values in T, is *compatible with* A if A is the multiset $\{\alpha_i : i = 1, \ldots, k\}$. We have that $A \preccurlyeq_S B$ if and only if there exists some vector α compatible with A and vector β compatible with B such that α weak Pareto dominates β. We also define $\emptyset \preccurlyeq_S \emptyset$. Sorted-Pareto $\preccurlyeq_S$ is a partial order on M^T.

Consider the following two properties for orderings of M^T, where x and y are arbitrary elements of T, and A, B and C are arbitrary T-multisets. The first property relates the ordering on T with the ordering on singleton multisets. The second is a kind of independence ($\uplus$ is multiset sum).

(1) $x \leq y \Rightarrow \{x\} \preccurlyeq \{y\}$.
(2) $A \preccurlyeq B \Rightarrow A \uplus C \preccurlyeq B \uplus C$.

Proposition 3. *The Sorted-Pareto ordering $\preccurlyeq_S$ on multisets of T is the unique minimal preorder on M^T satisfying (1) and (2). That is, $\preccurlyeq_S$ satisfies (1) and (2), and if $\preccurlyeq$ is an ordering of multisets of T satisfying (1) and (2) and A and B are both T-multisets then $A \preccurlyeq_S B \Rightarrow A \preccurlyeq B$.*

Another characterisation of Sorted-Pareto is given by Property (3) below, where A and B are arbitrary multisets in T. Proposition 4 shows that the Sorted-Pareto ordering is the unique ordering (that only compares multisets of equal cardinality) satisfying (3).

(3) $A \preccurlyeq B$, if and only if,
$\min(A) \leq \min(B)$ and $A - \{\min(A)\} \preccurlyeq B - \{\min(B)\}$.

Proposition 4. *Let $\preccurlyeq$ be an ordering on M^T that only compares multisets of equal cardinality, i.e., such that $A \preccurlyeq B \Rightarrow |A| = |B|$. Then $\preccurlyeq$ satisfies Property (3) if and only if $\preccurlyeq = \preccurlyeq_S$.*

B. A Semantics for Sorted-Pareto

We now look at a semantics for the Sorted-Pareto dominance. Firstly, as given in Section II, this semantics assumes some totally ordered qualitative or ordinal scale T, which is ordered by $\leq$, and each decision $\alpha \in \mathcal{A}$ to be characterised by its vector of m evaluations, where each evaluation is on the scale T.

One way of comparing decisions using these evaluations is to map the qualitative scale values onto quantitative values, for example, representing some sort of cost. To do this, one can define a weights function f on the scale values, e.g., $f : T \to \mathbb{R}^+$, where the function is monotonic with respect to the ordering of the scale, i.e., for all $v, v' \in T$, $v \leq v' \Leftrightarrow f(v) \leq f(v')$. Therefore, for our set of decisions $\mathcal{A}$, and using some weights function f, the decisions can be compared and ordered using an order relation $\leq_f$, which is given by the sum of the m weights, i.e., for some $\alpha, \beta \in \mathcal{A}$, $\alpha \leq_f \beta$, if and only if: $\sum_{i=1}^m f(\alpha_i) \leq \sum_{i=1}^m f(\beta_i)$. This order relation $\leq_f$ is a total preorder on decisions, but for different mappings (i.e., different f), the resulting orders could be different.

TABLE I
RESULTING SUM OF WEIGHTS FOR f_1 AND f_2.

$\mathcal{A}$	$\sum_{i=1}^m f_1(v_i)$	$\sum_{i=1}^m f_2(v_i)$
$\alpha = (low, low)$	4	2
$\beta = (low, med)$	5	5
$\gamma = (low, high)$	8	6
$\delta = (med, med)$	6	8
$\epsilon = (med, high)$	9	9
$\sigma = (high, high)$	12	10

For example, Table I shows the resulting sum of weights when two functions f_1 and f_2 are applied to a set of decisions $\mathcal{A}$, where f_1 is defined by $f_1(high) = 6$, $f_1(med) = 3$, and $f_1(low) = 2$, and f_2 is defined by $f_2(high) = 5$, $f_2(med) = 4$, and $f_2(low) = 1$). Both f_1 and f_2 are monotonic w.r.t. to the scale but the resulting orders given by the sum of weights are different (i.e., $\delta \leq_{f_1} \gamma$, but $\gamma \leq_{f_2} \delta$).

When it is possible to provide a weights function (like f_1 or f_2) to map the scale values to some quantitative measure, then it is easy to compare decisions by using the sum of these weights. However, sometimes it is not possible to create this quantitative mapping, e.g., when this information is not available or is uncertain, so we can consider a different order that does not rely on this quantitative information. If we consider all possible weights functions $f : T \to \mathbb{R}^+$ (such that f is monotonic with respect to the ordering of T), then we can define an order relation $\leq_F$ on $\mathcal{A}$ as:

$$\forall \alpha, \beta \in \mathcal{A},\ \alpha \leq_F \beta \Leftrightarrow \alpha \leq_f \beta,\ \forall f \text{ monotonic w.r.t. } T$$

This relation is the intersection of all possible order relations $\leq_f$, for all monotonic functions f defined on T. By Theorem 1, this is equal to the Weak Sorted-Pareto order $\preccurlyeq_S$.

Theorem 1. *$\leq_F$ is equal to the Weak Sorted-Pareto order $\preccurlyeq_S$.*

Proof: First we show that for all $\alpha, \beta \in \mathcal{A}, \alpha \preccurlyeq_S \beta \Rightarrow \alpha \leq_F \beta$. Assume $\alpha \preccurlyeq_S \beta$. This implies, by definition of $\preccurlyeq_S$, $\alpha^\uparrow \preccurlyeq_P \beta^\uparrow$, which means that $\alpha_{(i)} \leq \beta_{(i)}$, for all $i \in \{1, \ldots, m\}$. Therefore, for any f, $\sum_{i=1}^m f(\alpha_{(i)}) \leq_f \sum_{i=1}^m f(\beta_{(i)})$. Since $\alpha^\uparrow$ and $\beta^\uparrow$ are permutations of α and β respectively, then $\sum_{i=1}^m f(\alpha_{(i)}) \leq_f \sum_{i=1}^m f(\beta_{(i)}) \Leftrightarrow \sum_{i=1}^m f(\alpha_i) \leq \sum_{i=1}^m f(\beta_i)$, which implies, for any monotonic function f, $\alpha \leq_f \beta$. Since this is true for any f, then $\alpha \leq_F \beta$.

Now we show that for all $\alpha, \beta \in \mathcal{A}$, $\alpha \not\preccurlyeq_S \beta \Rightarrow \alpha \not\leq_F \beta$. Assume $\alpha \not\preccurlyeq_S \beta$. This implies, by definition of $\preccurlyeq_S$, $\alpha^\uparrow \not\preccurlyeq_P \beta^\uparrow$. Therefore, $\exists i \in \{1, \ldots, m\}$ such that $\alpha_{(i)} \not\leq \beta_{(i)}$. We can construct a monotonic function f such that $\alpha \not\leq_f \beta$, and therefore $\alpha \not\leq_F \beta$. For instance, assign f such that $f(v) =$

0 if $v < \alpha_{(i)}$, and $f(v) = 1$ otherwise. Hence $f(\alpha_{(j)}) = 1$ for all $j \geq i$ and $f(\beta_{(j)}) = 0$ for all $j \leq i$. Therefore, $\sum_{j=1}^{m} f(\alpha_{(j)}) \geq m - i + 1$ and $\sum_{j=1}^{m} f(\beta_{(j)}) \leq m - i$, so $\sum_{j=1}^{m} f(\alpha_{(j)}) > \sum_{j=1}^{m} f(\beta_{(j)})$. Hence, $\exists f$ such that $\alpha \not\preceq_f \beta$, which shows $\alpha \not\preceq_F \beta$. ■

This gives a semantics to Sorted-Pareto, as a relation that can be used in decision making situations where there may only be ordinal or qualitative information available, and it provides an ordering that is consistent with any possible weights function selected to map an ordinal scale to a numerical one. It can be viewed as a more cautious representation than a weighted constraints one, and it can be applied in many of the application areas of weighted constraints. The weighted constraints formalism assumes that the costs are on an additive scale, where the cost of A and B is the sum of the costs of A and B; however, in many situations this can be questionable. For example, suppose one is using a weighted constraints solver to find a most probable explanation (MPE problem) [18]. Elicitation of probabilities can be problematic and unreliable, so instead of taking the elicited values at face value, one considers them as just representing the ordering between the probabilities. In this case, Theorem 1 shows that Sorted-Pareto represents the order relation that all compatible probability assignments agree with.

IV. Sorted-Pareto in Soft Constraints

Soft Constraints [3][21, Ch. 9] can be used to model many real-world problems when there is a need to specify preferences on particular aspects of the problem solutions. A soft constraint associates a preference degree to an assignment of a set of decision or problem variables. All the preference degrees of an assignment given by the soft constraints can be combined to give the overall preference degree of the assignment, with the aim to ordering these assignments and obtaining a set of optimal solutions. In this section, we look at a Soft Constraints framework for Sorted-Pareto, and provide some algorithms for finding Sorted-Pareto optimal solutions to Soft Constraints problems.

A. A General Framework for Soft Constraints

Here we construct a general framework for Soft Constraints problems, and create a Sorted-Pareto instance of the framework. Firstly, a Preference Degree Structure (PDS) (based on [8]) is a tuple $\mathcal{P} = \langle I, \otimes, \preceq \rangle$, where I is a set of preference degrees, $\preceq$ is a partial order on I, and $\otimes$ is a commutative and associative operator, monotonic with respect to $\preceq$ (i.e., $a \preceq b \Rightarrow a \otimes c \preceq b \otimes c$) which is used to combine the preference degrees. We define a general Soft Constraints problem (based on [25]) to be a tuple $\mathcal{F} = \langle \mathcal{V}, \mathcal{D}, \mathcal{C}, \mathcal{P} \rangle$, where $\mathcal{V}$ is a set of problem variables, $\mathcal{D}$ is a set of variable domains, $\mathcal{C}$ is a multiset of soft constraints, and $\mathcal{P}$ is a PDS, and we describe this Soft Constraints problem briefly as follows.

Let $\mathcal{D}(X)$ denote the domain of variable $X \in \mathcal{V}$. An assignment is a tuple representing a mapping of some set of problem variables to values in their domains, i.e., an assignment is a tuple in $D(U)$, for some $U \subset \mathcal{V}$, where $D(U) = \prod_{X \in U} \mathcal{D}(X)$. A complete assignment is a tuple such that all problem variables have been assigned, i.e., some tuple in $D(\mathcal{V})$. A soft constraint $c \in \mathcal{C}$ is a mapping from a tuple defined on a set of variables V_c (known as the scope of the constraint) to a preference degree, so, $c : \mathcal{D}(V_c) \to I$. Thus, for some assignment u, each soft constraint $c \in \mathcal{C}$ is applied to the subtuple of the assignment corresponding to the scope of the constraint, i.e., $c(u^{\downarrow V_c})$, (where $u^{\downarrow V_c}$ is the projection of assignment u to the variables in V_c), and we abbreviate this to $c(u)$. The overall preference degree $\rho(u)$ of an assignment u is the combination of all the preference levels of all constraints, i.e. $\rho(u) = \bigotimes_{c \in \mathcal{C}} c(u)$. An optimal assignment u is a complete assignment such that there is no other complete assignment u' such that $\rho(u')$ is preferred to $\rho(u)$, i.e., there is no other assignment u' such that $\rho(u') \preceq \rho(u)$.

For the Sorted-Pareto instance of this problem, we use a Preference Degree Structure defined as follows. The set of preference degrees is the set of multisets of T, i.e., $I = M^T$, the combination operator is *multiset sum*, i.e., $\otimes = \uplus$, and the preference relation is the Sorted-Pareto ordering, i.e., $\preceq = \preccurlyeq_S$, extended in the obvious way to multisets in T of the same cardinality. So the overall preference degree $\rho(u)$ of an assignment u is $\rho(u) = \biguplus_{c \in \mathcal{C}} c(u) = \{c(u) : c \in \mathcal{C}\}$, i.e., the overall preference degree is given by the multiset containing all the preference degrees. An optimal assignment is one such that there is no assignment u' such that $\rho(u') \prec_S \rho(u)$.

B. Solving Soft Constraints problems

We now look at some algorithms for searching for a set of non-dominated solutions to a Soft Constraints Problem. We also assume that there is a set of hard constraints, $\mathcal{C}'$, i.e., constraints which allow or disallow certain tuples of domain values, so we extend our Soft Constraints Problem definition to include this set $\mathcal{C}'$. Therefore a Soft Constraints problem is a tuple $\mathcal{G} = \langle \mathcal{V}, \mathcal{D}, \mathcal{C}, \mathcal{C}', \mathcal{P} \rangle$.

BruteSearch *algorithm*

In a standard backtracking or depth first search, each problem variable in turn is chosen and assigned a value from its domain. The domains of the variables are updated and inconsistent values are removed, in order to maintain some form of consistency with the set of hard constraints. If a variable has no legal values left in its domain, then the search will backtrack. The algorithm also maintains a set of non-dominated or "optimal" assignments, and once a complete consistent assignment is encountered, the algorithm will compare the preference level of this assignment (given by the soft constraints) with any previously found non-dominated complete assignments, and update the set of non-dominated assignments if this new complete assignment is non-dominated. This algorithm, as described, can be considered as a 'brute force' approach to the soft constraints, so we label this algorithm BruteSearch.

DFBBSearch *algorithm*

In a depth first branch and bound (DFBB) search, which is a standard improvement on the simple backtracking search,

instead of only testing if complete assignments are dominated by the set of non-dominated solutions, the algorithm will also check to see if partial assignments are dominated. It does this as follows. First it calculates a lower bound for the current partial assignment, which is a lower bound on the preference level of any complete assignment extending this partial assignment. Then, if this bound is strictly dominated by some previously found non-dominated solution, the search will backtrack, since all completed assignments extending the current partial assignment will be dominated. This approach improves on the original algorithm by eliminating parts of the search space that do not contain any non-dominated solutions, therefore eliminating unnecessary dominance checking and reducing the amount of time the algorithm spent doing these checks. We label this algorithm DFBBSEARCH.

For Sorted-Pareto, we can calculate a lower bound preference level for some partial assignment u, as follows: Let $c_1, \ldots, c_m$ be the soft constraints in $\mathcal{C}$ once u has been instantiated. Then the lower bound preference level $\rho_*(u)$ of u is given as $\rho_*(u) = (c_1^{\min}, \ldots, c_m^{\min})$, where $c_i^{\min} = \min_t c_i(t)$, i.e., the minimum value of c_i over all tuples t, where $t \in \prod_{X \in V_{c_i}} \mathcal{D}(X)$.

PANDSEARCH *algorithm*

We now look at another algorithm which is motivated as follows. Since in this problem we are dealing with a partial order on the set of complete assignments, we have a set of non-dominated solutions at each point in the search. However, not all of these assignments are relevant in each part of the search space, so if it can be shown that some complete assignment s fails to dominate any complete assignment extending partial assignment u, then there is no need to consider assignment s in the search space extending below u. To determine which previously found complete assignments are relevant to a partial assignment at a particular point in the search, the algorithm calculates an upper bound for the current partial assignment, which is a upper bound on the preference level of any complete assignment extending the current assignment. If this bound is not strictly dominated by the previously found complete assignment s, then s can be ignored in the subsearch extending below u. This improves on the previous algorithm by further eliminating unnecessary dominance checks, but at the cost of performing this extra test. A similar idea in [26] has been shown to be effective in optimisation with respect to comparative preferences.

For Sorted-Pareto, we can calculate an upper bound preference level for some partial assignment u, as follows: Let $c_1, \ldots, c_m$ be the soft constraints in $\mathcal{C}$ once u has been instantiated. Then the upper bound preference level $\rho^*(u)$ of u is given as $\rho^*(u) = (c_1^{\max}, \ldots, c_m^{\max})$, where $c_i^{\max} = \max_t c_i(t)$, i.e., the maximum value of c_i over all tuples t, where $t \in \prod_{X \in V_{c_i}} \mathcal{D}(X)$.

Algorithm Description: The algorithm is given in Figure 1. Firstly, we describe the auxiliary functions used by the algorithm during the course of the search. NEXTVAR() returns the next variable to be assigned. HASNEXTVAR() returns *true* if there is another variable to assign. NEXTVAL(X) returns the next value from the domain of variable X to assign. HASNEXTVAL(X) returns *true* if there is another value in the domain of variable X. ISSDOMINATED(u, S) returns *true* if complete assignment u is strictly dominated by any $s \in S$. REMOVESDOMINATED(S, S') returns a new set, containing the elements of set S that are not dominated by any element of set S'. ISCONSISTENT(u) returns *true* if partial assignment u is consistent with the hard constraints of the problem. ISPAD(lb, OPT) returns *true* if the preference level bound lb is dominated by some solution in the set OPT. CALCULATEUPPERBOUND(u)) calculates an upper bound preference level of partial assignment u. CALCULATELOWERBOUND(u) calculates a lower bound preference level of partial assignment u. REDUCE(ub, OPT) returns a pair of sets $\langle S, S' \rangle$, where S contains elements of OPT that dominate the bound ub, and S' contains elements of OPT that do not dominate the bound ub.

The main details of the algorithm are outlined as follows. The recursive function PANDSEARCH takes as input a partial assignment u and a set RUS, which is the set of relevant undominated solutions inherited from the parent node. NEW is the set of undominated solutions found so far in leaf nodes (line 2). If u is a complete assignment, i.e., there are no more variables to be assigned (line 3), then, if u is not dominated by any solution in RUS (line 4), it is added to the set NEW (line 5). Any solutions in RUS that are dominated by u are removed (line 6), and the pair of sets RUS and NEW are returned (line 8). Otherwise, if u is not a complete assignment, then a variable X is chosen (line 11), and a value in the domain of X is chosen to extend partial assignment u (line 13). A lower bound preference level lb for u is calculated (line 14), and if the lower bound of u is not dominated by any other solution (line 15), then the search continues. An upper bound preference level ub for u is calculated (line 16), and any previously found complete assignment s that does not dominate this upper bound is removed from RUS, and added to set variable OTS to allow such s to be restored on backtracking (line 17). The search continues with the recursive call to PANDSEARCH (line 18), until all non-dominated solutions are found and returned (line 25).

V. EXPERIMENTAL RESULTS

To generate the Sorted-Pareto problem instances, we used our Random Binary Soft CSP generator, which generates random problems using the following parameters: n is the number of variables, d is the domain size, and since we have separate soft and hard constraints, we have separate parameters for soft (sd) and hard (hd) density, and soft (st) and hard (ht) tightness. $hd \in [0, 1]$ is the fraction of hard constraints in the problem w.r.t. the maximum number of possible hard constraints. Similarly, $sd \in [0, 1]$ and is the fraction of soft constraints in the problem. $ht \in [0, 1]$, which is the fraction of the tuples of each hard constraint that are forbidden. $st \in [0, 1]$, is the fraction of the tuples of each soft constraint that have a non-zero weight. Optionally, instead of

Input: u: assignment, RUS: set of optimal solutions
Output: $\langle \text{RUS}, \text{NEW} \rangle$: pair of solution sets

```
1: function PANDSEARCH(u, RUS)
2:     NEW ← ∅
3:     if (¬HASNEXTVAR())
4:         if (¬ISSDOMINATED(u, RUS))
5:             NEW ← u
6:             RUS ← REMOVESDOMINATED(RUS, NEW)
7:         end if
8:         return ⟨RUS, NEW⟩
9:     end if
10:    RUS′ ← RUS
11:    X ← NEXTVAR()
12:    while (HASNEXTVAL(X)) do
13:        u ← u ∪ (X, NEXTVAL(X))
14:        lb ← CALCULATELOWERBOUND(u)
15:        if ([ISCONSISTENT(u)] ∧ [¬ISPAD(lb, RUS′)])
16:            ub ← CALCULATEUPPERBOUND(u)
17:            ⟨RUS′, OTS⟩ ← REDUCE(ub, RUS′)
18:            ⟨RUS′, NEW′⟩ ← PANDSEARCH(u, RUS′)
19:            OTS ← REMOVESDOMINATED(OTS, NEW′)
20:            RUS′ ← RUS′ ∪ OTS ∪ NEW′
21:        end if
22:    end while
23:    NEW ← RUS′ \ RUS
24:    RUS ← RUS′ ∩ RUS
25:    return ⟨RUS, NEW⟩
26: end function
```

Fig. 1. PANDSEARCH algorithm

specifying the soft and hard density values, the exact number of soft and hard constraints can be specified using the hc and sc parameters. For the purposes of the experiments, our solver uses a MAC3 algorithm [16] to maintain consistency, the min domain over degree heuristic [2] for variable selection, and a min sum of weights heuristic for value selection. The experiments were run on a dual Intel Xeon E5430 Processor (2.66Ghz) machine, with 12GB RAM.

Comparing Pareto and Sorted-Pareto: Table II shows the average number of consistent solutions |SOL|; the average number of Pareto non-dominated solutions $|\text{OPT}_P|$; and the average number of Sorted-Pareto non-dominated solutions $|\text{OPT}_S|$, for a set of 50 problem instances which were randomly generated using the following parameters: $d = 2$, $hd = 0.06$, $ht = 0.25$, $sd = 0.25$, $st = 1.00$. These parameters represent a family of problems where the number of consistent solutions grows exponentially with the size of the problem, given the range of the values of n used, (however at bigger values of increasing n, the rate of growth of the number of consistent solutions will drop off and eventually decline). For these small size problems we can see that for increasing values of n, the size of the set of Pareto non-dominated solutions grows very rapidly, whereas the the size of the set of Sorted-Pareto non-dominated solutions experiences much slower growth. Table III shows the sizes of the sets of Sorted-Pareto non-dominated solutions for larger values of n, showing that for larger n these sets are still moderately sized.

TABLE II
AVERAGE NUMBER OF CONSISTENT, PARETO NON-DOMINATED, AND SORTED-PARETO NON-DOMINATED SOLUTIONS OVER 50 INSTANCES, FOR $d = 2$, $hd = 0.06$, $ht = 0.25$, $sd = 0.25$, $st = 1.00$.

n	10	12	14	16	18	20
|SOL|	432	1292	3830	8785	19974	47018
$\|\text{OPT}_P\|$	38	141	419	1120	3838	13443
$\|\text{OPT}_S\|$	7	13	22	28	43	75

TABLE III
AVERAGE NUMBER OF CONSISTENT AND SORTED-PARETO NON-DOMINATED SOLUTIONS OVER 50 INSTANCES, FOR $d = 2$, $hd = 0.06$, $ht = 0.25$, $sd = 0.25$, $st = 1.00$

n	24	28	32	36	40
|SOL|	126500	417507	702907	1476080	1759033
$\|\text{OPT}_S\|$	128	216	247	352	403

Comparing algorithms for Sorted-Pareto: Figure 2 shows the average time in milliseconds (ms) for both BRUTESEARCH and DFBBSEARCH algorithms for solving 100 Sorted-Pareto problem instances which were randomly generated using the following parameters: $d = 2$, $hd = 0.06$, $ht = 0.25$, $sd = 0.20$, and $st = 0.50$. As in Tables II and III, these parameters are representative of problems where the number of consistent solutions grows exponentially with the size of the problem. The dotted line shows the average number of consistent solutions for the problems.

Figure 3 shows the average time in ms for both BRUTESEARCH and DFBBSEARCH algorithms for solving 100 Sorted-Pareto problem instances, randomly generated using the following parameters: $d = 3$, $ht = 0.44$, $sd = 0.20$, $st = 0.50$, and where the parameter for the number of hard constraints hc was varied to generate problems where the expected number of consistent solutions for each problem was roughly 1000. In both sets of results, the DFBBSEARCH algorithm significantly outperforms the BRUTESEARCH algorithm.

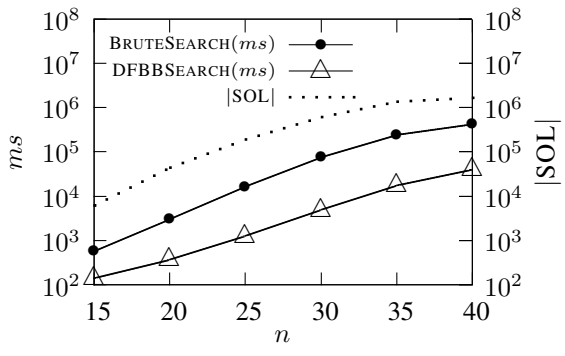

Fig. 2. BRUTESEARCH vs. DFBBSEARCH problems with number of consistent solutions growing, average time in ms on y-axis, number of variables n on x-axis. The second y-axis shows the average number of consistent solutions for the problems (dotted line).

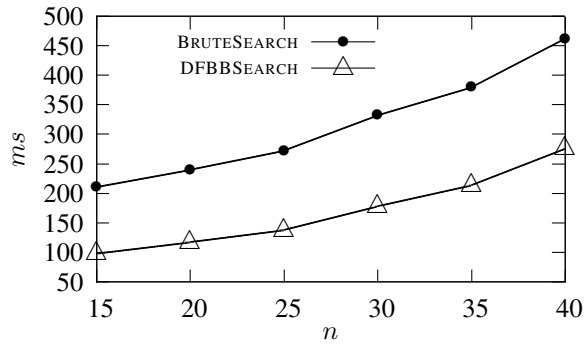

Fig. 3. BRUTESEARCH vs. DFBBSEARCH problems with number of consistent solutions ∼1000, average time in ms on y-axis, number of variables n on x-axis.

For the data in Figure 2, the DFBBSEARCH algorithm was on average 10.9 times faster than BRUTESEARCH.

Figure 4 shows the average time in ms for BRUTESEARCH, DFBBSEARCH and PANDSEARCH algorithms for solving 50 Sorted-Pareto problem instances, randomly generated using the following parameters: $d = 2$, $hd = 0.06$, $ht = 0.25$, $sc = 10$, $st = 1.0$. This set of parameters models a family of problems where the solutions are evaluated over 10 aspects, and the average sizes of the set of Sorted-Pareto non-dominated solutions are larger (e.g., for $n = 36$, the average number of solutions is 3861). The results show that there exists families of problems such that the PANDSEARCH algorithm outperforms the DFBBSEARCH algorithm.

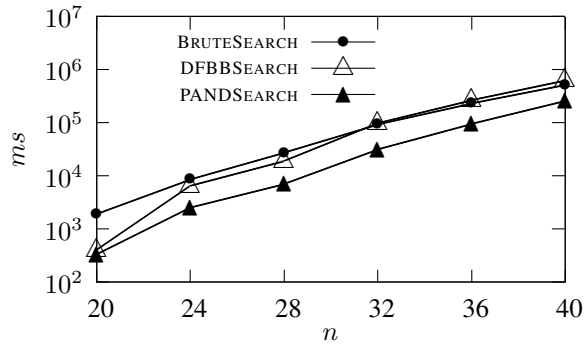

Fig. 4. BRUTESEARCH vs. DFBBSEARCH vs. PANDSEARCH problems with solutions evaluated over 10 aspects, average time in ms on y-axis, number of variables n on x-axis.

Non-random problems: As well as running our solver on randomly generated problems, Table IV shows some results for running our solver on some non-random weighted CSP problems from the planning problems domain. The table shows for each problem the number of variables (n), number of soft constraints (sc), number of hard constraints (hc), the number of min-sum-optimal solutions when solving the problem a standard weighted CSP ($|\text{OPT}_M|$), and the number of Sorted-Pareto non-dominated solutions when solving the problem using Sorted-Pareto ($|\text{OPT}_S|$). In these instances, the sets of Sorted-Pareto non-dominated solutions are small, but often still larger than the number of min-sum-optimal solutions (and every min-sum-optimal solution is always Sorted-Pareto non-dominated).

TABLE IV
NUMBER OF MIN-SUM-OPTIMAL SOLUTIONS VS. NUMBER OF SORTED-PARETO NON-DOMINATED SOLUTIONS

	n	sc	hc	$\lvert\text{OPT}_M\rvert$	$\lvert\text{OPT}_S\rvert$
zenotravel02ac	116	538	5223	1	1
zenotravel02bc	116	538	5223	1	2
zenotravel02c	116	538	5223	1	1
zenotravel02cc	116	538	5223	1	1
zenotravel04ac	239	1112	16904	1	2
zenotravel04bc	239	1112	16904	1	1
zenotravel04c	239	1112	16904	1	2
zenotravel04cc	239	1112	16904	1	5

VI. SORTED-PARETO EXTENSIONS

In this section, we briefly look at a couple of extensions to the Sorted-Pareto relation (with a view to motivating possible future work): the first involves a situation where there is additional information as to the importance of the aspects on which decisions are evaluated; the second looks at further refining a set of Sorted-Pareto non-dominated decisions.

A. Lexicographic Sorted-Pareto

Let $\mathcal{L} = \{L_1, \ldots, L_k\}$ represent an ordered partition of the set of aspects $\mathcal{S}$, ordered by $>$ in terms of importance, where each $i \in \mathcal{S}$ appears in only one $L \in \mathcal{L}$. The Lex-Sorted permutation, for some $\alpha \in \mathcal{A}$, is defined as $\alpha^{\uparrow}_{\mathcal{L}} = (\alpha^{\uparrow}[L_1], \alpha^{\uparrow}[L_2] \ldots, \alpha^{\uparrow}[L_k])$, where $\alpha^{\uparrow}[L_i]$ is the sorted permutation of $\alpha[L_i]$, i.e., the sorted permutation of the evaluations of α for the aspects of L_i. Then we define a relation as follows: for all $\alpha, \beta \in \mathcal{A}$, α Lex-Sorted-Pareto dominates β, written as $\alpha \preccurlyeq^{\mathcal{L}}_{S} \beta$, if and only if, $\exists j$, such that, for all $i < j$, $\alpha^{\uparrow}[L_i] \equiv_P \beta^{\uparrow}[L_i] \wedge \alpha^{\uparrow}[L_j] \prec_P \beta^{\uparrow}[L_j]$.

This represents a situation where decision aspects are given higher priority than others, for example, the aspects might be states that are more likely to occur, criteria that are more important, or voters with more weight. This is similar to the approach taken by [12] for handling preferences between criteria in multi-criteria problems, and to Lexicographic Constraint Satisifaction Problems in [9].

B. MinMax-Sorted-Pareto

When we have a set of Sorted-Pareto non-dominated solutions OPT_S to a particular problem, we can order the set using the MinMax-Sorted-Pareto relation $\preccurlyeq^{\max}_{S}$, defined as follows: for all $\alpha, \beta \in \text{OPT}_S$, $\alpha \preccurlyeq^{\max}_{S} \beta$, if and only if, $\max(\alpha) < \max(\beta)$. This relation forms a total pre-order on the set of decisions, and could be used as a tiebreaker between decisions, to present a single solution or very small set of solutions to a decision maker. It can also be considered as an

egalitarian approach, since it prefers the decisions that minimise the maximum weights, (if the weights are representing some sort of cost).

VII. Related Work

The notion of Sorted-Pareto appears in [15], which focuses more on the elicitation of the preferences and the normalisation of different criteria scales, whereas in our paper we assume that such a normalisation process has occurred and we look at an application of the relation in Soft Constraints. Another version appears in [13], where it is called "Ordered Pareto", and is used for handling preferences and comparing alternatives using possibilistic logic. [24] looks at "Pareto-rank dominance" where Pareto dominance is applied to ordered income distributions, however it is the nature of the problem that these income distributions are already ordered before the Pareto dominance relation is applied. Also [19] and [4] look at preference based search for generating sets of optimal solutions for shortest path problems, which is related to Sorted-Pareto (as described in Section III).

Some work that looks at branch and bound algorithms for partially ordered Soft Constraints includes [10] and [25]. Both [23] and [17] look at algorithms for multi-criteria optimisation in Soft Constraints for approximating Pareto optimal solution sets, and the work in [5] looks at depth first branch and bound algorithms for the computation of leximin optimal solutions in Constraint Networks.

VIII. Summary and Discussion

In this paper, we defined an extension to Pareto dominance called Sorted-Pareto dominance, and gave characterisations of the relation along with a semantics. We also explored Sorted-Pareto in the context of Soft Constraints, and implemented and experimentally tested three different depth-first algorithms for providing a set of Sorted-Pareto optimal solutions to Soft Constraints problems, which also involve hard constraints. The first algorithm only uses the Sorted-Pareto relation between solutions; the second uses a lower bound to prune the search tree, and the third also makes use of an upper bound to restrict the previously found undominated solutions that need to be considered in a subtree. The lower bound pruning generates an order of magnitude speedup, and the use of the upper bound can also sometimes be helpful.

Sorted-Pareto is very relevant to a situation where we have a weighted constraints problem [21, Ch. 9] (or, similarly, a GAI decomposition [1]) but the numerical values are only on an ordinal scale; Theorem 1 shows that one decision weakly Sorted-Pareto dominates another if and only if it weakly dominates it in all compatible standard weighted constraints problems. The experimental results showed that often the resulting set of optimal solutions is relatively small, and of certainly a much more manageable size than the set of Pareto-undominated solutions. If necessary, the set of non-dominated solutions can be refined by selecting (for example) the solutions that are also Minimax optimal. Future work will involve investigation and development of such refinements, for computing sets of optimal solutions for different notions of optimality in different situations.

Acknowledgments

This material is based upon works supported by the Science Foundation Ireland under Grant No. 08/PI/I1912.

References

[1] Fahiem Bacchus and Adam J. Grove. Graphical models for preference and utility. In *Proc. UAI, 1995*, pages 3–10, 1995.

[2] C. Bessière and J.C. Régin. MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems. In *Proc. CP, 1996*, pages 61–75, 1996.

[3] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *JACM*, 44:pp. 201–236, 1997.

[4] U. Bossong and D. Schweigert. Minimal paths on ordered graphs. *Mathematica Slovaca*, 56:pp. 23–31, 2006.

[5] S. Bouveret and M. Lemaître. Computing leximin-optimal solutions in constraint networks. *Artif. Intell.*, 173(2):pp. 343–364, 2009.

[6] R. Congar and F. Maniquet. A trichotomy of attitudes for decision-making under complete ignorance. *Mathematical Social Sciences*, 59(1):pp. 15–25, 2010.

[7] 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.

[8] H. Fargier, E. Rollon, and N. Wilson. Enabling local computation for partially ordered preferences. *Constraints*, 15(4):pp. 516–539, 2010.

[9] E.C. Freuder, R. Heffernan, R.J. Wallace, and N. Wilson. Lexicographically-ordered constraint satisfaction problems. *Constraints*, 15(1):1–28, 2010.

[10] M. Gavanelli. Partially ordered constraint optimization problems. In *Proc. CP 2001*, page 763, 2001.

[11] C. Gonzales, P. Perny, and J. Dubus. Decision making with multiple objectives using GAI networks. *Artif. Intell.*, 175:pp. 1153–1179, 2011.

[12] U. Junker. Preference-based search and multi-criteria optimization. *Annals OR*, 130(1-4):pp. 75–115, 2004.

[13] S. Kaci and H. Prade. Mastering the processing of preferences by using symbolic priorities in possibilistic logic. In *Proc. ECAI 2008*, pages 376–380, 2008.

[14] R.L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[15] O.I. Larichev and H.M. Moshkovich. ZAPROS-LM – a method and system for ordering multiattribute alternatives. *European Journal of Operational Research*, 82(3):pp. 503–521, 1995.

[16] A.K. Mackworth. Consistency in networks of relations. *Artif. Intell.*, 8(1):pp. 99–118, 1977.

[17] R. Marinescu. Efficient approximation algorithms for multi-objective constraint optimization. In *Proc. ADT 2011*, pages 150–164, 2011.

[18] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, San Fran., CA, USA, 1988.

[19] P. Perny and O. Spanjaard. A preference-based approach to spanning trees and shortest paths problems. *European Journal of Operational Research*, 162:pp. 584–601, 2005.

[20] P. Perny, O. Spanjaard, and L. Storme. A decision-theoretic approach to robust optimization in multivalued graphs. *Annals of Operations Research*, 147:pp. 317–341, 2006.

[21] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming*. Elsevier Science Inc., New York, NY, USA, 2006.

[22] A.K. Sen. *Collective choice and social welfare*. North-Holland Publishing Co., Amsterdam, 1970.

[23] M. Torrens and B. Faltings. Using soft CSPs for approximating Pareto-optimal solution sets. In *Proc. AAAI 2002 Workshop: Preferences in AI and CP: Symbolic Approaches*, 2002.

[24] S. Traub, C. Seidl, and U. Schmidt. An experimental study on individual choice, social welfare, and social preferences. *European Economic Review*, 53(4):pp. 385–400, 2009.

[25] N. Wilson and H. Fargier. Branch-and-bound for soft constraints based on partially ordered degrees of preference. In *Proc. ECAI 08 Workshop (WIGSK08)*, 2008.

[26] N. Wilson and W. Trabelsi. Pruning rules for constrained optimisation for conditional preferences. In *Proc. CP 2011*, pages 804–818, 2011.