Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Incremental Weight Elicitation for Multiobjective State Space Search

Nawal Benabbou and **Patrice Perny**
Sorbonne Universites, UPMC Univ Paris 06, UMR 7606, LIP6
CNRS, UMR 7606, LIP6, F-75005, Paris, France
4 Place Jussieu, 75005 Paris, France
nawal.benabbou@lip6.fr, patrice.perny@lip6.fr

Abstract

This paper proposes incremental preference elicitation methods for multiobjective state space search. Our approach consists in integrating weight elicitation and search to determine, in a vector-valued state-space graph, a solution path that best fits the Decision Maker's preferences. We first assume that the objective weights are imprecisely known and propose a state space search procedure to determine the set of possibly optimal solutions. Then, we introduce incremental elicitation strategies during the search that use queries to progressively reduce the set of admissible weights until a nearly-optimal path can be identified. The validity of our algorithms is established and numerical tests are provided to test their efficiency both in terms of number of queries and solution times.

Introduction

Preference-based search is an active topic in Artificial Intelligence with various applications to constraint satisfaction, planning, search, resource allocation and electronic commerce (see e.g. (Boutilier et al. 2004; Brafman and Domshlak 2009; Domshlak et al. 2011)). Although many algorithmic contributions focus on the elaboration of efficient algorithms to determine a best solution given a preference model, some others concern preference elicitation strategies so as to best fit the decision model to Decision Maker's (DM) preferences. In this paper, we address both aspects simultaneously. We study the potential of incremental elicitation methods (White III , Sage, and Dozono 1984; Wang and Boutilier 2003; Braziunas and Boutilier 2007) in the framework of multiobjective state space search (Stewart and White III 1991; Mandow and De la Cruz 2005). This is a decision context where solutions are very numerous and defined implicitly. Hence usual elicitation methods based on systematic pairwise comparisons are often ineffective.

Elicitation on combinatorial domains is a challenging issue that recently motivated several contributions in various contexts, e.g. in constraint satisfaction problems (Boutilier et al. 2006), in Markov Decision Processes (Regan and Boutilier 2011; Weng and Zanuttini 2013), in stable matching problems (Drummond and Boutilier 2014) and in multiattribute spaces (Gonzales and Perny 2004; Braziunas and Boutilier 2007; Koriche and Zanuttini 2010). In the context of multiobjective state space search, our aim here is to propose a new approach integrating preference elicitation and search. We consider a state space graph endowed with q evaluation criteria, i.e. q cost functions to be minimized (e.g. time, distance, energy, risk). Each path is therefore valued by a cost vector and preference over paths are inherited from the preference over their cost vectors. Preference over cost vectors are defined using a parameterized aggregation function $f_\omega : \mathbb{R}^q_+ \rightarrow \mathbb{R}_+$ defining the overall cost (or disutility) $f_\omega(x)$ attached to any cost vector $x \in \mathbb{R}^q_+$, where ω is a vector of preference parameters (weights) representing the relative importance of criteria. We want to find a path from an initial node to a goal node that minimizes the overall cost function f_ω that represents the DM's preferences. However, for the elicitation purpose, we assume that ω is not known precisely, and we intend to elicit the weights during the search so as to determine the best possible solution path.

The elicitation of criteria weights defining ω can be performed in different ways. The first way, quite standard, consists in scanning the space of weights and repeatedly selecting a vector ω to generate an f_ω-optimal solution until one meets the satisfaction of the DM; see e.g. (Zionts and Wallenius 1976; Vanderpooten and Vincke 1997). In multiobjective state space search, when f_ω is linear (i.e. $f_\omega(x) = \omega \cdot x$) the generation of an f_ω-optimal solution at every step of the exploration can simply be performed by a standard A^* search (Hart, Nilsson, and Raphael 1968) after a preliminary scalarization of costs using f_ω. When f_ω is not linear, other algorithms have been proposed, more complex, to solve the optimization problem for various classes of aggregation functions, e.g. multiattribute utility functions (Dasgupta, Chakrabarti, and DeSarkar 1995), weighted Tchebycheff norms (Galand and Perny 2006), and Choquet integrals (Galand and Perny 2007). They can be used to generate the current solution in an interactive exploration method.

A second way of performing preference elicitation in multiobjective combinatorial problems, less explored, consists in integrating an incremental elicitation method into the search procedure. In our case, starting with an initial set of possible weights, it consists in generating preference queries during the search so as to progressively reduce the set of admissible parameters until an optimal path can be determined

or approximated with some guarantees. This is the line we explore in this paper. Assuming that f_ω is a linear model and that ω is not known, we first propose a state space search procedure to determine the set of possibly f_ω-optimal solutions. Then, we introduce incremental elicitation strategies into the search so as to progressively reduce the set of admissible weights until a nearly-optimal path is found.

The General Framework

Let $G=(N,A)$ be a state space graph where N is the finite set of nodes representing the states and A is the set of arcs representing the feasible state transitions. More precisely, $A=\{(n,n') : n\in N, n'\in S(n)\}$ where $S(n)\subseteq N$ is the set of all successors of n. We consider a finite set of criteria $g_i : A\to\mathbb{R}_+$, $i\in Q=\{1,\ldots,q\}$. We assume that g_i's are cost functions to be minimized. Hence, graph G is endowed with a vector valuation function $g : A\to\mathbb{R}^q_+$ which assigns to each arc $a\in A$ the cost vector $g(a)=(g_1(a),\ldots,g_q(a))$. A path between n and n' is characterized by a list of nodes of type $\langle n,\ldots,n'\rangle$ and $\circ$ is the path appending operator. The set of all paths linking n to n' is denoted $P(n,n')$ and the set of all paths linking n to any goal node $\gamma\in\Gamma$ is denoted $P(n,\Gamma)$; thus, the set of solution paths, starting at the source node $s\in N$, is denoted $P(s,\Gamma)$. The cost vector of a path p is denoted $g(p)$ and is the sum of the cost vector of its constituent arcs. The set of feasible cost vectors is denoted $\mathcal{X}=\{g(p) : p\in P(s,\Gamma)\}$. It represents the image of all solution paths in the space of criteria.

We assume that the DM's preferences are represented by a linear function $f_\omega(x)=\omega\cdot x$ measuring the overall cost of paths. Since ω is not known precisely, we consider a set Ω containing all normalized weighting vectors compatible with the observed preference statements. By default, Ω is initially defined as the simplex $\{\omega\in\text{int}(\mathbb{R}^q_+) : \sum_{i=1}^q\omega_i=1\}$ where int represents the interior of the cone $\mathbb{R}^q_+$ (this prevents to have any zero in the weighting vector and guarantees the Pareto-optimality of f_ω-optimal cost vectors). Due to the linearity of f_ω, any preference statement of type "x is at least as good as y" induces a linear constraint $f_\omega(x)-f_\omega(y)\leq 0$ over the simplex. Hence, throughout the paper, we assume without loss of generality that Ω is a convex polyhedron. Given the uncertainty set Ω, we first consider the problem of determining the set $\text{PO}_\Omega(\mathcal{X})$ of possibly f_ω-optimal cost vectors in $\mathcal{X}$, i.e. the set of cost vectors $x\in\mathcal{X}$ that minimize f_ω for some $\omega\in\Omega$. Formally:

Definition 1. $\forall X\subseteq\mathbb{R}^q_+$, $\text{PO}_\Omega(X)=\bigcup_{\omega\in\Omega}\arg\min_{x\in X} f_\omega(x)$.

When X is a set of cost vectors associated to paths in G, the subset of paths having their cost vector in $\text{PO}_\Omega(X)$ is called the set of possibly optimal paths. We first address the computation of $\text{PO}_\Omega(X)$, for a finite set of vectors $X\subset\mathbb{R}^q_+$ given in extension. Then we will propose an adaptation of MOA* search so as to construct $\text{PO}_\Omega(\mathcal{X})$.

Determination of Possibly Optimal Elements

Let $X=\{x^1,\ldots,x^m\}\subset\mathbb{R}^q_+$ be a non-empty and finite set of vectors. Let us introduce the Ω-dominance relation $\prec_\Omega$ defined on subsets of vectors by:

Definition 2 (Ω-dominance). *For all $Y,Z\subseteq\mathbb{R}^q_+$:*
$Z\prec_\Omega Y$ if and only if $\forall y\in Y,\forall\omega\in\Omega,\exists z\in Z,\omega\cdot z<\omega\cdot y$

This relation is useful to determine $\text{PO}_\Omega(X)$ due to the following proposition.

Proposition 1. *If there exists $Y,Z\subseteq X$ such that $Z\prec_\Omega Y$ then $Y\cap\text{PO}_\Omega(X)=\emptyset$.*

Proof. Consider $Y,Z\subseteq X$ such that $Z\prec_\Omega Y$. By Definition 2, we know that, for all $y\in Y$, for all $\omega\in\Omega$, there exists $z\in Z\subseteq X$ such that $\omega\cdot z<\omega\cdot y$. Hence no element $y\in Y$ is f_ω-optimal for some $\omega\in\Omega$. Therefore no element of Y belongs to $\text{PO}_\Omega(X)$, which establishes the result. □

This proposition enables us to propose the following algorithm to compute $\text{PO}_\Omega(X)$:

Algorithm 1: Ω-Filter(X)

Input: Ω; $X=\{x^1,\ldots,x^m\}$; σ: permutation of $\{1,\ldots,m\}$
Output: X^σ_m

```
X^σ_0 ← X
for i = 1 ... m do
    if X^σ_{i-1} ≺_Ω {x^{σ(i)}} then X^σ_i ← X^σ_{i-1} \ {x^{σ(i)}}
    else X^σ_i ← X^σ_{i-1}
```

Before establishing the validity of Algorithm 1, let us remark that it builds nested sets X^σ_i, $i=1\ldots m$, by iteratively testing whether $X^\sigma_{i-1}\prec_\Omega\{x^{\sigma(i)}\}$ and discarding $x^{\sigma(i)}$ if the test succeeds, where σ is a given permutation of $\{1,\ldots,m\}$. Since Ω is described by a (reasonably small) set of linear constraints (at most one per preference statement obtained from the DM), testing whether $X^\sigma_{i-1}\prec_\Omega\{x^{\sigma(i)}\}$ can be performed by solving $\max_{\omega\in\Omega}\min_{x\in X^\sigma_{i-1}}\omega\cdot(x-x^{\sigma(i)})$ due to the following proposition:

Proposition 2. *For all $Y,Z\subseteq\mathbb{R}^q_+$:*

$$Z\prec_\Omega Y\Leftrightarrow\left[\forall y\in Y,\max_{\omega\in\Omega}\min_{z\in Z}\ \omega\cdot(z-y)<0\right]$$

Proof. Let $Y,Z\subseteq\mathbb{R}^q_+$ such that $Z\prec_\Omega Y$. Let $y\in Y$. For all $\omega\in\Omega$, there exists $z\in Z$ such that $\omega\cdot z<\omega\cdot y$ (Definition 2). Hence, we have $\min_{z\in Z}\ \omega\cdot(z-y)<0$ for all $\omega\in\Omega$ and in particular, we have $\max_{\omega\in\Omega}\min_{z\in Z}\ \omega\cdot(z-y)<0$. Reciprocally, let $Y,Z\subseteq\mathbb{R}^q_+$ such that for all $y\in Y$, $\max_{\omega\in\Omega}\min_{z\in Z}\omega\cdot(z-y)<0$. Let $y\in Y$. Then, for all $\omega\in\Omega$, we have $\min_{z\in Z}\ \omega\cdot(z-y)<0$ and so there exists $z\in Z$ such that $\omega\cdot z<\omega\cdot y$. Thus $Z\prec_\Omega Y$ (Definition 2). □

Hence, Algorithm 1 requires to solve only m linear programs of bounded size and therefore is polynomial in the size of X. Now, to establish the validity of Algorithm 1, we first establish Lemma 1 showing that the output of Algorithm 1 is independent of the permutation σ.

Lemma 1. *For any two permutations σ_1 and σ_2, the two sets $X^{\sigma_1}_m$ and $X^{\sigma_2}_m$ are equal, where $X^{\sigma_1}_m$ (resp. $X^{\sigma_2}_m$) denotes the output of Algorithm 1 obtained from σ_1 (resp. σ_2).*

Proof. Let σ_1 and σ_2 be two permutations of $\{1,\dots,m\}$. Let us prove (by contradiction) that $X_m^{\sigma_1} \subseteq X_m^{\sigma_2}$. Let $x^i \in X_m^{\sigma_1}$ and assume $x^i \notin X_m^{\sigma_2}$. First, let $j \in \{1,\dots,m\}$ be the iteration step such that $i = \sigma_1(j)$. Since $x^i \in X_m^{\sigma_1}$, we necessarily have not$(X_{j-1}^{\sigma_1} \prec_\Omega \{x^i\})$, i.e. there exists $\omega' \in \Omega$ such that $\omega' \cdot x \geq \omega' \cdot x^i$ for all $x \in X_{j-1}^{\sigma_1}$ (Definition 2). Then, let $k \in \{1,\dots,m\}$ be the iteration step such that $i = \sigma_2(k)$. Since $x^i \notin X_m^{\sigma_2}$, we necessarily have $X_{k-1}^{\sigma_2} \prec_\Omega \{x^i\}$, i.e. for all $\omega \in \Omega$, there exists $x \in X_{k-1}^{\sigma_2}$ such that $\omega \cdot x < \omega \cdot x^i$ (Definition 2); hence, the set $X' = \{x \in X : \omega' \cdot x < \omega' \cdot x^i\}$ is not empty since we have $X_{k-1}^{\sigma_2} \subseteq X$ by construction. Then, by definition of ω', we have $X' \cap X_{j-1}^{\sigma_1} = \emptyset$. Hence $X' \cap X_m^{\sigma_1} = \emptyset$ since $X_{j-1}^{\sigma_1} \supseteq X_m^{\sigma_1}$ by construction. Therefore, for all $l \in \{1,\dots,m\}$ such that $x^{\sigma_1(l)} \in X'$, we necessarily have $x^{\sigma_1(l)} \in X_{l-1}^{\sigma_1}$ and $x^{\sigma_1(l)} \notin X_l^{\sigma_1}$. Hence, letting L denote the largest iteration step l such that $x^{\sigma_1(l)} \in X'$, we have $X' \cap X_{L-1}^{\sigma_1} = \{x^{\sigma_1(L)}\}$. Then, since we know that $x^{\sigma_1(L)} \notin X_L^{\sigma_1}$, we necessarily have $X_{L-1}^{\sigma_1} \prec_\Omega \{x^{\sigma_1(L)}\}$ and so there exists $x' \in X_{L-1}^{\sigma_1}$ such that $\omega' \cdot x' < \omega' \cdot x^{\sigma_1(L)}$ (Definition 2). Moreover, since $X' \cap X_{L-1}^{\sigma_1} = \{x^{\sigma_1(L)}\}$, we also know that $x' \notin X'$. Finally, since $x^{\sigma_1(L)} \in X'$ by definition, we have $\omega' \cdot x' < \omega' \cdot x^{\sigma_1(L)} < \omega' \cdot x^i$ which contradicts the fact that $x' \notin X'$; hence $x^i \in X_m^{\sigma_2}$. Finally, since σ_1 and σ_2 can be any two permutations, we conclude that $X_m^{\sigma_1} = X_m^{\sigma_2}$. □

Thus, Algorithm 1 returns the same subset of X whatever the permutation σ of $\{1,\dots,m\}$.

Theorem 1. *Algorithm 1 returns exactly the set* $\mathrm{PO}_\Omega(X)$.

Proof. Let X_m denote the output of Algorithm 1. First, let us prove that $\mathrm{PO}_\Omega(X) \subseteq X_m$. Let $x^i \in \mathrm{PO}_\Omega(X)$. By Definition 1, there exists $\omega' \in \Omega$ such that $x^i \in \arg\min_{x \in X} \omega' \cdot x$, i.e. $\omega' \cdot x \geq \omega' \cdot x^i$ for all $x \in X$. Hence we have not$(X \prec_\Omega \{x^i\})$ by Definition 2. Thus, we have not$(X_{1-1}^{\sigma} \prec_\Omega \{x^{\sigma(1)}\})$ for any permutation σ of $\{1,\dots,m\}$ such that $\sigma(1) = i$ and then we have $x^i \in X_m^{\sigma}$ by construction; therefore, we have $x^i \in X_m$ by Lemma 1. Now, let us prove that $X_m \subseteq \mathrm{PO}_\Omega(X)$. Let $x^i \in X_m$. Let σ be a permutation of $\{1,\dots,m\}$ such that $\sigma(1) = i$. Since $x^i \in X_m$, we have $x^i \in X_m^{\sigma}$ by Lemma 1. Then, by construction of X_m^{σ}, we have not$(X_{1-1}^{\sigma} \prec_\Omega \{x^i\})$, i.e. not$(X \prec_\Omega \{x^i\})$. Therefore, by Definition 2, there exists $\omega' \in \Omega$ such that $\omega' \cdot x \geq \omega' \cdot x^i$ for all $x \in X$, i.e. $x^i \in \arg\min_{x \in X} \omega' \cdot x$. Thus, we have $x^i \in \mathrm{PO}_\Omega(X)$. □

Algorithm 1 assumes that X is given explicitly. It cannot directly be applied on $\mathcal{X}$ which is only implicitly known as the image of all solution paths in the space of criteria. Instead, we propose now a search procedure directly focusing on possibly optimal cost vectors.

Search of Possibly Optimal Paths

Preference-based search methods in multiobjective optimization are often based on the exploration of the set of Pareto-optimal solutions. The so-called MOA^* algorithm (Stewart and White III 1991; Mandow and De la Cruz 2005) is a multiobjective extension of A^* (Hart, Nilsson, and Raphael 1968) that determines $ND(\mathcal{X}) \subseteq \mathcal{X}$ the set of Pareto non-dominated cost vectors attached to paths in $P(s,\Gamma)$ and returns one path for each element. Formally $ND(\mathcal{X}) = \{x \in \mathcal{X} : \forall y \in \mathcal{X}, \text{not}(y \prec_P x)\}$ where $\prec_P$ is the Pareto dominance relation on cost vectors, i.e. $x \prec_P y$ (read x Pareto-dominates y) if and only if $x_i \leq y_i$ for all $i \in Q$ and $x_k < y_k$ for some $k \in Q$. At some point, we will also consider the weak Pareto dominance, i.e. $x \precsim_P y$ (read x weakly Pareto-dominates y) if and only if $x_i \leq y_i$ for all $i \in Q$. Since all preference models considered in multicriteria analysis are compatible with Pareto dominance or weak Pareto dominance, the MOA^* algorithm is a useful basis to develop more specific preference-based search procedures. To obtain $\mathrm{PO}_\Omega(\mathcal{X})$, a two-stage procedure consisting first in using MOA^* to determine the set of Pareto-optimal feasible vectors $ND(\mathcal{X})$ and then in applying Algorithm 1 to determine $\mathrm{PO}_\Omega(\mathcal{X})$ would be admissible. However, it would not be very efficient due to the possibly large size of $ND(\mathcal{X})$ (which can be exponential in the number of nodes in G). We introduce now a variant of MOA^* to directly determine the set $\mathrm{PO}_\Omega(\mathcal{X})$ and one path for each of its elements.

In the multiobjective case, there possibly exists several optimal paths with different cost vectors to reach a given node. Therefore, the basic graph exploration procedure consists in iteratively expanding labels attached to subpaths rather than nodes, like in MOA^* (Mandow and De la Cruz 2005). Labels are of the form $\ell = [n_\ell, p_\ell, g_\ell]$ where p_ℓ denotes a path from s to n_ℓ and $g_\ell = g(p_\ell)$ denotes its cost. At any iteration of the algorithm, a label is selected for expansion. The expansion of a label ℓ^* generates the set of its successors $\{[n, p_{\ell^*} \circ n, g(p_{\ell^*} \circ n)] : n \in S(n_{\ell^*})\}$. The set of generated labels is divided into two disjoint sets: a set $\mathcal{C}$ of *closed* labels (yet expanded) and a set $\mathcal{O}$ of *open* labels (candidate to expansion). The set $\mathcal{C}$ (resp. $\mathcal{O}$) restricted to labels ℓ such that $p_\ell \in P(s,n)$ attached to node n is denoted $\mathcal{C}(n)$ (resp. $\mathcal{O}(n)$). Moreover, the expanded labels corresponding to the current possibly optimal solution paths are stored in a set denoted $\mathcal{S}$. Another feature imported from MOA^* is that, for each generated label ℓ, a set $F(\ell) = \{g_\ell + h : h \in H(n_\ell)\}$ of cost vectors is computed to estimate the cost vectors of the solution paths extending p_ℓ, where $H(n_\ell)$ is a set of heuristic costs estimating the set $\{g(p) : p \in P(n_\ell, \Gamma)\}$. The main differences with MOA^* lie in the definition of pruning rules that use sharper conditions than those based on Pareto-dominance tests. Our approach relies on the following property of Ω-dominance:

Proposition 3 (Additivity).

$$\forall X, Y, Z \subseteq \mathbb{R}_+^q, X \prec_\Omega Y \Rightarrow X + Z \prec_\Omega Y + Z$$

where $B + C = \{b + c : b \in B, c \in C\}$ *for all* $B, C \subseteq \mathbb{R}_+^q$.

Proof. Let $X, Y \subseteq \mathbb{R}_+^q$ be such that $X \prec_\Omega Y$. Let $Z \subseteq \mathbb{R}_+^q$. Let $u \in Y + Z$. Since $u \in Y + Z$, there exists $y \in Y$ and $z \in Z$ such that $u = y + z$. Let $\omega \in \Omega$. Since $X \prec_\Omega Y$, there exists $x \in X$ such that $\omega \cdot x < \omega \cdot y$. Then $\omega \cdot x + \omega \cdot z < \omega \cdot y + \omega \cdot z$ by the addition property of inequalities, i.e. $\omega \cdot (x+z) < \omega \cdot u$. Finally, since $x + z \in X + Z$, the result is established. □

Hence, we establish two propositions enabling the identi-

fication of labels corresponding to subpaths that cannot be extended into a possibly f_ω-optimal solution path.

Proposition 4. *At any node $n \in N$, if there exists $\ell' \in \mathcal{O}(n)$ and a set $L \subseteq \mathcal{O}(n) \cup \mathcal{C}(n)$ such that $\{g_\ell : \ell \in L\} \prec_\Omega \{g_{\ell'}\}$, then path $p_{\ell'}$ is not part of a possibly optimal solution path.*

Proof. Consider $n \in N$, $\ell' \in \mathcal{O}(n)$ and $L \subseteq \mathcal{O}(n) \cup \mathcal{C}(n)$ such that $\{g_\ell : \ell \in L\} \prec_\Omega \{g_{\ell'}\}$. Let $p \in P(n, \Gamma)$. We want to prove that $g(p_{\ell'} \circ p) \notin \mathrm{PO}_\Omega(\mathcal{X})$. Since $\prec_\Omega$ is additive (Proposition 3), we have $\{g_\ell : \ell \in L\} + \{g(p)\} \prec_\Omega \{g_{\ell'}\} + \{g(p)\}$, i.e. $\{g_\ell + g(p) : \ell \in L\} \prec_\Omega \{g_{\ell'} + g(p)\}$. Therefore, we have $\{g(p_\ell \circ p) : \ell \in L\} \prec_\Omega \{g(p_{\ell'} \circ p)\}$. Finally, since $g(p_{\ell'} \circ p) \in \mathcal{X}$ and $\{g(p_\ell \circ p) : \ell \in L\} \subseteq \mathcal{X}$, we have $g(p_{\ell'} \circ p) \notin \mathrm{PO}_\Omega(\mathcal{X})$ by Proposition 1. □

This proposition says that we can discard any label $\ell' \in \mathcal{O}(n)$ during the search whenever $\{g_\ell : \ell \in L\} \prec_\Omega \{g_{\ell'}\}$ for some subset $L \subseteq \mathcal{O}(n) \cup \mathcal{C}(n)$. In practice, there is no need to enumerate and test all subsets L to discard ℓ'. It is indeed sufficient to make the test for $L = \mathcal{O}(n) \cup \mathcal{C}(n)$ by definition of Ω-dominance. Then, the elimination of labels in $\mathcal{O}(n)$ based on Proposition 4 is performed by a single call to Ω-Filter (see Algorithm 1) on the input $\mathcal{O}(n) \cup \mathcal{C}(n)$ and we intersect the output with $\mathcal{O}(n)$ so as to obtain the labels that deserve further development. To summarize, we will use the following pruning rule:

Rule R1. *Discard the labels in $\mathcal{O}(n)$ that do not belong to Ω-Filter$(\mathcal{O}(n) \cup \mathcal{C}(n))$.*

The second pruning rule we are going to introduce assumes, as in MOA*, that heuristic H is admissible (i.e. H provides an optimistic evaluation of solution paths to the goal). This assumption is formalized as follows: for all $n \in N$, for all $p \in P(n, \Gamma)$, there exists $h \in H(n)$ such that $h \precsim_{\mathrm{P}} g(p)$. Under this assumption, the following proposition holds:

Proposition 5. *At any $n \in N$, if there exists $\ell' \in \mathcal{O}(n)$ and $L \subseteq \mathcal{S} \cup \bigcup_{\gamma \in \Gamma} \mathcal{O}(\gamma)$ such that $\{g_\ell : \ell \in L\} \prec_\Omega \{g_{\ell'}\} + H(n)$, then path $p_{\ell'}$ is not part of a possibly optimal solution path.*

Proof. Consider $n \in N$, $\ell' \in \mathcal{O}(n)$ and $L \subseteq \mathcal{S} \cup \bigcup_{\gamma \in \Gamma} \mathcal{O}(\gamma)$ such that $\{g_\ell : \ell \in L\} \prec_\Omega \{g_{\ell'}\} + H(n)$. For any $p \in P(n, \Gamma)$ we want to prove that $g(p_{\ell'} \circ p) \notin \mathrm{PO}_\Omega(\mathcal{X})$. Since H is admissible, there exists $h \in H(n)$ such that $h \precsim_{\mathrm{P}} g(p)$. Then we have $g_{\ell'} + h \precsim_P g_{\ell'} + g(p) = g(p_{\ell'} \circ p)$. Therefore, for all $\omega \in \Omega$, we have $\omega \cdot (g_{\ell'} + h) \leq \omega \cdot g(p_{\ell'} \circ p)$ since ω is a vector of positive weights. Then, since $\{g_\ell : \ell \in L\} \prec_\Omega \{g_{\ell'}\} + H(n)$, for all $\omega \in \Omega$, there exists $x \in \{g_\ell : \ell \in L\}$ such that $\omega \cdot x < \omega \cdot (g_{\ell'} + h)$; hence $\omega \cdot x < \omega \cdot g(p_{\ell'} \circ p)$ by transitivity of inequalities. Therefore, we have $\{g_\ell : \ell \in L\} \prec_\Omega \{g(p_{\ell'} \circ p)\}$. Finally, since $g(p_{\ell'} \circ p) \in \mathcal{X}$ and $\{g_\ell : \ell \in L\} \subseteq \mathcal{X}$, we have $g(p_{\ell'} \circ p) \notin \mathrm{PO}_\Omega(\mathcal{X})$ by Proposition 1. □

As for rule R1, there is no need to test all subsets L to discard ℓ'. It is sufficient to make the test for $L = \mathcal{S} \cup \bigcup_{\gamma \in \Gamma} \mathcal{O}(\gamma)$. Hence we will use this second rule.

Rule R2. *Discard $\ell' \in \mathcal{O}(n)$ if $\{g_\ell : \ell \in L\} \prec_\Omega \{g_{\ell'}\} + H(n)$ where $L = \mathcal{S} \cup \bigcup_{\gamma \in \Gamma} \mathcal{O}(\gamma)$.*

Algorithm 2: $\mathrm{PO}_\Omega(\mathcal{X})$ computation

Input: $G = (N, A)$; s; Γ; H; Ω
Output: $\mathcal{S}$

1 **foreach** $n \in N$ **do**
2 $\mathcal{O}(n) \leftarrow \emptyset$; $\mathcal{C}(n) \leftarrow \emptyset$
3 $\mathcal{O}(s) \leftarrow \{[s, \langle s \rangle, (0, \ldots, 0)]\}$; $\mathcal{S} \leftarrow \emptyset$
4 **while** $\mathcal{O} \neq \emptyset$ **do**
5 $\ell^* \leftarrow$ SELECT($\mathcal{O}$)
6 Move ℓ^* from $\mathcal{O}$ to $\mathcal{C}$
7 **if** $n_{\ell^*} \in \Gamma$ and $\forall \ell \in \mathcal{S}$, not$(g_\ell \precsim_{\mathrm{P}} g_{\ell^*})$ **then**
8 Add ℓ^* to $\mathcal{S}$
9 **else**
10 **foreach** $n \in S(n_{\ell^*})$ **do**
11 Generate $\ell' = [n, p_{\ell^*} \circ \langle n \rangle, g_{\ell^*} + g((n_{\ell^*}, n))]$
12 **if** $\forall \ell \in \mathcal{O}(n) \cup \mathcal{C}(n)$, not$(g_\ell \precsim_{\mathrm{P}} g_{\ell'})$ **then**
13 Add ℓ' to $\mathcal{O}(n)$
14 Apply rule R2 to ℓ'
15 **if** ℓ' is not discarded **then**
16 Apply rule R1 to $\mathcal{O}(n)$
17 $\mathcal{S} \leftarrow \Omega$-Filter($\mathcal{S}$)

Using the above results, we propose Algorithm 2 to compute $\mathrm{PO}_\Omega(\mathcal{X})$ where SELECT(L) returns a label $\ell \in L$ such that: $\exists f \in F(\ell), \forall \ell' \in \mathcal{O}, \forall f' \in F(\ell')$, not$(f' \precsim_{\mathrm{P}} f)$.

Theorem 2. *Algorithm 2 returns exactly the set $\mathrm{PO}_\Omega(\mathcal{X})$ and one solution path for each returned element.*

Proof. First, let us remark that Algorithm 2 is based on a MOA* search of Pareto-optimal cost vectors attached to solution paths, refined by two pruning rules R1 and R2. This focus on Pareto-optimal solutions is justified by the fact that all vectors in Ω have strictly positive components by hypothesis (hence possibly optimal vectors are necessarily Pareto-optimal). Then, we know that we cannot loose a possibly optimal cost vector by discarding labels with rule R1 or R2 due to propositions 4 and 5. Hence the set of returned labels $\mathcal{S}$ contains necessarily all elements in $\mathrm{PO}_\Omega(\mathcal{X})$ at the end of the while loop. Finally, the last call of Ω-Filter on $\mathcal{S}$ (see line 17 of Algorithm 2) eliminates, if necessary, any solution that does not belong to $\mathrm{PO}_\Omega(\mathcal{X})$. □

Incremental Elicitation During the Search

In Section 2, we were considering that Ω, the set of admissible weights, is stable over the time. We consider now that new preference statements about paths are obtained at different times (denoted $1, \ldots, T$) during the search (we assume that Algorithm 2 starts at time 0 and terminates at time $T+1$). This induces additional constraints that progressively reduce the set Ω. Let Ω_t be the set Ω at time t of the process. We obtain a nested sequence of sets since $\Omega_{t+1} \subseteq \Omega_t$ for all $t \in \{0, \ldots, T\}$ and therefore $\mathrm{PO}_{\Omega_{t+1}}(\mathcal{X}) \subseteq \mathrm{PO}_{\Omega_t}(\mathcal{X})$ by Definition 1. Hence the labels discarded at time t correspond to subpaths that cannot be part of a possibly optimal solution path at any time $t' \in \{t+1, \ldots, T+1\}$. Therefore, when Ω reduces over the time, there is no need to restart Algorithm 2. It will output the set $\mathrm{PO}_{\Omega_{T+1}}(\mathcal{X})$ at the end of the process. This remark suggests the possibility of inserting

preference queries in the search algorithm so as to obtain additional constraints on Ω, thus enabling a faster focus on the preferred solution paths.

Query Selection Strategies. We use query selection strategies based on the Minimax Regret criterion so as to reduce the uncertainty. Minimax Regret is a decision criterion usually used for decision making under total uncertainty. It is also a useful criterion to select preference queries as shown in (Wang and Boutilier 2003; Boutilier et al. 2006). According to this decision criterion, the most promising cost vector $x \in \mathcal{X}$ given the uncertainty set Ω is characterized by the following definitions of regrets, for all $x, y \in \mathcal{X}$:

Definition 3.

Pairwise Max Regret: $\mathrm{PMR}(x, y; \Omega) = \max_{\omega \in \Omega} \{\omega \cdot x - \omega \cdot y\}$

Max Regret: $\mathrm{MR}(x, \mathcal{X}; \Omega) = \max_{y \in \mathcal{X}} \mathrm{PMR}(x, y; \Omega)$

Minimax Regret: $\mathrm{MMR}(\mathcal{X}; \Omega) = \min_{x \in \mathcal{X}} \mathrm{MR}(x, \mathcal{X}; \Omega)$

$\mathrm{MR}(x, \mathcal{X}; \Omega)$ is the worst-case regret of choosing x instead of any $y \in \mathcal{X}$. According to the Minimax Regret criterion, the optimal cost vectors (named MMR-vectors hereafter) are those minimizing MR, i.e. those achieving the MMR value. Choosing a MMR-vector allows one to guarantee that the worst-case loss is minimized. Given the set Ω, the worst-case loss measured by MMR might still be too large for certifying the quality of the solution. Therefore, the Minimax Regret criterion can be used to select the most effective queries to reduce the MMR-value (the answers will further restrict the set Ω). Ideally, we would like to obtain $\mathrm{MMR} = 0$, which corresponds to the identification of a *necessarily optimal* solution, i.e. a solution which is optimal for all remaining $\omega \in \Omega$. This idea was successfully implemented in (Wang and Boutilier 2003; Boutilier et al. 2006) by the *Current Solution Strategy* (CSS). The CSS consists in generating a preference query asking the DM to compare two potentially good cost vectors in $\mathcal{X}$: an MMR-vector x^* and another vector y maximizing $\mathrm{PMR}(x^*, y; \Omega)$. The set Ω is then reduced by inserting the linear constraint induced by the answer, so as to keep consistency with DM's preferences. The CSS is based on the repeated computation of $\mathrm{PMR}(x, y; \Omega)$ for many pairs (x, y) of feasible solutions (all in the worst-case), which may induce prohibitive computation times in our context due to the size of $\mathcal{X}$. Therefore, instead of computing $\mathcal{X}$ and then applying this elicitation scheme, we propose to integrate the CSS to the search performed by Algorithm 2. We present now two strategies (S1 and S2) for generating queries during the search so as to obtain a set Ω sufficiently small to identify a necessarily optimal solution.

Strategy S1 aims at using new preference statements to better select the next label to expand. To this end, the call to SELECT($\mathcal{O}$) in Algorithm 2 line 5 selects the next label to expand by repeatedly asking queries applying the CSS over the set $X = \{g_\ell + h : \ell \in \mathcal{O}, h \in H(n_\ell)\}$ of heuristic cost vectors until the MMR equals zero. Note that if $\mathrm{MMR}(X; \Omega)$ is initially equal to zero, no query is asked during the call to SELECT($\mathcal{O}$). Then, the label $\ell^* \in \mathcal{O}$ selected for expansion corresponds to an element of X with a null MR value, i.e. the label ℓ^* is such that $\mathrm{MR}(g_{\ell^*} + h, X; \Omega) = 0$ for some $h \in H(n_{\ell^*})$. Thus, using this strategy amounts to only extending paths associated with a necessarily optimal heuristic cost vector. Finally, if the MMR value of $X = \{g_\ell : \ell \in \mathcal{S}\}$ is strictly larger than zero at the end of Algorithm 2, then S1 iteratively applies the CSS over X until the MMR equals zero; this ensures that there exists a necessarily optimal solution path in the set $\mathcal{S}$ returned by Algorithm 2.

Strategy S2 only uses new preference statements to keep the MMR of $X = \{g_\ell : \ell \in \mathcal{S}\}$ at value zero. S2 ensures that there exists, at any time of Algorithm 2, a solution that is necessarily optimal in $\mathcal{S}$. More precisely, whenever the insertion of ℓ^* in $\mathcal{S}$ (line 8) makes the MMR value of X be strictly larger than zero, then S2 repeatedly asks queries applying the CSS over X until the MMR equals zero. Hence, Algorithm 2 outputs a necessarily optimal solution.

Note that, in both strategies, the DM is only asked to compare pairs of vectors associated with complete paths from s to a goal node which makes sense. Although these cost vectors are heuristic evaluations of solution paths in strategy S1 they represent actual costs of solution paths in strategy S2. Hence in strategy S2, queries could be complemented by the presentation of paths under consideration to the DM to better ground her decision. This option seems less appropriate in S1 because it involves heuristic estimations of costs rather than actual costs of solution paths.

Algorithm 2 combined with strategy S1 or S2 returns a necessarily optimal solution. However it should be seen as a theoretical answer to our multiobjective search problem because in practice, making the MMR decrease to 0 at any stage of the procedure may entail a prohibitive number of queries. For a practical implementation of this procedure, we slightly relax the constraint on the MMR value by only requiring that MMR remains below a positive threshold λ. Let us develop this idea.

Search Implementation Let us relax the notion of f_ω-optimality by considering, in any set of vectors X, the set of *near-optimal* solutions defined by $\{x \in X : f_\omega(x) - \min_{y \in X} f_\omega(y) \leq \lambda\}$, for some given threshold $\lambda > 0$. This leads us to introduce the following:

Definition 4. *The set of possibly near-optimal vectors in X is:* $\mathrm{PO}^\lambda_\Omega(X) = \bigcup_{\omega \in \Omega} \{x \in X : f_\omega(x) - \min_{y \in X} f_\omega(y) \leq \lambda\}$.

As for $\mathrm{PO}_\Omega(\mathcal{X})$, the queries generated during the search of the possibly near-optimal solution paths will reduce Ω and therefore $\mathrm{PO}^\lambda_\Omega(X)$ until a *necessarily near-optimal* vector $x \in X$ is found, i.e. such that $f_\omega(x) - \min_{y \in X} f_\omega(y) \leq \lambda$ for all remaining $\omega \in \Omega$. Note that, by definition, a necessarily near-optimal vector in X has a MR value strictly lower than λ. Let us explain how Algorithm 2 and strategies S1 and S2 must be adapted to compute a necessarily near-optimal solution path. Consider the following dominance:

Definition 5. *For all $Y, Z \subseteq \mathbb{R}^q_+$:*

$Z \prec^\lambda_\Omega Y$ *if and only if* $\forall y \in Y, \forall \omega \in \Omega, \exists z \in Z, \omega \cdot y - \omega \cdot z > \lambda$

This relation is useful to determine $\mathrm{PO}^\lambda_\Omega$ due to the following result that includes Proposition 1 as special case ($\lambda = 0$).

Proposition 6. *If there exists $Y, Z \subseteq X$ such that $Z \prec_\Omega^\lambda Y$ then $Y \cap \mathrm{PO}_\Omega^\lambda(X) = \emptyset$.*

The proof is deliberately omitted, it is very similar to that of Proposition 1. Moreover, $\prec_\Omega^\lambda$ obviously shares several useful properties with $\prec_\Omega$, including transitivity and additivity. Therefore $\prec_\Omega^\lambda$ is to $\mathrm{PO}_\Omega^\lambda$ as $\prec_\Omega$ is to PO_Ω. Hence, if $\prec_\Omega$ is substituted by $\prec_\Omega^\lambda$ then Algorithm 1 returns $\mathrm{PO}_\Omega^\lambda(X)$ instead of $\mathrm{PO}_\Omega(X)$; consequently, if the same substitution is performed in R1 and R2 then Algorithm 2 returns $\mathrm{PO}_\Omega^\lambda(\mathcal{X})$ instead of $\mathrm{PO}_\Omega(\mathcal{X})$. Furthermore, our elicitation strategies can be used during the search to reduce the uncertainty set Ω so as to determine a necessarily near-optimal solution path. It is sufficient to modify S1 and S2 so that MMR values are not required to be equal to zero, but only to be lower than λ.

Numerical Tests

In this section, we report various numerical tests aiming at evaluating the performance of elicitation procedures S1 and S2 within Algorithm 2, both in terms of computation time and number of preference queries. These tests are used to determine a necessarily near-optimal path for the approximation threshold $\lambda = 0.01$ (meaning that the MMR value will be below 0.01). We consider instances of $G = (N, A)$ with one goal node γ and generated as follows: nodes in N are uniformly drawn in the two dimension grid $\{1, \ldots, 1000\} \times \{1, \ldots, 1000\}$, except γ and the source node s which are respectively located in $(1000, 500)$ and $(1, 500)$. Then, each generated node is linked to its five nearest nodes by arcs uniformly drawn in $\{0, \ldots, 100\}^q$. As admissible heuristic H, we consider that $H(n)$ only contains the ideal point $I(n) = (I_1, \ldots, I_q)$ for all $n \in N$, where $I_k = \min_{x \in \{g(p): p \in P(n,\gamma)\}} x_k$ for all $k \in Q$. Simulated DMs answer to queries according to a linear scalarizing function f_ω, where ω is randomly chosen in $\{\mathrm{int}(\mathbb{R}_+^q) : \sum_{i \in N} \omega_i = 1\}$.

The tests aim at estimating the impact of q (the number of criteria, see Table 1) and the impact of $|N|$ (the number of nodes, see Table 2) on the performance of Algorithm 2 implemented with S1 or S2. For comparison, we also consider the two-stage procedure (named S0 hereafter) consisting in first running MOA* and then applying the CSS over the resulting set of Pareto-optimal cost vectors. In Tables 1 and 2, the computation time is given in minutes and results are obtained by averaging over 30 runs. Linear optimizations are performed using the Gurobi library of Java.

Looking at the results for S1, we first see that Algorithm 2 used with S1 is faster than S0 when we consider more than three criteria (see Table 1). In particular, S1 is almost 100 times faster than S0 for $q = 4, 5$. Moreover, the number of queries tends to be smaller than that of S0 as q increases: S1 indeed generates 70%, 51%, 43% and 20% more queries than S0 when q is respectively equal to $2, 3, 4$ and 5. Besides, we can see that Algorithm 2 combined with S1 is much faster than S0 on bigger instances (see Table 2): S1 is almost five times faster than S0 and generates 8% less queries for $|N| = 500$.

Looking now at the results obtained for S2, we first see that Algorithm 2 used with S2 generates less preference queries than S0. For instance, S2 generates 44% less queries for $q = 5$ (see Table 1) and save about 25% of queries on the bigger instances (see Table 2); therefore S2 significantly reduces the DM's burden. However, computation times are worse with S2, especially on small instances and when considering two or three criteria. Overall, Algorithm 2 combined with strategy S1 turns out to be faster but S2 may still be preferred to minimize the number of queries.

Table 1: Impact of q the number of criteria ($|N| = 200$).

	$q=2$		$q=3$		$q=4$		$q=5$	
	time	queries	time	queries	time	queries	time	queries
S0	0.02	3.7	1.3	8.3	131.5	12.5	192.7	23.5
S1	0.5	6.3	1.1	12.5	1.5	17.9	2.2	28.1
S2	2.9	3.5	25.1	6.8	78.2	10.2	603.2	13.1

Table 2: Impact of $|N|$ the number of nodes ($q = 3$).

	$\|N\|=100$		$\|N\|=300$		$\|N\|=400$		$\|N\|=500$	
	time	queries	time	queries	time	queries	time	queries
S0	0.2	7.7	6	9.4	15.8	10.4	28.8	11.5
S1	0.32	12.3	8.3	13.2	3.8	11.6	5.8	10.6
S2	2.4	6.5	158.4	7.7	65.6	9.1	154.4	8.3

Conclusion and Perspectives

We have presented a new approach to multiobjective preference-based search where the elicitation of criteria weights is performed incrementally during the search. We have proposed, implemented and tested several elicitation strategies enabling a balanced tradeoff between two conflicting objectives: minimizing the number of queries and maximizing the quality of the recommendation. The tests show that, in multiobjective state space search problems, performing the elicitation during the search is better than first generating the Pareto set with MOA* and then applying standard elicitation techniques on the Pareto set. Moreover, the number of queries needed to make a recommendation with the desired guarantee remains low despite the combinatorial nature of the problem.

The next step is now to extend our approach to work with a non-linear f_ω function so as to obtain better fitting capacities to the DM's preferences and enhance the possibilities of exploring the Pareto set. For example, an incremental elicitation procedure for Choquet integrals has been recently proposed in (Benabbou, Perny, and Viappiani 2014) for explicit sets of alternatives. However, its extension to multiobjective state-space search raises challenging algorithmic questions. The use of a Choquet integral indeed complicates the definition of pruning rules during the search as well as the exploitation of new preference information.

Finally, integrating incremental preference elicitation into other general solution schemes (e.g. Branch and Bound, And/Or search) would also be worth investigating.

Acknowledgments

Work supported by the French National Research Agency through the Idex Sorbonne Universités under grant ANR-11-IDEX-0004-02.

References

Benabbou, N.; Perny, P.; and Viappiani, P. 2014. Incremental elicitation of Choquet capacities for multicriteria decision making. In *21st European Conference on Artificial Intelligence*, 87–92.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. Preference-based constrained optimization with cp-nets. *Computational Intelligence* 20(2):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 Journal* 170(8-9):686–713.

Brafman, R. I., and Domshlak, C. 2009. Preference handling - an introductory tutorial. *AI Magazine* 30(1):58–86.

Braziunas, D., and Boutilier, C. 2007. Minimax regret based elicitation of generalized additive utilities. In *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, 25–32.

Dasgupta, P.; Chakrabarti, P.; and DeSarkar, S. 1995. Utility of *pathmax* in partial order heuristic search. *J. of algorithms* 55:317–322.

Domshlak, C.; Hüllermeier, E.; Kaci, S.; and Prade, H. 2011. Preferences in AI: an overview. *Artificial Intelligence Journal* 175(7-8):1037–1052.

Drummond, J., and Boutilier, C. 2014. Preference elicitation and interview minimization in stable matchings. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 645–653.

Galand, L., and Perny, P. 2006. Search for Compromise Solutions in Multiobjective State Space Graphs. In *17th European Conference on Artificial Intelligence*, 93–97.

Galand, L., and Perny, P. 2007. Search for Choquet-optimal paths under uncertainty. In *Proceedings of the 23rd conference on Uncertainty in Artificial Intelligence*, 125–132.

Gonzales, C., and Perny, P. 2004. GAI Networks for Utility Elicitation. In *Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning*, 224–234.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Man, and Cybernetics* 4(2):100–107.

Koriche, F., and Zanuttini, B. 2010. Learning conditional preference networks. *Artif. Intell.* 174(11):685–703.

Mandow, L., and De la Cruz, J. L. P. 2005. A new approach to multiobjective A* search. In *International Joint Conference on Artificial Intelligence*, 218–223.

White III, C. C.; Sage, A. P.; and Dozono, S. 1984. A model of multiattribute decisionmaking and trade-off weight determination under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics* 14(2):223–229.

Regan, K., and Boutilier, C. 2011. Eliciting additive reward functions for markov decision processes. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2159–2164.

Stewart, B. S., and White III, C. C. 1991. Multiobjective A*. *Journal of ACM* 38(4):775–814.

Vanderpooten, D., and Vincke, P. 1997. Description and analysis of some representative interactive multicriteria procedures. *Appl. Math. Comp.* 83(2-3):261–280.

Wang, T., and Boutilier, C. 2003. Incremental Utility Elicitation with the Minimax Regret Decision Criterion. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 309–316.

Weng, P., and Zanuttini, B. 2013. Interactive Value Iteration for Markov Decision Processes with Unknown Rewards. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2415–2421.

Zionts, S., and Wallenius, J. 1976. An interactive programming method for solving the multiple criteria problem. *Management Science* 22(6):652–663.