Interactive resolution of multiobjective combinatorial optimization problems by incremental elicitation of criteria weights

Nawal Benabbou, Patrice Perny

EURO Journal on Decision Processes manuscript No.
(will be inserted by the editor)

Interactive Resolution of Multiobjective Combinatorial Optimization Problems by Incremental Elicitation of Criteria Weights

Nawal Benabbou · Patrice Perny

Received: date / Accepted: date

Abstract We propose an introduction to the use of incremental preference elicitation methods in the field of multiobjective combinatorial optimization. We consider three different optimization problems in vector-valued graphs, namely the shortest path problem, the minimum spanning tree problem and the assignment problem. In each case, the preferences of the decision maker over cost vectors are assumed to be representable by a weighted sum but the weights of criteria are initially unknown. We then explain how to interweave preference elicitation and search in order to quickly determine a near-optimal solution with a limited number of preference queries. This leads us to successively introduce an interactive version of dynamic programming, greedy search, and branch and bound to solve the problems under consideration. We then present numerical tests showing the practical efficiency of these algorithms that achieve a good compromise between the number of queries asked and the solution times.

Keywords Multiobjective Combinatorial Optimization · Preference Elicitation · Imprecise Weights · Minimax Regret

1 Introduction

The increasing complexity of applications encountered in multicriteria optimization, multiagent optimization and optimization under uncertainty leads us today to apply decision models to combinatorial sets of solutions which are implicitly defined. This significantly complicates the decision process and, in particular, the construction of a preference model fitting the objectives of the Decision Maker

Nawal Benabbou
National University of Singapore,
NUS School of Computing, COM1, 13 Computing Drive, 117417, Singapore.
E-mail: benabbou@comp.nus.edu.sg

Patrice Perny
CNRS, Laboratoire d'informatique de Paris 6, LIP6,
Sorbonne Université, 75005 Paris, France.
E-mail: patrice.perny@lip6.fr

(DM) as well as the calculation of the optimal decision. In the recent years, these difficulties have motivated the development of algorithmic decision theory as a research field which constitutes the reference frame of this paper.

We consider here new algorithms for the interactive elicitation of preferential parameters in a decision model and the efficient calculation of optimal solutions with respect to this model. Instead of approaching independently and separately these two subjects, we introduce incremental decision procedures aiming to integrate and combine the elicitation of preferences and the calculation of the preferred solution in order to determine the optimal choice without fully specifying the decision model. In these new interactive resolution schemes, asking questions about the DM's preferences during the exploration of the set of solutions allows to focus the elicitation of preferences on information that is really useful for separating competing solutions and thus reducing the number of questions required. This is the main benefit of the incremental approach to decision making.

The incremental approach has already been successfully considered for preference elicitation in non-combinatorial problems in various contexts such as multiattribute utility theory (White III et al, 1984; Braziunas and Boutilier, 2007), multicriteria decision making (Benabbou et al, 2015, 2017), decision making under risk (Ha and Haddawy, 1997; Chajewska et al, 2000; Wang and Boutilier, 2003; Hines and Larson, 2010; Perny et al, 2016; Gilbert et al, 2017) and collective decision making (Lu and Boutilier, 2011; Dery et al, 2014; Benabbou et al, 2016). Preference elicitation on combinatorial domains is a challenging issue that recently motivated several contributions in various contexts such as constraint satisfaction (Gelain et al, 2010), committee election (Benabbou and Perny, 2016), matching (Drummond and Boutilier, 2014), sequential decision making under risk (Regan and Boutilier, 2009; Weng and Zanuttini, 2013; Gilbert et al, 2015; Benabbou and Perny, 2017), multiattribute decision making (Koriche and Zanuttini, 2010) and multicriteria optimization (Benabbou and Perny, 2015b,c; Kaddani et al, 2017; Bourdache and Perny, 2017). In this paper we focus on preference-based search for multiobjective combinatorial optimization with the aim of explaining how preference queries can be inserted in standard combinatorial optimization algorithms for both increasing our knowledge of the DM's preferences and facilitating the construction of the optimal solution. This approach is incremental in the sense that we learn the decision model and the notion of optimality during the exploration of the set of feasible solutions and not in a preliminary step.

Our purpose is to introduce the general principle of progressive regret reduction present in many incremental elicitation approaches and to explain how it can be efficiently implemented within combinatorial algorithms to solve different types of multiobjective optimization problems involving multiple criteria. For the sake of simplicity, we here assume that DM's preferences are representable by a weighted sum of criterion values but the criteria weights are initially unknown. In this context, we study the incremental elicitation of criteria weights so as to efficiently determine a (near-)optimal solution.

The paper is organized as follows. In Section 1, we recall the basic principles of incremental elicitation methods based on regret-minimization. In particular, we recall how worst-case regrets can be used for the active selection of preference queries in incremental decision methods and how they can support the determination of robust recommendations under imperfect knowledge of preferences. Then the three following sections explain how preference queries used for weight elicita-

tion can be interleaved with a combinatorial optimization algorithm progressively constructing an optimal solution. The integration of preference elicitation steps during the search is presented for three different solution methods classically used in combinatorial optimization, namely multiobjective dynamic programming applied to state space search (in Section 2), multiobjective greedy search applied to the minimum spanning tree problem (in Section 3), and multiobjective branch and bound applied to the multiobjective assignment problem (in Section 4). In all these sections, we introduce incremental elicitation strategies during the search that use preference queries to progressively reduce the set of admissible weights until a (near-)optimal solution can be identified. The validity of our algorithms is established and numerical tests are provided to show their efficiency both in terms of number of generated preference queries and solution times.

2 Incremental Elicitation based on the Minimax Regret Criterion

In this section, we first introduce a formal framework for multicriteria combinatorial optimization problems where the weights of criteria are not precisely known. Then we recall how the minimax regret criterion can be used for decision making under imperfect knowledge of criterion weights, using an incremental elicitation procedure to determine a (near-)optimal solution without asking too many queries in practice.

2.1 Preliminaries

Let $\mathcal{X}$ denote the set of solutions that need to be compared in order to make a decision; in this paper, we assume that $\mathcal{X}$ is implicitly defined as the set of feasible solutions of a multiobjective combinatorial optimization problem. More precisely, we consider a set of q objective functions to be minimized simultaneously (e.g., time, cost, distance). Denoting x_j the evaluation of any solution $x \in \mathcal{X}$ on criterion j, $j \in \mathcal{Q} = \{1, \ldots, q\}$, every solution $x \in \mathcal{X}$ is characterized by the cost vector $(x_1, \ldots, x_q)$. For simplicity, throughout the paper, x will indifferently denote a solution or its performance vector. Moreover, we assume that the DM's preferences are representable by a linear function $f_\omega(x) = \sum_{j=1}^{q} \omega_j x_j$ measuring the overall cost of any solution $x \in \mathcal{X}$ and that the weighting vector $\omega = (\omega_1, \ldots, \omega_q)$ is initially not known or only imprecisely known.

In this context, we consider the set Ω containing all admissible normalized weighting vectors ω at a given step of the decision process. By default, Ω is initially defined as the simplex $\{\omega \in \text{int}(\mathbb{R}^q_+) : \sum_{j=1}^{q} \omega_j = 1\}$ where $\text{int}(\mathbb{R}^q_+)$ represents the interior of the cone $\mathbb{R}^q_+$. Later in the decision process, Ω may represent the subset of normalized weighting vectors that are still compatible with the set of preference statements obtained from the DM. Due to the linearity of $f_\omega(x)$ in ω for any fixed cost vector x, any preference statement of type "x is at least as good as y" induces the linear constraint $f_\omega(x) \leq f_\omega(y)$ over the simplex. As a consequence, throughout this paper, we can assume that Ω is a convex bounded polyhedron without loss of generality.

2.2 Minimax Regret

The Minimax regret criterion is a decision criterion standardly used in optimization under total uncertainty and in robust optimization (Savage, 1954; Kouvelis and Yu, 1997). This criterion has been more recently advocated to make robust decisions under imprecise utilities (Salo and Hämäläinen, 2001; Boutilier et al, 2006). In our context, the minimax regret decision criterion can be used to determine a robust solution given the parameter imprecision defined by Ω. It is formally defined using the following definitions of regrets:

Definition 1. *The pairwise max regret (PMR) of solution $x \in \mathcal{X}$ with respect to solution $y \in \mathcal{X}$ is defined as:*

$$\mathrm{PMR}(x, y, \Omega) = \max_{\omega \in \Omega} \{f_\omega(x) - f_\omega(y)\}$$

By definition, the pairwise max regret of solution x with respect to solution y represents the worst-case loss that can be induced by recommending x instead of y to the DM. In particular, solution x is necessarily preferred to solution y whenever we have $\mathrm{PMR}(x, y, \Omega) \leq 0$; in this case, we indeed have $f_\omega(x) \leq f_\omega(y)$ for all $\omega \in \Omega$. Note that pairwise max regrets can be efficiently computed by means of linear programming. The max regret is then derived from pairwise max regrets as follows:

Definition 2. *The max regret (MR) of solution $x \in \mathcal{X}$ is defined as:*

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

In other words, the max regret associated to x represents the worst-case loss that one may obtain when recommending solution x instead of any other solution, and in particular any adversary's choice in $\arg\max_{y \in \mathcal{X}} \mathrm{PMR}(x, y, \Omega)$. Then comes the minimax regret:

Definition 3. *The minimax regret (mMR) is defined as:*

$$\mathrm{mMR}(\mathcal{X}, \Omega) = \min_{x \in \mathcal{X}} \mathrm{MR}(x, \mathcal{X}, \Omega)$$

An optimal solution with respect to the minimax regret is a solution $x \in \mathcal{X}$ that minimizes the maximum regret (i.e., any element of $\arg\min_{x \in \mathcal{X}} \mathrm{MR}(x, \mathcal{X}, \Omega)$). This criterion enables to protect one against the worst case scenario. It is a bit conservative but leads to robust recommendations under imperfect knowledge of preferences.

2.3 Incremental Regret-based Preference Elicitation

Given a set Ω of admissible weighting vectors, the worst-case loss ensured by the recommendation of any optimal solution for the minimax regret criterion might still be too large for certifying the quality of the solution. Note that this worst-case loss may decrease by considering additional preference statements obtained from

the DM since they induce new constraints on the set of admissible parameters. For any set $\Omega' \subseteq \Omega$, we indeed have:

$$\mathrm{PMR}(x, y, \Omega') \leq \mathrm{PMR}(x, y, \Omega), \forall x, y \in \mathcal{X} \quad (1)$$

$$\mathrm{MR}(x, \mathcal{X}, \Omega') \leq \mathrm{MR}(x, \mathcal{X}, \Omega), \forall x \in \mathcal{X} \quad (2)$$

$$\mathrm{mMR}(\mathcal{X}, \Omega') \leq \mathrm{mMR}(\mathcal{X}, \Omega) \quad (3)$$

These inequalities show that the minimax regret cannot increase by collecting new preference statements from the DM; in practice, this value often strictly decreases when preference queries are carefully chosen (see e.g., Braziunas (2011)). Hence, this decision criterion can be used within an incremental elicitation process that progressively asks preference queries to the DM until the minimax regret drops below a given tolerance threshold $\delta \geq 0$. At that moment, recommending any optimal solution for the minimax regret criterion ensures that the loss incurred by not choosing the true optimal solution is bounded above by that threshold. Ideally, we would like to ask preference queries until the minimax regret is equal to 0, which corresponds to the identification of an optimal solution. However, to reduce the elicitation burden, it is often more efficient to consider a threshold $\delta > 0$ representing the maximum admissible gap to optimality.

Choosing good queries is essential in order to be able to reduce the minimax regret reasonably quickly (i.e. asking only few queries). Different types of preference queries can be used when designing such an incremental elicitation process. Just to name a few examples, bound queries ask if the overall utility (aggregate value) of a solution is higher or lower than a given reference value whereas comparison queries require the DM to compare a pair of solutions and state which one is preferred among the two. Bound queries are only necessary when we want to construct an absolute evaluation scale in which some numeric values have a special meaning and serve as references. In this paper we prefer resorting to comparison queries because they only rely on relative assessments; this is sufficient to determine an optimal solution or to rank the solutions of a decision problem. Even when the type of query is fixed, there are of course several possible queries that can be asked at each step. Notice however that some preference queries are less informative than others. For instance, the minimax regret will not change after asking to compare a solution with another one that is Pareto-dominated by the former. Instead, asking to compare two potentially good solutions is much more likely to constitute a good preference query. This idea was successfully implemented in (Wang and Boutilier, 2003; Boutilier et al, 2006) by the query generation strategy called *Current Solution Strategy* (CSS). The CSS consists in asking the DM to compare, at each iteration step, an optimal solution x^* for the minimax regret decision criterion with one of its adversarial choices y^* (arbitrarily chosen in $\arg\max_{y \in \mathcal{X}} \mathrm{PMR}(x^*, y, \Omega)$)[1].

At each iteration step, the minimax regret can be determined by computing the value $\mathrm{PMR}(x, y, \Omega)$ for all ordered pairs of distinct solutions in $\mathcal{X}$ (see Definition 3). Even if the computational effort can be significantly reduced by using standard pruning rules for min aggregators (as shown in Braziunas (2011)), the number of

[1] Using the CSS, we will necessarily reach a point where we have $\mathrm{mMR}(\mathcal{X}, \Omega) \leq \delta$ for any threshold $\delta \geq 0$. Indeed, at each iteration step, we can always choose x^* and y^* such that we have $\mathrm{PMR}(x^*, y^*, \Omega) > 0$ and $\mathrm{PMR}(y^*, x^*, \Omega) > 0$ before asking the query, and after collecting the DM's answer, one of these values will become smaller or equal to zero. Then the result derives from the fact that the set $\mathcal{X}$ of solutions is finite.

PMR-computations remains quadratic in the worst-case. This approach therefore induces prohibitive computation times when the set $\mathcal{X}$ is composed of all solutions of a combinatorial optimization problem. In this context, instead of determining the set $\mathcal{X}$ and then applying this elicitation scheme, we propose to integrate an incremental elicitation method to the search procedure: starting with an initial set Ω of possible weights, we generate preference queries during the search so as to progressively reduce the set Ω until being able to determine a near-optimal solution (i.e. a solution $x \in \mathcal{X}$ such that $\text{MR}(x, \mathcal{X}, \Omega) \leq \delta$). The motivation behind our proposal is to:

- save computation time by using the DM's answers to quickly focus the search on the most promising solutions and
- reduce the elicitation burden by generating preference queries only to discriminate between some solutions found during the search.

In the following sections, we will see that integrating incremental preference elicitation into dynamic programming (Section 3), greedy (Section 4) and branch and bound (Section 5) algorithms enables us to save both computation time and preference queries needed to determine a near-optimal solution in practice.

3 Multiobjective State Space Search

Preference-based search is an active topic in Operations Research and Artificial Intelligence with various applications to constraint satisfaction, planning, search, resource allocation and electronic commerce (e.g., Boutilier et al (2004); Brafman and Domshlak (2009); Domshlak et al (2011)). In this section, we study the potential of incremental elicitation methods 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 feasible solutions are very numerous and defined implicitly as the paths starting from a source node to a goal node in a state space graph. In vector-valued graphs, the set of Pareto non-dominated cost vectors attached to feasible paths grows, in some family of instances, exponentially with the size of the problem (Hansen, 1980), which precludes any explicit enumeration. Assuming that the DM's preferences can be represented by a weighted sum, we propose here to combine incremental preference elicitation and state space search so as to efficiently determine a (near-)optimal solution with a limited number of preference queries.

3.1 The General Framework

Let $G = (N, A)$ be a state space graph where N is the finite set of nodes representing the different states and A is the set of arcs representing the feasible state transitions. Formally, $A = \{(n, n') : n \in N, n' \in \Pi(n)\}$ where $\Pi(n) \subseteq N$ is the set of all nodes that can be reached from node n by a single transition. A path between n and n' is characterized by a list of nodes of type $\langle n_1, \ldots, n_k \rangle$ where $n_1 = n$, $n_k = n'$ and $n_{i+1} \in \Pi(n_i)$ for all $i \in \{1, \ldots, k-1\}$. The set of all paths between node n and node n' will be denoted by $P(n, n')$ in the sequel; in particular, the set of all solution paths, starting at the source node $s \in N$ and reaching any element of the

set $\Gamma \subset N$ of goal nodes, will be denoted by $P(s,\Gamma)$. Besides, we consider a finite set of criteria $g_j : A \to \mathbb{R}_+$, with $j \in \mathcal{Q} = \{1,\ldots,q\}$, that need to be minimized (e.g., distance, time, travel cost). Hence, graph G is endowed with a vector valuation function $g : A \to \mathbb{R}^q_+$ which assigns the cost vector $g(a) = (g_1(a),\ldots,g_q(a))$ to each arc $a \in A$. Moreover, each path p in graph G is associated with a cost vector $g(p) = (g_1(p),\ldots,g_q(p))$ which is defined by $g_j(p) = \sum_{(n,n')\in p} g_j((n,n'))$ for all $j \in \mathcal{Q}$. Finally, the set of feasible cost vectors is denoted by $\mathcal{X} = \{g(p) : p \in P(s,\Gamma)\}$. This set represents the image of all solution paths in the space of criteria.

In this section, we consider the problem of finding a near-optimal solution path, i.e. an optimal solution path for the minimax regret decision criterion such that the gap to optimality, quantified by the minimax regret, is bounded above by threshold δ. First, given a set of Ω of admissible weighting vectors, we propose a search procedure that enables to detect paths that cannot be part of a solution path with a max regret below δ. Then, we propose incremental procedures that reduce Ω during the search by alternating preference elicitation steps and search steps so that a near-optimal solution path is returned at the end of the search with a limited number of queries.

3.2 Search of Possibly Near-Optimal Solution 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 et al, 1968) that determines the set $\mathrm{ND}(\mathcal{X})$ of Pareto non-dominated cost vectors attached to paths in $P(s,\Gamma)$ and returns one path for each element of $\mathrm{ND}(\mathcal{X})$. Formally $\mathrm{ND}(\mathcal{X}) = \{x \in \mathcal{X} : \forall y \in \mathcal{X}, \mathrm{not}(y \prec_P x)\}$ 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 $x_j \leq y_j$ for all $j \in \mathcal{Q}$ and $x_j < y_j$ for some $j \in \mathcal{Q}$. Since all preference models considered in multicriteria analysis are compatible with Pareto dominance or weak Pareto dominance (defined by $x \precsim_P y$ if and only if $x_j \leq y_j$ for all $j \in \mathcal{Q}$), the MOA* algorithm is a useful basis to develop more specific preference-based search procedures. We introduce now a search procedure based on recent variants of MOA* (Mandow and De la Cruz, 2005) which uses new pruning rules that are able to detect paths that cannot lead to solution paths with a max regret below the tolerance threshold δ.

In vector-valued graphs, there possibly exist several optimal paths with different cost vectors to reach a given node in the graph. Therefore, the basic graph exploration procedure consists in iteratively expanding labels attached to subpaths rather than nodes, labels being of the form $\ell = [n_\ell, p_\ell, g_\ell]$ where p_ℓ is a path from node s to node n_ℓ and $g_\ell = g(p_\ell)$ denotes its cost vector (with a slight abuse of notation). At every iteration of the search procedure, a label is selected for expansion. The expansion of a label ℓ^* generates the set of its successors $\{[n, p_{\ell^*} \circ \langle n\rangle, g(p_{\ell^*} \circ \langle n\rangle)] : n \in \Pi(n_{\ell^*})\}$, where $\circ$ is the path appending operator. At any time, the set of generated labels is divided into two disjoint sets: a set $\mathcal{C}$ of *closed* labels (already 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 $p_\ell \in P(s,n)$ will be denoted by $\mathcal{C}(n)$ (resp. $\mathcal{O}(n)$) in the sequel. During the search, all expanded labels corresponding to solution paths are stored in a set denoted by $\mathcal{S}$, the correspond-

ing set of cost vectors being denoted by $g_{\mathcal{S}}$ hereafter. The pruning rules that we are proposing to detect paths that cannot be part of a solution path with a max regret below threshold δ are based on the following dominance relation:

Definition 4 ($\prec^{\delta}_{\Omega}$-dominance)**.** *For all $X, Y \subset \mathbb{R}^q_+$:*

$$X \prec^{\delta}_{\Omega} Y \Leftrightarrow \forall y \in Y, \forall \omega \in \Omega, \exists x \in X, f_\omega(y) - f_\omega(x) > \delta$$

This dominance relation, defined for all non-negative real vectors, satisfies the following convenient property:

Proposition 1. *For all $X, Y \subset \mathbb{R}^q_+$:*

$$X \prec^{\delta}_{\Omega} Y \Leftrightarrow \forall y \in Y, \min_{\omega \in \Omega} \max_{x \in X} \{f_\omega(y) - f_\omega(x)\} > \delta$$

Proof. ($\Rightarrow$) Let $X, Y \subset \mathbb{R}^q_+$ be two sets of vectors such that we have $X \prec^{\delta}_{\Omega} Y$ and let y be an element of Y. Then, for all $\omega \in \Omega$, there exists a vector $x \in X$ such that the inequality $f_\omega(y) - f_\omega(x) > \delta$ holds (by definition of relation $\prec^{\delta}_{\Omega}$). Therefore, we have $\max_{x \in X}\{f_\omega(y) - f_\omega(x)\} > \delta$ for all $\omega \in \Omega$, and in particular we have $\min_{\omega \in \Omega} \max_{x \in X}\{f_\omega(y) - f_\omega(x)\} > \delta$.
($\Leftarrow$) Consider two sets of vectors $X, Y \subset \mathbb{R}^q_+$ such that, for all vectors $y \in Y$, the inequality $\min_{\omega \in \Omega} \max_{x \in X}\{f_\omega(y) - f_\omega(x)\} > \delta$ holds. This inequality implies that we have $\max_{x \in X}\{f_\omega(y) - f_\omega(x)\} > \delta$ for all vectors $y \in Y$ and all weighting vector $\omega \in \Omega$. Hence, for all $y \in Y$ and all $\omega \in \Omega$, there exists a vector $x \in X$ such that $f_\omega(y) - f_\omega(x) > \delta$ holds. Thus, we have $X \prec^{\delta}_{\Omega} Y$. □

Since Ω is a convex polyhedron and $f_\omega(x)$ is linear in ω for any fixed $x \in \mathbb{R}^q_+$, this proposition enables to efficiently perform $\prec^{\delta}_{\Omega}$-dominance tests using linear programming. Then, the first pruning rule we are proposing relies on the following property of the $\prec^{\delta}_{\Omega}$-dominance relation:

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

$$X \prec^{\delta}_{\Omega} Y \Rightarrow X + Z \prec^{\delta}_{\Omega} Y + Z$$

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 two sets of vectors such that we have $X \prec^{\delta}_{\Omega} Y$. Let $Z \subseteq \mathbb{R}^q_+$ be another set of vectors. Let u be an element of the set $Y + Z$. Since $u \in Y + Z$, there exists a vector $y \in Y$ and a vector $z \in Z$ such that we have $u = y + z$. Let ω be any weighting vector in the set Ω. Since $X \prec^{\delta}_{\Omega} Y$ holds by hypothesis, there exists a vector $x \in X$ such that we have $f_\omega(y) - f_\omega(x) > \delta$, i.e. $f_\omega(y) + f_\omega(z) - f_\omega(x) - f_\omega(z) > \delta$. Then, by linearity of function f_ω, we obtain $f_\omega(y+z) - f_\omega(x+z) > \delta$, which can be rewritten $f_\omega(u) - f_\omega(x+z) > \delta$. Finally, since vector $x + z$ is an element of the set $X + Z$ by definition, the previous inequality establishes the result. □

This nice property enables to use a dynamic programming approach based on relation $\prec^{\delta}_{\Omega}$ as shown by the following proposition:

Proposition 3. *At any node $n \in N$, if there exists an open label $\ell' \in \mathcal{O}(n)$ and a set of labels $L \subseteq \mathcal{O}(n) \cup \mathcal{C}(n)$ such that $\{g_\ell : \ell \in L\} \prec^{\delta}_{\Omega} \{g_{\ell'}\}$, then path $p_{\ell'}$ cannot be part of a solution path with a max regret below threshold δ.*

Proof. Consider a node $n \in N$, a label $\ell' \in \mathcal{O}(n)$ and a set of labels $L \subseteq \mathcal{O}(n) \cup \mathcal{C}(n)$ such that we have $\{g_\ell : \ell \in L\} \prec^\delta_\Omega \{g_{\ell'}\}$. Let p be a path in $P(n, \Gamma)$ and Ω' be a subset of Ω. We want to prove that $\mathrm{MR}(g(p_{\ell'} \circ p), \mathcal{X}, \Omega') > \delta$ holds. Since the $\prec^\delta_\Omega$-dominance relation is additive (see Proposition 2), we derive $\{g_\ell : \ell \in L\} + \{g(p)\} \prec^\delta_\Omega \{g_{\ell'}\} + \{g(p)\}$. Hence, we have $\{g_\ell + g(p) : \ell \in L\} \prec^\delta_\Omega \{g_{\ell'} + g(p)\}$ which can be rewritten as follows: $\{g(p_\ell \circ p) : \ell \in L\} \prec^\delta_\Omega \{g(p_{\ell'} \circ p)\}$. Therefore, for all weighting vectors $\omega \in \Omega$, there exists a vector $x \in \{g(p_\ell \circ p) : \ell \in L\}$ such that we have $f_\omega(g(p_{\ell'} \circ p)) - f_\omega(x) > \delta$. Since $\Omega' \subseteq \Omega$ and $\{g(p_\ell \circ p) : \ell \in L\} \subseteq \mathcal{X}$, we know that the following statement is true: for all weighting vectors $\omega \in \Omega'$, there exists a solution $x \in \mathcal{X}$ such that $f_\omega(x) - f_\omega(x) > \delta$ holds. Hence, we have $\max_{\omega \in \Omega'} \max_{x \in \mathcal{X}} \{f_\omega(x) - f_\omega(x)\} = \mathrm{MR}(g(p_{\ell'} \circ p), \mathcal{X}, \Omega') > \delta$. □

This proposition says that we can discard any label $\ell' \in \mathcal{O}(n)$ during the search whenever $\{g_\ell : \ell \in L\} \prec^\delta_\Omega \{g_{\ell'}\}$ is verified by some subset $L \subseteq \mathcal{O}(n) \cup \mathcal{C}(n)$. This indeed ensures that path $p_{\ell'}$ cannot be completed into a solution path with a max regret below threshold δ, even if we further restrict the set of feasible weights Ω by asking preference queries to the DM. Note that, in practice, there is no need to enumerate and test all subsets L to know whether this proposition enables to discard label ℓ'. It is indeed sufficient to make the test for $L = \mathcal{O}(n) \cup \mathcal{C}(n)$ by definition of the $\prec^\delta_\Omega$-dominance relation. Therefore, we use here the following pruning rule:

Rule R1. *Discard all labels $\ell' \in \mathcal{O}(n)$ such that $\{g_\ell : \ell \in \mathcal{O}(n) \cup \mathcal{C}(n)\} \prec^\delta_\Omega \{g_{\ell'}\}$.*

Our search procedure imports another feature from MOA*: 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 $\{g(p) : p \in P(n_\ell, \Gamma)\}$. These sets are used here to define another pruning rule that is able to detect paths that necessarily lead to solution paths with a max regret strictly greater than δ. More precisely, as in MOA*, we assume that heuristic H is admissible, i.e. H provides an optimistic evaluation of the cost to reach the goal. This assumption is formalized as follows: for all nodes $n \in N$, for all paths $p \in P(n, \Gamma)$, there exists $h \in H(n)$ such that $h \precsim_P g(p)$. Under this assumption, the following proposition holds:

Proposition 4. *At any node $n \in N$, if there exists an open label $\ell' \in \mathcal{O}(n)$ such that $g_\mathcal{S} \prec^\delta_\Omega F(\ell')$ holds, then path $p_{\ell'}$ cannot be part of a solution path with a max regret below threshold δ.*

Proof. Consider a node $n \in N$ and an open label $\ell' \in \mathcal{O}(n)$ such that $g_\mathcal{S} \prec^\delta_\Omega F(\ell')$ holds. We want to prove that we have $\mathrm{MR}(g(p_{\ell'} \circ p'), \mathcal{X}, \Omega') > \delta$ for any path $p' \in P(n, \Gamma)$ and any set of weighting vectors $\Omega' \subseteq \Omega$. Since H is admissible, we know that there exists a heuristic cost vector $h' \in H(n)$ such that we have $h' \precsim_P g(p')$, and therefore $g_{\ell'} + h' \precsim_P g_{\ell'} + g(p') = g(p_{\ell'} \circ p')$. Since we have $f_\omega(x) \leq f_\omega(y)$ for any two vectors $x, y \in \mathbb{R}^q_+$ such that $x \precsim_P y$, then we derive $f_\omega(g_{\ell'} + h') \leq f_\omega(g(p_{\ell'} \circ p'))$ for all weighting vectors $\omega \in \Omega'$. Moreover, since $g_\mathcal{S} \prec^\delta_\Omega F(\ell') = \{g_{\ell'} + h : h \in H(n)\}$ holds, there exists a label $\ell \in \mathcal{S}$ such that we have $f_\omega(g_{\ell'} + h') - f_\omega(g_\ell) > \delta$ for all weighting vectors $\omega \in \Omega'$ (by Definition 4). Therefore, the inequality $f_\omega(g(p_{\ell'} \circ p')) - f_\omega(g_\ell) > \delta$ holds for all weighting vectors $\omega \in \Omega'$. Hence, we have $\max_{\ell \in \mathcal{S}} \{f_\omega(g(p_{\ell'} \circ p')) - f_\omega(g_\ell)\} > \delta$ for all vectors $\omega \in \Omega'$. Then, since we have $g_\mathcal{S} \subseteq \mathcal{X}$, we obtain $\max_{x \in \mathcal{X}} \{f_\omega(g(p_{\ell'} \circ p')) - f_\omega(x)\} > \delta$ for

all vectors $\omega \in \Omega'$. Therefore, we have $\max_{\omega\in\Omega'} \max_{x\in\mathcal{X}}\{f_\omega(g(p_{\ell'} \circ p')) - f_\omega(x)\} = \mathrm{MR}(g(p_{\ell'} \circ p'), \mathcal{X}, \Omega') > \delta$. □

Hence, if there exists an open label $\ell' \in \mathcal{O}$ such that $g_{\mathcal{S}} \prec^{\delta}_{\Omega} F(\ell')$ holds at some point of the search procedure, then Proposition 4 ensures that path $p_{\ell'}$ cannot be completed into a solution path with a max regret below threshold δ (even if we further restrict the set of admissible weights Ω). This result gives us the following pruning rule:

Rule R2. *Discard label* $\ell' \in \mathcal{O}(n)$ *if* $g_{\mathcal{S}} \prec^{\delta}_{\Omega} \{g_{\ell'}\} + H(n)$.

To summarize, we are proposing a search procedure which refines a MOA* search of Pareto-optimal cost vectors by using two additional pruning rules based on the $\prec^{\delta}_{\Omega}$-dominance relation, namely R1 and R2 (see Algorithm 1).

Algorithm 1: Multiobjective state space search with imprecise preferences

Input: $G = (N, A)$; s; Γ; H; Ω
Output: $\mathcal{S}$: a set of labels associated to solution paths

foreach $n \in N$ **do**
  $\mathcal{O}(n) \leftarrow \emptyset$
  $\mathcal{C}(n) \leftarrow \emptyset$
end
$\mathcal{O}(s) \leftarrow \{[s, \langle s\rangle, (0, \ldots, 0)]\}$
$\mathcal{S} \leftarrow \emptyset$
while $\mathcal{O} \neq \emptyset$ **do**
  Select ℓ^* in the set
    $L^* = \{\ell \in \mathcal{O} : \exists f \in F(\ell), \forall \ell' \in \mathcal{O}, \forall f' \in F(\ell'), not(f' \precsim_P f)\}$
  Move ℓ^* from $\mathcal{O}$ to $\mathcal{C}$
  if $n_{\ell^*} \in \Gamma$ *and* $\forall \ell \in \mathcal{S}$, $not(g_\ell \precsim_P g_{\ell^*})$ **then**
    Add ℓ^* to $\mathcal{S}$
  else
    foreach $n \in \Pi(n_{\ell^*})$ **do**
      Generate $\ell' = [n, p_{\ell^*} \circ \langle n\rangle, g_{\ell^*} + g((n_{\ell^*}, n))]$
      if $\forall \ell \in \mathcal{O}(n) \cup \mathcal{C}(n)$, $not(g_\ell \precsim_P g_{\ell'})$ **then**
        Add ℓ' to $\mathcal{O}(n)$
        Apply rule R2 to ℓ'
        if ℓ' *is not discarded* **then**
          Apply rule R1 to $\mathcal{O}(n)$
        end
      end
    end
  end
end
return $\mathcal{S}$

The following example presents a complete execution of Algorithm 1 on a small instance of $G = (N, A)$:

Example 1. *Consider the instance of $G = (N, A)$ given in Figure 1 with $\delta = 0$. With no preference information, the set Ω of admissible weights $\omega = (\omega_1, \omega_2)$ is formally defined by $\Omega = \{\omega \in int(\mathbb{R}^2_+) : \omega_1 + \omega_2 = 1\}$. Note that, due to normalization, Ω can also be defined as the set of vectors of type $(\omega_1, 1 - \omega_1)$ with $0 < \omega_1 < 1$.*

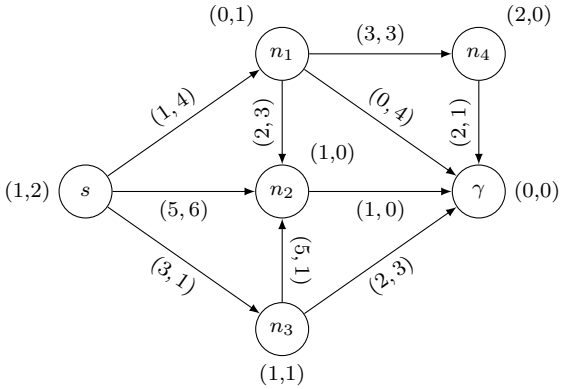


Fig. 1 An instance of $G = (N, A)$ with $N = \{s, n_1, n_2, n_3, \gamma\}$; cost vectors are given next to arcs and heuristic cost vectors are given next to nodes.

Initially, we have $\mathcal{O} = \{\ell_s\}$ where $\ell_s = [s, \langle s \rangle, (0, 0)]$ (see line 5). At the first iteration step of the while loop, label ℓ_s is moved from $\mathcal{O}$ to $\mathcal{C}$ to be expanded (see lines 8-9). Since $s \notin \Gamma$ and $\Pi(s) = \{n_1, n_2, n_3\}$, the following labels are generated: $\ell_1 = [n_1, \langle s, n_1 \rangle, (1, 4)]$, $\ell_2 = [n_2, \langle s, n_2 \rangle, (5, 6)]$ et $\ell_3 = [n_3, \langle s, n_3 \rangle, (3, 1)]$ (see lines 13-14). Since we have $\mathcal{O}(n_i) \cup \mathcal{C}(n_i) = \emptyset$ for all $i \in \{1, 2, 3\}$ (see lines 15-16), these generated labels are then inserted in $\mathcal{O}$. Moreover, these labels are not discarded by rule R2 since $g_{\mathcal{S}}$ is empty, and they are not eliminated by rule R1 since we have $\mathcal{O}(n_i) \cup \mathcal{C}(n_i) = \{\ell_i\}$ for all $i \in \{1, 2, 3\}$.

At the second iteration step, we have $\mathcal{O} = \{\ell_1, \ell_2, \ell_3\}$ where $F(\ell_1) = \{(1, 5)\}$, $F(\ell_2) = \{(6, 6)\}$ and $F(\ell_3) = \{(4, 2)\}$. Since $(6, 6)$ is Pareto-dominated by $(1, 5)$, we can only expand ℓ_1 or ℓ_3 in line 8. Assume that label ℓ_3 is selected at this step. In that case, label ℓ_3 is moved from $\mathcal{O}$ to $\mathcal{C}$ (line 9). Since $n_3 \notin \Gamma$ and $\Pi(n_3) = \{n_2, \gamma\}$, two labels are generated: $\ell'_2 = [n_2, \langle s, n_3, n_2 \rangle, (8, 2)]$ and $\ell_\gamma = [\gamma, \langle s, n_3, \gamma \rangle, (5, 4)]$. Then, the algorithm proceeds as follows:

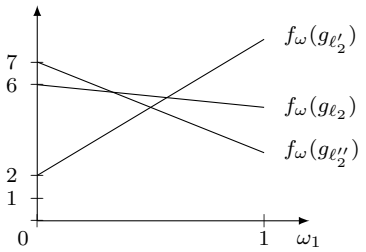
- *Label ℓ'_2: since $\mathcal{O}(n_2) = \{\ell_2\}$, $\mathcal{C}(n_2) = \emptyset$ and not$(g_{\ell_2} \precsim_P g_{\ell'_2})$, label ℓ'_2 is inserted in $\mathcal{O}$ in lines 15-16. Then label ℓ'_2 is not discarded by rule R2 since $g_{\mathcal{S}} = \emptyset$ (see line 17). Finally, none of the labels in $\mathcal{O}(n_2) = \{\ell_2, \ell'_2\}$ is eliminated by rule R1 since $\{g_{\ell_2}, g_{\ell'_2}\} \prec^{\delta}_{\Omega} \{g_{\ell_2}\}$ and $\{g_{\ell_2}, g_{\ell'_2}\} \prec^{\delta}_{\Omega} \{g_{\ell'_2}\}$ do not hold: for instance, we have $f_\omega(g_{\ell_2}) - f_\omega(g_{\ell'_2}) \leq \delta$ with $\omega = (3/4, 1/4) \in \Omega$ and $f_\omega(g_{\ell'_2}) - f_\omega(g_{\ell_2}) \leq \delta$ with $\omega = (1/2, 1/2) \in \Omega$.*
- *Label ℓ_γ: since $\mathcal{O}(\gamma) \cup \mathcal{S} = \emptyset$, label ℓ_γ is inserted in $\mathcal{O}$ and is discarded by neither rule R1 nor rule R2.*

Finally, we have $\mathcal{O} = \{\ell_1, \ell_2, \ell'_2, \ell_\gamma\}$ at the end of the second step with $F(\ell_1) = \{(1, 5)\}$, $F(\ell_2) = \{(6, 6)\}$, $F(\ell'_2) = \{(9, 2)\}$ and $F(\ell_\gamma) = \{(5, 4)\}$. Since $(6, 6)$ is Pareto-dominated by $(1, 5)$, only ℓ_1, ℓ'_2 and ℓ_γ can be expanded at the third step (see line 8).

Assume that ℓ_γ is selected at this step. In that case, label ℓ_γ is moved from $\mathcal{O}$ to $\mathcal{C}$ (see line 9) and is inserted in $\mathcal{S}$ (see lines 10-11).

At the fourth iteration step, we have $\mathcal{O} = \{\ell_1, \ell_2, \ell'_2\}$ where $F(\ell_1) = \{(1,5)\}$, $F(\ell_2) = \{(6,6)\}$ and $F(\ell'_2) = \{(9,2)\}$. At this step, only label ℓ_1 and label ℓ'_2 can be expanded since $(6,6)$ is Pareto-dominated. Assume that ℓ_1 is selected for expansion (in line 8). In that case, label ℓ_1 is moved from $\mathcal{O}$ to $\mathcal{C}$ in line 9. Moreover, since $n_1 \notin \Gamma$ and $\Pi(n_1) = \{n_2, \gamma, n_4\}$, the following labels are generated: $\ell''_2 = [n_2, \langle s, n_1, n_2\rangle, (3,7)]$, $\ell'_\gamma = [\gamma, \langle s, n_1, \gamma\rangle, (1,8)]$ and $\ell_4 = [n_4, \langle s, n_1, n_4\rangle, (4,7)]$ (see lines 13-14). Then:

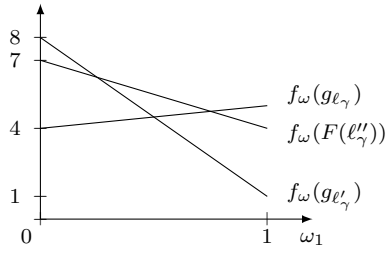
- *Label ℓ''_2: since $\mathcal{O}(n_2) = \{\ell_2, \ell'_2\}$, $\mathcal{C}(n_2) = \emptyset$, not$(g_{\ell_2} \precsim_P g_{\ell''_2})$ and not$(g_{\ell'_2} \precsim_P g_{\ell''_2})$, label ℓ''_2 is inserted in the set $\mathcal{O}$ in lines 15-16. Then, label ℓ''_2 is not discarded by rule R2 since we have $\mathcal{S} = \{\ell_\gamma\}$ and $\{g_{\ell_\gamma}\} \prec^\delta_\Omega F(\ell''_2)$ does not hold: for instance, we have $f_\omega(F(\ell''_2)) - f_\omega(g_{\ell_\gamma}) \leq \delta$ with $\omega = (4/5, 1/5) \in \Omega$. However, ℓ_2 is deleted by rule R1 in line 19 since we have $\{g_{\ell_2}, g_{\ell'_2}, g_{\ell''_2}\} \prec^\delta_\Omega \{g_{\ell_2}\}$, as illustrated below:*



- *Label ℓ'_γ: since $\mathcal{O}(\gamma) = \emptyset$, $\mathcal{S} = \{\ell_\gamma\}$ and not$(g_{\ell_\gamma} \precsim_P g_{\ell'_\gamma})$, ℓ'_γ is inserted in $\mathcal{O}$ (see lines 15-16). Then, it is not discarded by rules R1 and R2 since $\{g_{\ell_\gamma}\} \prec^\delta_\Omega \{g_{\ell'_\gamma}\}$ does not hold: for instance, we have $f_\omega(g_{\ell'_\gamma}) - f_\omega(g_{\ell_\gamma}) \leq \delta$ with $\omega = (3/4, 1/4) \in \Omega$.*
- *Label ℓ_4: since $\mathcal{O}(n_4) \cup \mathcal{C}(n_4) = \emptyset$, label ℓ_4 is inserted in $\mathcal{O}$ in line 16. However, we know that $\{g_{\ell_\gamma}\} \prec^\delta_\Omega F(\ell_4)$ holds since $F(\ell_4)$ is Pareto-dominated by g_{ℓ_γ} and so label ℓ_4 is necessarily eliminated by rule R2 in line 17.*

Finally, we have $\mathcal{O} = \{\ell'_2, \ell''_2, \ell'_\gamma\}$ at the end of the fourth step, where $F(\ell'_2) = \{(9,2)\}$, $F(\ell''_2) = \{(4,7)\}$ and $F(\ell'_\gamma) = \{(1,8)\}$. Since none of these vectors is Pareto-dominated, any of them can be expanded at the fifth step. Assume that ℓ'_γ is chosen. In that case, it is moved from $\mathcal{O}$ to $\mathcal{C}$ (in line 9) and it is inserted in $\mathcal{S}$ (in line 11).

At the beginning of the sixth step, we have $\mathcal{O} = \{\ell'_2, \ell''_2\}$ and either ℓ'_2 or ℓ''_2 can be selected for expansion. If label ℓ''_2 is expanded, then it is moved from $\mathcal{O}$ to $\mathcal{C}$ in line 9. Then, since $n_2 \notin \Gamma$ and $\Pi(n_2) = \{\gamma\}$, label $\ell''_\gamma = [\gamma, \langle s, n_1, n_2, \gamma\rangle, (4,7)]$ is generated (see lines 13-14). Since $\mathcal{O}(\gamma) = \emptyset$, $\mathcal{S} = \{\ell_\gamma, \ell'_\gamma\}$, not$(g_{\ell_\gamma} \precsim_P g_{\ell''_\gamma})$ and not$(g_{\ell'_\gamma} \precsim_P g_{\ell''_\gamma})$, label ℓ''_γ is inserted in $\mathcal{O}$ (see lines 15-16). However, rule R2 eliminates ℓ''_γ in line 17 since $\{g_{\ell_\gamma}, g_{\ell'_\gamma}, g_{\ell''_\gamma}\} \prec^\delta_\Omega F(\ell''_\gamma)$ holds as illustrated below:



At the beginning of the seventh step, we have $\mathcal{O} = \{\ell'_2\}$ *and so* ℓ'_2 *is moved from* $\mathcal{O}$ *to* $\mathcal{C}$ *in line 9. Then, since* $n_2 \notin \Gamma$ *and* $\Pi(n_2) = \{\gamma\}$, *label* $\ell'''_\gamma = [\gamma, \langle s, n_3, n_2, \gamma\rangle, (9, 2)]$ *is generated. Since* $\mathcal{O}(\gamma) = \emptyset$, $\mathcal{S} = \{\ell_\gamma, \ell'_\gamma\}$, $not(g_{\ell_\gamma} \precsim_P g_{\ell'''_\gamma})$ *and* $not(g_{\ell'_\gamma} \precsim_P g_{\ell'''_\gamma})$, *label* ℓ'''_γ *is inserted in* $\mathcal{O}$ *(see line 17). Then, we know that this label is not eliminated by rules R1 and R2 since* $\{g_{\ell_\gamma}, g_{\ell'_\gamma}, g_{\ell'''_\gamma}\} \prec^\delta_\Omega \{g_{\ell'''_\gamma}\}$ *does not hold: for instance, we have* $f_\omega(g_{\ell'''_\gamma}) - f_\omega(g_{\ell_\gamma}) \leq \delta$ *and* $f_\omega(g_{\ell'''_\gamma}) - f_\omega(g_{\ell'_\gamma}) \leq \delta$ *for* $\omega = (1/5, 4/5) \in \Omega$. *Thus, we have* $\mathcal{O} = \{\ell'''_\gamma\}$ *at the end of this step.*

At the eighth step, label ℓ'''_γ *is expanded: it is moved from* $\mathcal{O}$ *to* $\mathcal{C}$ *in line 9 and then it is inserted in* $\mathcal{S}$ *in line 11. Since we have* $\mathcal{O} = \emptyset$ *at the end of this iteration step, the while loop ends here. Finally the algorithm stops after returning set* $\mathcal{S}$ *which includes only three solution paths out of the six feasible solutions of this problem, namely* $\langle s, n_3, \gamma\rangle$, $\langle s, n_1, \gamma\rangle$ *and* $\langle s, n_3, n_2, \gamma\rangle$.

3.3 Combining Incremental Elicitation and Search

Algorithm 1 can be used to focus the search on relevant Pareto-optimal solution paths by detecting and discarding paths that cannot be part of a solution path with a max regret below δ. However, for a given set Ω, it may be the case that no solution path $p \in P(s, \Gamma)$ is such that $\mathrm{MR}(g(p), \mathcal{X}, \Omega) \leq \delta$. We introduce now a sufficient condition on $\mathrm{mMR}(g_\mathcal{S}, \Omega)$ to guarantee the existence of a solution path with a max regret below δ:

Proposition 5. *If* $\mathrm{mMR}(g_\mathcal{S}, \Omega) \leq \delta$ *at the end of the search performed by Algorithm 1, then* $\mathrm{MR}(g(p^*), \mathcal{X}, \Omega) \leq \delta$ *for any solution path* $p^* \in \arg\min_{p_\ell : \ell \in \mathcal{S}} \mathrm{MR}(g_\ell, g_\mathcal{S}, \Omega)$.

Proof. Let p^* be an element of $\arg\min_{p_\ell:\ell\in\mathcal{S}} \mathrm{MR}(g_\ell, g_\mathcal{S}, \Omega)$. We want to prove that we have $\mathrm{MR}(g(p^*), \mathcal{X}, \Omega) \leq \delta$. By definition of max regrets, this amounts to prove that $\mathrm{PMR}(g(p^*), g(p^0), \Omega) \leq \delta$ holds for any solution path p^0 such that $g(p^0) \in \mathcal{X}$. Hence, we want to prove that we have $f_\omega(g(p^*)) - f_\omega(g(p^0)) \leq \delta$ for every $\omega \in \Omega$. With this aim in view, we now divide the solution paths into three categories:

Category 1: there exists a label $\ell_0 \in \mathcal{S}$ such that $p^0 = p_{\ell_0}$. In that case, we can directly infer the result since we have $f_\omega(g(p^*)) - f_\omega(g(p^0)) \leq \mathrm{PMR}(g(p^*), g(p^0), \Omega) \leq \mathrm{MR}(g(p^*), g_\mathcal{S}, \Omega) = \mathrm{mMR}(g_\mathcal{S}, \Omega) \leq \delta$.

Category 2: there exists a label ℓ_0 such that path p_{ℓ_0} is a subpath of p^0 and label ℓ_0 was eliminated during the search due to rule R2. In that case, there exists a set of labels $\mathcal{S}' \subseteq \mathcal{S}$ such that $g_{\mathcal{S}'} \prec^\delta_\Omega F(\ell_0)$. Therefore, for all $h \in H(n_{\ell_0})$ and all $\omega \in \Omega$, there exists a path $p^h_\omega \in \{p_\ell : \ell \in \mathcal{S}'\}$ such that $f_\omega(g_{\ell_0} + h) - f_\omega(g(p^h_\omega)) > \delta$. Since H is admissible, there exists $h' \in H(n_{\ell'})$ such that $g_{\ell_0} + h' \precsim_P g(p^0)$. Then, since $f_\omega(x) \leq f_\omega(y)$ for any two vectors $x, y \in \mathbb{R}^q_+$ such that $x \precsim_P y$, we have $f_\omega(g_{\ell_0} + h') \leq f_\omega(g(p^0))$. As a consequence, we have $f_\omega(g(p^0)) - f_\omega(g(p^{h'}_\omega)) > \delta$, which can be rewritten $f_\omega(g(p^{h'}_\omega)) - f_\omega(g(p^0)) < -\delta$. Moreover, since $\mathcal{S}' \subseteq \mathcal{S}$ and $\mathrm{MR}(g(p^*), g_\mathcal{S}, \Omega) \leq \delta$, we necessarily have $f_\omega(g(p^*)) - f_\omega(g(p^{h'}_\omega)) \leq \delta$. By summing the two previous inequalities, we obtain $f_\omega(g(p^*)) - f_\omega(g(p^0)) < \delta - \delta = 0 \leq \delta$.

Category 3: there exists a label ℓ_0 such that path p_{ℓ_0} is a subpath of p^0 and label ℓ_0 was eliminated during the search by rule R1. In that case, there exists a set of labels L such that $\{p_\ell : \ell \in L\} \subseteq P(s, n_{\ell_0})$ and $\{g_\ell : \ell \in L\} \prec^\delta_\Omega \{g_{\ell_0}\}$.

Let $p \in P(n_{\ell_0}, \Gamma)$ be the unique path such that $p^0 = p_{\ell_0} \circ p$. Since we have $\{g_\ell : \ell \in L\} \prec^\delta_\Omega \{g_{\ell_0}\}$ and since relation $\prec^\delta_\Omega$ is additive (see Proposition 2), we have $\{g_\ell : \ell \in L\} + \{g(p)\} \prec^\delta_\Omega \{g_{\ell_0}\} + \{g(p)\}$, i.e. $\{g_\ell + g(p) : \ell \in L\} \prec^\delta_\Omega \{g_{\ell_0} + g(p)\}$. Hence, we have $\{g(p_\ell \circ p) : \ell \in L\} \prec^\delta_\Omega \{g(p^0)\}$. Therefore, there exists $\ell \in L$ such that $f_\omega(g(p^0)) - f_\omega(g(p^1)) > \delta$ (by definition of relation $\prec^\delta_\Omega$), where $p^1 = p_\ell \circ p$. Hence, we have $f_\omega(g(p^1)) - f_\omega(g(p^0)) < -\delta$. Note that we necessarily have $p^1 \in P(s, \Gamma)$ because $p_\ell \in P(s, n_{\ell_0})$ and $p \in P(n_{\ell_0}, \Gamma)$. Since path p^1 is an element of $P(s, \Gamma)$, then we can iterate this whole reasoning on path p^1. More precisely, the two following cases may occur:

- **Case 1:** path p^1 belongs to category 1 or 2. In that case, we just proved that we necessarily have $f_\omega(g(p^*)) - f_\omega(g(p^1)) \leq \delta$. Moreover, since $f_\omega(g(p^1)) - f_\omega(g(p^0)) < -\delta$ by definition of p^1, then we obtain $f_\omega(g(p^*)) - f_\omega(g(p^0)) < 0 \leq \delta$ by summing the two previous inequalities.
- **Case 1:** path p^1 belongs to category 3. In that case, we can prove that there exists a solution path p^2 such that $f_\omega(g(p^2)) - f_\omega(g(p^1)) < -\delta$ with the exact same reasoning. Since we have $f_\omega(g(p^1)) - f_\omega(g(p^0)) < -\delta$ by definition of p^1, then we obtain $f_\omega(g(p^2)) - f_\omega(g(p^0)) < -2\delta < -\delta$ by summing the two previous inequalities. Hence, we can now iterate the reasoning on p^2 in order to prove that we have $f_\omega(g(p^*)) - f_\omega(g(p^0)) \leq \delta$: if p^2 belongs to category 2 or 3, then we can infer the result with the same reasoning; otherwise, we have to consider a path p^3 and so on. Note that this iterative process necessarily stops after a finite number of steps since we have $f_\omega(g(p^{k+1})) - f_\omega(g(p^k)) < -\delta$ for all steps $k \geq 0$ and $f_\omega(g(p)) \geq 0$ for all solution paths $p \in P(s, \Gamma)$.

□

Thus, if the value $\text{mMR}(g_\mathcal{S}, \Omega)$ is smaller than threshold δ at the end of the procedure, then we know that any solution path $p^* \in \arg\min_{p_\ell : \ell \in \mathcal{S}} \text{MR}(g_\ell, g_\mathcal{S}, \Omega)$ satisfies $\text{MR}(g(p^*), \mathcal{X}, \Omega) \leq \delta$. Otherwise, one may consider a two stage procedure that consists first in applying Algorithm 1 to compute $\mathcal{S}$ and then in applying an efficient incremental elicitation method designed for explicit sets of solutions (e.g., the CSS method) so as to reduce Ω until the value $\text{mMR}(g_\mathcal{S}, \Omega)$ drops below δ. However, this approach would not be very efficient in practice due to the possibly large size of $\mathcal{S}$. To implement this idea more efficiently, we propose instead to refine Algorithm 1 by interleaving preference elicitation and search steps. Collecting preference information during the search is made possible by the following straightforward result:

Proposition 6. *For all Ω, Ω' such that $\Omega' \subseteq \Omega$ and for all sets $X, Y \subset \mathbb{R}^q_+$:*

$$X \prec^\delta_\Omega Y \Rightarrow X \prec^\delta_{\Omega'} Y$$

This proposition shows that new preference statements obtained during the search do not invalidate the pruning operations made so far while increasing our pruning opportunities for the rest of the search. We now present two strategies (S1 and S2) for generating preference queries during the search so as to obtain a set Ω sufficiently small to identify a near-optimal solution. These elicitation strategies directly integrate the CSS strategy to Algorithm 1 so as to generate informative preference queries during the search.

Strategy S1 aims to use new preference statements to better select the next label to expand. More precisely, in Algorithm 1 line 8, instead of selecting ℓ^* in L^* arbitrarily, we iteratively ask queries applying the CSS method over the set $X = \{g_\ell + h : \ell \in L^*, h \in H(n_\ell)\}$ of heuristic cost vectors until the value $\text{mMR}(X, \Omega)$ drops below δ. Note that if $\text{mMR}(X, \Omega)$ is initially smaller than δ at some step, then no query is asked at this step. Then, the label $\ell^* \in L^*$ selected for expansion corresponds to an element of X with a max regret below δ, i.e. the label ℓ^* is such that $\text{MR}(g_{\ell^*} + h, X, \Omega) \leq \delta$ for some $h \in H(n_{\ell^*})$. Thus, using this strategy amounts to only extending paths associated with a near-optimal heuristic cost vector. Finally, if $\text{mMR}(g_{\mathcal{S}}, \Omega)$ is strictly larger than δ at the end of Algorithm 1, then S1 applies the CSS method over $g_{\mathcal{S}}$ until the $\text{mMR}(g_{\mathcal{S}}, \Omega) \leq \delta$; this ensures that a near-optimal solution path is returned by Algorithm 1 (see Proposition 5).

Strategy S2 only uses new preference statements to keep the value $\text{mMR}(g_{\mathcal{S}}, \Omega)$ below δ at all times. More precisely, whenever the insertion of ℓ^* in $\mathcal{S}$ (line 11) makes the value $\text{mMR}(g_{\mathcal{S}}, \Omega)$ be strictly larger than δ, then S2 repeatedly asks queries applying the CSS over $g_{\mathcal{S}}$ until $\text{mMR}(g_{\mathcal{S}}, \Omega)$ drops below δ. Therefore, S2 ensures that a near-optimal solution path is returned by Algorithm 1 (see Proposition 5).

The following example gives an execution of Algorithm 1 combined with strategy S2 on a small instance.

Example 2. *Consider the instance of $G = (N, A)$ given in Figure 1 with $\delta = 0$ and assume that the DM's preferences are represented by a weighted sum with the hidden weight $\omega_0 = (0.6, 0.4)$. With no preference information, the set Ω of admissible weights $\omega = (\omega_1, \omega_2)$ is characterized by the constraint $0 < \omega_1 < 1$, the weight ω_2 being implicitly defined by the normalization constraint (i.e. $\omega_2 = 1 - \omega_1$).*

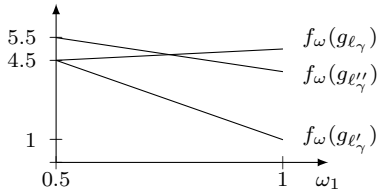
Recall that strategy S2 only generates preference queries when $\text{mMR}(g_{\mathcal{S}}, \Omega)$ becomes strictly larger than $\delta = 0$ after inserting a new label in $\mathcal{S}$. Therefore the first four iteration steps of the while loop are here the same as those of Example 1. Hence, at the beginning of the fifth iteration step, we have:

- $\mathcal{O} = \{\ell'_2, \ell''_2, \ell'_\gamma\}$ *where* $\ell'_2 = [n_2, \langle s, n_3, n_2\rangle, (8, 2)]$, $\ell''_2 = [n_2, \langle s, n_1, n_2\rangle, (3, 7)]$ *and* $\ell'_\gamma = [\gamma, \langle s, n_1, \gamma\rangle, (1, 8)]$.
- $\mathcal{C} = \{\ell_s, \ell_1, \ell_3, \ell_\gamma\}$, *where* $\ell_s = [s, \langle s\rangle, (0, 0)]$, $\ell_1 = [n_1, \langle s, n_1\rangle, (1, 4)]$, $\ell_3 = [n_3, \langle s, n_3\rangle, (3, 1)]$ *and* $\ell_\gamma = [\gamma, \langle s, n_3, \gamma\rangle, (5, 4)]$.
- $\mathcal{S} = \{\ell_\gamma\}$.

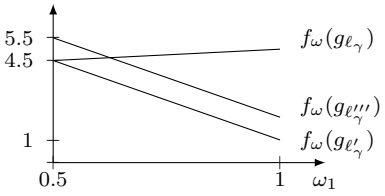
Moreover, we have $F(\ell'_2) = \{(9, 2)\}$, $F(\ell''_2) = \{(4, 7)\}$ and $F(\ell'_\gamma) = \{(1, 8)\}$. Since none of these vectors is Pareto-dominated, we can expand ℓ'_2, ℓ''_2 or ℓ'_γ at this iteration step. Assume that label ℓ'_γ is selected for expansion. In that case, this label is moved from $\mathcal{O}$ to $\mathcal{C}$ in line 9 and then it is inserted in $\mathcal{S}$ in line 11. Now, we have to compute $\text{mMR}(g_{\mathcal{S}}, \Omega)$ (following strategy S2). Since the latter value is stricly larger than δ, strategy S2 asks the DM to compare the cost vectors $(6, 4)$ and $(1, 8)$ which correspond to ℓ_γ and ℓ'_γ respectively. Since $f_{\omega_0}(5, 4) = 4.6 \geq f_{\omega_0}(1, 8) = 3.8$, the DM states that she prefers the second solution path to the first one. This induces the linear constraint $f_\omega(5, 4) = 5\omega_1 + 4(1 - \omega_1) \geq f_\omega(1, 8) = \omega_1 + 8(1 - \omega_1)$ on the space of admissible weighting vectors. This constraint can be rewritten as $\omega_1 \geq \frac{1}{2}$. Thus, we now have $\Omega = \{(\omega_1, 1 - \omega_1) : \frac{1}{2} \leq \omega_1 < 1\}$ and $\text{mMR}(g_{\mathcal{S}}, \Omega) = \text{MR}((1, 8), g_{\mathcal{S}}, \Omega) \leq 0 = \delta$.

Therefore, no more queries are generated at this step and label ℓ'_γ corresponds to the best solution found so far.

At the sixth iteration step, we can expand either ℓ'_2 or ℓ''_2. Assume that label ℓ''_2 is selected for expansion. In that case, label ℓ''_2 is moved from set $\mathcal{O}$ to set $\mathcal{C}$. Then, since $n_2 \notin \Gamma$ and $\Pi(n_2) = \{\gamma\}$, the following label is generated: $\ell''_\gamma = [\gamma, \langle s, n_1, n_2, \gamma\rangle, (4, 7)]$ (see lines 13-14). This label is then inserted in set $\mathcal{O}$ since we have not$(g_{\ell_\gamma} \precsim_P g_{\ell''_\gamma})$ and not$(g_{\ell'_\gamma} \precsim_P g_{\ell''_\gamma})$ (see lines 15-16). However, label ℓ''_γ is then deleted by our pruning rule R2 because we have $\{g_{\ell_\gamma}, g_{\ell'_\gamma}, g_{\ell''_\gamma}\} \prec^\delta_\Omega \{g_{\ell''_\gamma}\}$, as illustrated below:



Thus we have $\mathcal{O} = \{\ell'_2\}$ at the end of this iteration step. Therefore, at the seventh iteration step, label ℓ'_2 is necessarily selected for expansion (see line 8). It is first moved from set $\mathcal{O}$ to set $\mathcal{S}$ in line 9. Then, since $n_2 \notin \Gamma$ and $\Pi(n_2) = \{\gamma\}$, the following label is generated: $\ell'''_\gamma = [\gamma, \langle s, n_3, n_2, \gamma\rangle, (9, 2)]$. This label is then inserted in set $\mathcal{O}$ since we have not$(g_{\ell_\gamma} \precsim_P g_{\ell'''_\gamma})$ and not$(g_{\ell'_\gamma} \precsim_P g_{\ell'''_\gamma})$ (see lines 15-16). However, label ℓ'''_γ is eliminated by rule R2 since we have $\{g_{\ell_\gamma}, g_{\ell'_\gamma}, g_{\ell'''_\gamma}\} \prec^\delta_\Omega \{g_{\ell'''_\gamma}\}$ as illustrated below:



Hence set $\mathcal{O}$ is now empty and therefore the while loop must stop here. Finally, our interactive procedure returns the set $\mathcal{S}$ including label ℓ_γ which corresponds to a necessarily optimal solution path by construction, namely $\langle s, n_3, \gamma\rangle$. Thus, Algorithm 1 combined with strategy S2 is here able to determine a optimal solution path with only one generated preference query.

3.4 Numerical Tests

In this subsection, we report various numerical tests aiming to evaluate the performance of elicitation procedures S1 and S2 within Algorithm 1. For comparison, we also consider the two-stage procedure (named S0 hereafter) consisting in first running MOA* and then applying the CSS method over the resulting set of Pareto-optimal cost vectors. These algorithms are used here to determine a near-optimal solution path for the tolerance threshold $\delta = 0.01$ (meaning that the minimax regret will be below 0.01 at the end). These interactive algorithms are compared both in terms of computation times and number of preference queries. Starting from an empty set of preferences statements, simulated DMs answer queries according to

a linear scalarizing function f_ω, where the weighting vector ω is randomly chosen in the set $\{\omega \in \text{int}(\mathbb{R}^q_+) : \sum_{j\in\mathcal{Q}} \omega_j = 1\}$.

We consider instances of graph $G = (N, A)$ with one goal node γ and generated as follows: the nodes in N are uniformly drawn in the two dimension grid $\{1, \ldots, 1000\} \times \{1, \ldots, 1000\}$, except the source node s and the goal node γ which are respectively located in $(1, 500)$ and $(1000, 500)$. Then, each generated node is linked to its five nearest nodes (w.r.t. Euclidean distance) and the associated cost vectors are uniformly drawn in $\{0, \ldots, 100\}^q$. As heuristic H, we consider here that $H(n)$ only contains the ideal point $I(n) = (I_1, \ldots, I_q)$ for all nodes $n \in N$, where $I_j = \min_{x\in\{g(p):p\in P(n,\gamma)\}} x_j$ for all $j \in \mathcal{Q}$.

The tests aim to estimate 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 method S0 and and Algorithm 1 implemented with S1 or S2. In Tables 1 and 2, the results are obtained by averaging over 30 runs[2]. Linear optimizations (i.e. $\prec^{\delta}_{\Omega}$-dominance tests) are here performed using the Gurobi library of Java.

	$q=2$		$q=3$		$q=4$		$q=5$	
	time	queries	time	queries	time	queries	time	queries
S0	0.0	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 1 Impact of q the number of criteria ($|N| = 200$, times in minutes).

	$\|N\|=100$		$\|N\|=300$		$\|N\|=400$		$\|N\|=500$	
	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	0.3	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

Table 2 Impact of $|N|$ the number of nodes ($q = 3$, times in minutes).

Let us first compare method S0 and Algorithm 1 used with strategy S1. We see that Algorithm 1 used with strategy S1 is faster than S0 when we consider more than three criteria (see Table 1). In particular, strategy S1 is almost 100 times faster than S0 for $q = 4, 5$. Moreover, the number of generated 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 1 combined with S1 is much faster than S0 on bigger instances (see Table 2). More precisely, strategy S1 is almost five times faster than S0 while generating 8% less queries for $|N| = 500$. Looking now at the results obtained for S2, we first see that Algorithm 1 used with S2 generates less preference queries than S0. For instance, S2 generates 44% less queries for $q = 5$ (see Table 1) and saves

[2] The numerical tests were performed on an Intel Core i7-4770 CPU with 15GB de RAM.

about 25% of queries on the bigger instances (see Table 2). Therefore, strategy 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 1 combined with strategy S1 turns out to be faster but S2 may still be preferred to minimize the number of queries.

4 The Multicriteria Spanning Tree Problem

In this section, we study how incremental elicitation methods can be integrated into a greedy algorithm. We focus here on the multicriteria spanning tree problem, an easy optimization problem in the single objective case that can be solved in polynomial time using standard greedy algorithms (Kruskal, 1956; Prim, 1957). The multicriteria spanning tree problem appears naturally in various situations, for example, when a set of clients must be connected to one another through a communication or transportation network. Unfortunately, this problem becomes intractable when considering at least two criteria to optimize: the number of Pareto-optimal cost vectors attached to spanning trees is exponential in the number of nodes in the graph in the worst case (see e.g., Hamacher and Ruhe (1994)). This precludes one from performing an exhaustive enumeration in order to determine the best solution for the DM. Instead, assuming that the DM's preferences can be represented by a weighted sum, we propose to construct a (near-)optimal spanning tree using a greedy algorithm which alternates search steps and elicitation steps.

4.1 The General Framework

We consider here a multicriteria spanning tree problem defined by a connected graph $G = (N, E)$ where N is a finite set of nodes and each edge $e \in E$ is valued by a cost vector $x^e \in \mathbb{R}^q_+$ giving the cost of e with respect to different criteria (e.g., prices, construction time, distance). The set of criteria is denoted by $\mathcal{Q} = \{1, \ldots, q\}$, and every criterion is assumed to be additive over the edges, i.e. the cost vector x^F associated with any set of edges $F \subseteq E$ is defined by $x^F = \sum_{e \in F} x^e$. A spanning tree of graph G is a connected subgraph of G which includes every node of G while containing no cycle. Notice that a spanning tree T of G is completely characterized by its set of edges. Therefore, for the sake of simplicity, T will indifferently denote the tree or its set of edges in the sequel. Moreover, the set of cost vectors associated with the spanning trees of G will be denoted by $\mathcal{X}$. This set represents the image of all spanning trees in the space of criteria.

4.2 A Greedy Algorithm Incorporating an Incremental Elicitation Method

In this subsection, we consider the problem of determining a near-optimal spanning tree in the graph, i.e. a spanning tree with a max regret bounded above by a given tolerance threshold $\delta \geq 0$. Note that, for a given set Ω of admissible weighting vectors, it may be the case that no such spanning tree exists. In that case, we need to collect preference information to reduce the set of admissible weights Ω until being able to determine a spanning tree T such that $\mathrm{MR}(x^T, \mathcal{X}, \Omega) \leq \delta$. To

achieve this, one may consider a two stage procedure that consists first in computing the set $\mathcal{X}$ (or the Pareto set) and then in applying an incremental elicitation method designed for explicit sets of solutions (e.g., strategy CSS presented in Section 2.2). However, this approach would not be very efficient in practice due to the possibly large size of the Pareto set, as discussed above. Instead, we propose a multiobjective extension of Prim's algorithm (Prim, 1957) that generates preference queries during the search so as to guarantee that the returned spanning tree is near-optimal.

Let us recall briefly Prim's algorithm, which is a greedy algorithm constructing a spanning tree of minimum cost (in the single objective case). Starting from an initial node n_0, it first selects an adjacent edge of minimum cost. Then, it iteratively selects a min-cost edge in the cocycle $C(N')$, where N' is the set of nodes covered so far by the selected edges. Recall that the cocycle of a set N' is the set of all edges adjacent to N', i.e. $C(N') = \{(n_1, n_2) \in E : n_1 \in N', n_2 \in N \backslash N'\}$. Similarly, we propose here a greedy search that selects, at any step, an edge in the current cocycle. More precisely, at step i (selection of i^{th} edge), we collect preference information to discriminate between the edges of the current cocycle until determining an edge with a max regret below δ_i, where δ_i is a fraction of the admissibility threshold δ such that $\sum_{i=1}^{|N|-1} \delta_i = \delta$. In order to limit the number of generated preference queries, we use here the CSS method (presented in Section 2.2) to efficiently reduce the set Ω of admissible weights. Thus, we propose Algorithm 2:

Algorithm 2: Interactive search for multicriteria spanning tree problems

Input: $G = (N, E)$; n_0; $\delta_i, i \in \{1, \ldots, |N| - 1\}$: positive values summing to δ

1 $N' \leftarrow \{n_0\}$
2 $F \leftarrow \emptyset$
3 **for** $i = 1 \ldots |N| - 1$ **do**
4 ND $\leftarrow \{e \in C(N') : \forall e' \in C(N'), \text{not}(x^{e'} \prec_P x^e)\}$
5 $X \leftarrow \{x^e : e \in \text{ND}\}$
6 **while** $\text{mMR}(X, \Omega) > \delta_i$ **do**
7 Ask one preference query to the DM according to the CSS
8 Update Ω by inserting the linear constraint associated with the answer
9 **end**
10 Select an edge $e \in \text{ND}$ such that $x^e \in \arg\min_{x \in X} \text{MR}(x, X, \Omega)$
11 $F \leftarrow F \cup \{e\}$
12 $N' \leftarrow N' \cup \{n_e\}$, where n_e is the endpoint of e that is not in N'
13 **end**
14 **return** spanning tree F

Notice that, at each iteration step i, the number of iterations of the while loop (line 6) is bounded above by $|C(N')| \leq |N|^2$, which garantees the termination of the algorithm after a polynomial number of steps. The following theorem shows that this algorithm returns a near-optimal spanning tree at the end of the execution:

Theorem 1 *The spanning tree T returned by Algorithm 2 verifies* $\mathrm{MR}(x^T, \mathcal{X}, \Omega_f) \leq \delta$, *where Ω_f denotes the final set of admissible weights.*

Proof. We want to prove that we have $\mathrm{PMR}(x^T, x^{T_0}, \Omega_f) \leq \delta$ for any spanning tree T_0 of graph G. For any step $i \in \{1, \ldots, |N|-1\}$, let e_i be the i^{th} edge inserted in T and let N_i (resp. Ω_i) be the set N' (resp. Ω) at the end of step i. Let j denote the first iteration step such that $e_j \notin T_0$. Since T_0 is a spanning tree, there exists a chain c in T_0 linking the two endpoints of e_j. Moreover, since e_j links one node in N_{j-1} to one node in $N \backslash N_{j-1}$, there exists an edge e'_j in chain c that links one node in N_{j-1} to one node in $N \backslash N_{j-1}$. Note that e'_j is necessarily in the cocycle $C(N_{j-1})$ by definition. Then, two cases may occur: either $e'_j \in \mathrm{ND}_j$ or $e'_j \notin \mathrm{ND}_j$ where ND_j is the set of non Pareto-dominated edges in $C(N_{j-1})$ (see line 4). Let us prove that we have $\mathrm{PMR}(x^{e_j}, x^{e'_j}, \Omega_f) \leq \delta_j$ in both cases:

Case 1: $e'_j \in \mathrm{ND}_j$. In that case, since edge e_j has been selected to build the spanning tree T, we know that $x^{e_j} \in \arg\min_{x \in X_j} \mathrm{MR}(x, X_j, \Omega_j)$ (see lines 10-11) where $X_j = \{x^e : e \in \mathrm{ND}_j\}$. Moreover, since $\mathrm{mMR}(X_j, \Omega_j) \leq \delta_j$ at the end of the while loop (see line 6), we necessarily have $\mathrm{PMR}(x^{e_j}, x^{e'_j}, \Omega_j) \leq \delta_j$. Then, since $\Omega_j \subseteq \Omega_f$ by construction (see line 8), we have $\mathrm{PMR}(x^{e_j}, x^{e'_j}, \Omega_f) \leq \delta_j$.

Case 2: $e'_j \notin \mathrm{ND}_j$. By definition of ND_j, there exists $e \in \mathrm{ND}_j$ such that $x^e \prec_P x^{e'_j}$. For all $\omega \in \Omega_f$, we have $f_\omega(x^e) < f_\omega(x^{e'_j})$ since $f_\omega(x) < f_\omega(y)$ holds for any two vectors $x, y \in \mathbb{R}^q_+$ such that $x \prec_P y$. Hence $\mathrm{PMR}(x^{e_j}, x^{e'_j}, \Omega_f) \leq \mathrm{PMR}(x^{e_j}, x^e, \Omega_f)$. Then, similarly to the first case, we can prove that $\mathrm{PMR}(x^{e_j}, x^e, \Omega_f) \leq \delta_j$ holds since $x^e \in \mathrm{ND}_j$. Hence, we have $\mathrm{PMR}(x^{e_j}, x^{e'_j}, \Omega_f) \leq \mathrm{PMR}(x^{e_j}, x^e, \Omega_f) \leq \delta_j$.

Let T_1 be the spanning tree obtained from T_0 by replacing e'_j by e_j. Let e_k be the first edge inserted in T that is not in T_1. Then, similarly to T_0, we can prove that we have $\mathrm{PMR}(x^{e_k}, x^{e'_k}, \Omega_f) \leq \delta_k$. By iterating, we obtain $\mathrm{PMR}(x^{e_l}, x^{e'_l}, \Omega_f) \leq \delta_l$ for all $l \in \{1, \ldots, |N|-1\}$ such that $e_l \notin T_0$. Moreover, for all $l \in \{1, \ldots, |N|-1\}$ such that $e_l \in T_0$, we have $\mathrm{PMR}(x^{e_l}, x^{e_l}, \Omega_f) = 0 \leq \delta_l$. Let $\pi : T \to T_0$ be the one-to-one correspondence such that $\pi(e_l) = e'_l$ if $e_l \notin T_0$ and $\pi(e_l) = e_l$ otherwise. By definition, $\mathrm{PMR}(x^{e_l}, x^{\pi(e_l)}, \Omega_f) \leq \delta_l$ holds for all $l \in L = \{1, \ldots, |N|-1\}$. Finally:

$$\begin{aligned}
\mathrm{PMR}(x^T, x^{T_0}, \Omega_f) &= \max_{\omega \in \Omega_f} \{f_\omega(x^T) - f_\omega(x^{T_0})\} \\
&= \max_{\omega \in \Omega_f} \{f_\omega(\sum_{l \in L} x^{e_l}) - f_\omega(\sum_{l \in L} x^{\pi(e_l)})\} \\
&= \max_{\omega \in \Omega_f} \sum_{l \in L} \{f_\omega(x^{e_l}) - f_\omega(x^{\pi(e_l)})\} \text{ by linearity of } f_\omega \\
&\leq \sum_{l \in L} \max_{\omega \in \Omega_f} \{f_\omega(x^{e_l}) - f_\omega(x^{\pi(e_l)})\} \\
&= \sum_{l \in L} \mathrm{PMR}(x^{e_l}, x^{\pi(e_l)}, \Omega_f) \\
&\leq \sum_{l \in L} \delta_l \\
&= \delta
\end{aligned}$$

□

The following example presents an execution of Algorithm 2 on a small graph:

Example 3. *Consider the instance of $G = (N, E)$ with four nodes, three criteria and $\delta = 0$, that is given in Figure 2.*

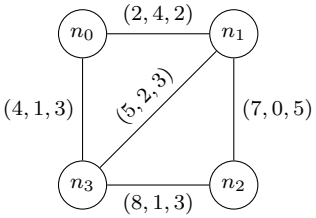


Fig. 2 An instance of graph $G = (N, E)$ with $N = \{n_0, n_1, n_2, n_3\}$ and $\mathcal{Q} = \{1, 2, 3\}$.

Assume that the DM's preferences are represented by a weighted sum with the hidden weights $\omega_0 = (0.4, 0.2, 0.4)$. With no preference information, the set Ω of admissible weights $\omega = (\omega_1, \omega_2, \omega_3)$ is defined by $\Omega = \{\omega \in int(\mathbb{R}^3_+) : \omega_1 + \omega_2 + \omega_3 = 1\}$. In Figure 3, the latter set is represented by the triangle ABC in the space (ω_1, ω_2), ω_3 being implicitly defined by $\omega_3 = 1 - \omega_1 - \omega_2$.

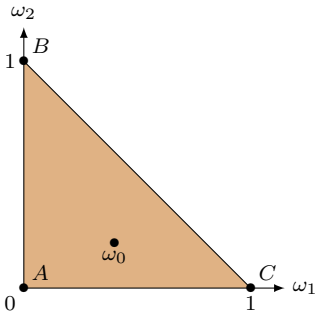


Fig. 3 The initial set Ω of admissible weights $\omega = (\omega_1, \omega_2, \omega_3)$, seen in the space (ω_1, ω_2).

We start with $N' = \{n_0\}$ and the set of selected edges F is initialized to the empty set. At the first iteration step, the cocycle $C(N')$ consists of the edges (n_0, n_1) and (n_0, n_3) with cost vectors $(2, 4, 2)$ and $(4, 1, 3)$ respectively. Since none of these vectors is Pareto-dominated, we have $ND = \{(n_0, n_1), (n_0, n_3)\}$ and $X = \{(2, 4, 2), (4, 1, 3)\}$. Moreover, we have $\mathrm{mMR}(X, \Omega) \approx 2 > \delta_1 = 0$. As a consequence, the procedure asks the DM to compare the later two cost vectors (following the CSS). Since $f_{\omega_0}(2, 4, 2) = 2.4 \leq f_{\omega_0}(4, 1, 3) = 3$, the DM states that she prefers $(2, 4, 2)$ to $(4, 1, 3)$. Then, the algorithm updates the set of admissible weights Ω by inserting the linear constraint $f_\omega(2, 4, 2) \leq f_\omega(4, 1, 3)$, i.e. $\omega_2 \leq \frac{1}{4}\omega_1 + \frac{1}{4}$. This constraint is represented by the (DE) line in Figure 4, the admissible area being below this line. Now we have $\mathrm{mMR}(X, \Omega) = \mathrm{MR}((2, 4, 2), \Omega) \leq \delta_1 = 0$. As a consequence, the while stops after asking only one query and the edge (n_0, n_1) is inserted in F.

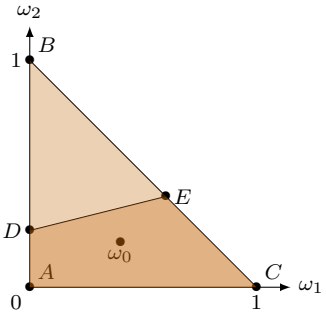


Fig. 4 The set Ω of admissible weights after the first iteration step, seen in the space (ω_1, ω_2).

At the second iteration step, the cocycle $C(N')$ *only includes the edges* (n_0, n_3), (n_1, n_3) *and* (n_1, n_2) *with cost vectors* $(4, 1, 3)$, $(5, 3, 4)$ *and* $(7, 0, 5)$ *respectively. Since* $(5, 3, 4)$ *is Pareto-dominated by* $(4, 1, 3)$*, we have* $ND = \{(n_0, n_3), (n_1, n_2)\}$ *and* $X = \{(4, 1, 3), (7, 0, 5)\}$*. Moreover,* $\text{mMR}(X, \Omega) = \text{MR}((4, 1, 3), \Omega) \approx -1.4 \leq \delta_1 = 0$*. Therefore, no query is needed at this step and edge* (n_0, n_3) *is directly inserted in* F*.*

At the third iteration step, the cocycle $C(N')$ *only includes* (n_1, n_2) *and* (n_2, n_3) *with cost vectors* $(7, 0, 5)$ *and* $(8, 1, 3)$ *respectively. Since none of these vectors is Pareto-dominated, we have* $ND = \{(n_1, n_2), (n_2, n_3)\}$ *and* $X = \{(7, 0, 5), (8, 1, 3)\}$*. Moreover, we have* $\text{mMR}(X, \Omega) \approx 1 > \delta_3 = 0$*. Therefore, the procedure asks the DM to compare these two cost vectors following the CSS. Since we have* $f_{\omega_0}(8, 1, 3) = 4.6 \leq f_{\omega_0}(7, 0, 5) = 4.8$*, the DM declares that she prefers* $(8, 1, 3)$ *to* $(7, 0, 5)$*. Then, the algorithm updates the set of admissible weights* Ω *by inserting the linear constraint* $f_\omega(8, 1, 3) \leq f_\omega(7, 0, 5)$*, i.e.* $\omega_2 \leq -\omega_1 + \frac{2}{3}$*. This constraint is represented by the line* (GH) *in Figure 5, the admissible are being below this line. Now we have* $\text{mMR}(X, \Omega) = \text{MR}((8, 1, 3), \Omega) = 0 \leq \delta_3$*. Therefore, the while loop stops after only one iteration and the edge* (n_2, n_3) *is inserted in set* F*.*

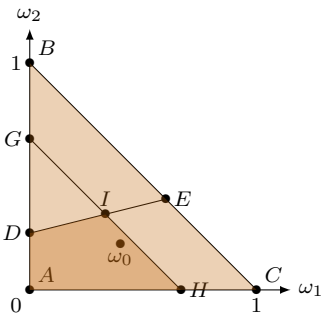


Fig. 5 The set Ω of admissible weights after the third iteration step, seen in the space (ω_1, ω_2).

Finally, the procedure stops after this step and returns the optimal spanning tree $\{(n_0, n_1), (n_0, n_3), (n_2, n_3)\}$*. Note that only two queries are here generated to discriminate between the eight feasible spanning trees.*

4.3 Numerical Tests

In this subsection, we report numerical tests of Algorithm 2 on random instances of the multicriteria spanning tree problem. The numerical tests were performed on an Intel Core i7-4770 CPU with 15GB de RAM. In these tests, we consider graphs with a number of nodes $|N|$ ranging from 25 to 100, a number of criteria q ranging from 2 to 6 and an edge density of 50% (the costs are drawn within $\{1, 1000\}$). In our experiments, simulated DMs answer queries according to a linear scalarizing function f_ω where ω is randomly chosen in $\Omega = \{\text{int}(\mathbb{R}^q_+) : \sum_{j\in\mathcal{Q}} \omega_j = 1\}$. To study the impact of threshold δ on the number of queries and the computation time, we run tests with two distinct values: $\delta = 0.05$ and $\delta = 0.1$. The values δ_j required at every step of the algorithm are set to $\delta_j = \delta/(|N|-1)$ for all $j \in \{1, \ldots, |N|-1\}$. Here also we use the Gurobi library of Java to compute pairwise max regrets at every step. Results have been obtained by averaging over 30 runs and are summarized in Table 3.

		$\delta = 0.05$		$\delta = 0.1$	
q	$\|N\|$	time	queries	time	queries
2	25	0.02	3.8	0.02	3.4
2	50	0.03	3.6	0.05	2.8
2	75	0.09	4.4	0.12	3.4
2	100	0.14	4.2	0.18	2.8
4	25	0.08	16.2	0.12	11.8
4	50	0.39	17.0	0.71	12.6
4	75	1.26	19.8	1.94	12.2
4	100	2.12	20.4	3.80	12.1
6	25	0.22	29.2	0.23	15.4
6	50	1.35	32.0	2.05	22.8
6	75	4.28	30.8	8.30	25.0
6	100	12.50	33.0	19.70	25.4

Table 3 Performance of Algorithm 2 (times in minutes, number of preference queries).

In Table 3, we observe that our interactive greedy procedure is quite efficient considering the highly combinatorial nature of the spanning tree problem and the number of criteria under consideration; recall that the state of the art literature on the computation of Pareto-optimal spanning trees considers no more than two or three criteria. For example, we need only twelve preference queries on average to determine a near-optimal spanning tree on instances with 100 nodes and 4 criteria (with $\delta = 0.1$). The relative efficiency of our search procedure is due to the possibility of collecting some preference information during the search. Moreover, as the value of δ increases from 0.05 to 0.1, the requirement on the performance guarantee is weakened; this explains the observed reduction of the number of preference queries generated during the search. Note also that reducing the number of queries tends to keep a larger set Ω of admissible parameters during the execution which impacts negatively on computation times, as can be observed in Table 3.

5 The Multicriteria Assignment Problem

In this section, we show how a combinatorial optimization problem with incomplete preference information can be efficiently solved using an interactive branch and bound procedure. For illustration purposes, we focus here on a multicriteria assignment problem where each item (e.g., objects, resources, tasks) must be assigned to a different agent so that the total cost of the assignment is minimized. In the single objective case, it is well known that this optimization problem can be solved in polynomial time (using, for example, the so-called Hungarian algorithm (Kuhn, 1955)). However, this problem becomes computationally difficult when considering several criteria due to the possibly large size of the Pareto set (Ehrgott, 2005), which precludes to perform any exhaustive enumeration. Instead, assuming that the DM's preferences can be represented by a linear aggregation function, we propose here to determine a (near-)optimal assignment by using a branch and bound procedure that generates preference queries at some key steps of the search to efficiently reduce the size of the search tree while limiting the number of queries.

5.1 The General Framework

Given a set of agents $N = \{1, \dots, n\}$ and a set of items T of exactly the same size, we can model the assignment problem by a bipartite graph $G = (N \cup T, E)$ where $E = \{(i,t) : i \in N, t \in T\}$ is a finite set of edges representing the possible assignments. Each edge $e \in E$ is valued by a cost vector $x^e = (x^e_1, \dots, x^e_q) \in \mathbb{R}^q_+$ giving the cost of e with respect to q different criteria (e.g., completion time, costs, resources). In the sequel, the set of criteria will be denoted by $\mathcal{Q} = \{1, \dots, q\}$. The cost vector x^F of any set of edges $F \subseteq E$ is defined by $x^F = \sum_{e \in F} x^e$. An assignment is defined by a set of edges $F \subseteq E$ such that $|\{(i,t) \in F : t \in T\}| = 1$ for all agents $i \in N$ and $|\{(i,t) \in F : i \in N\}| = 1$ for all items $t \in T$. The set of all possible assignments is denoted by $\mathcal{F}$ and the associated set of feasible cost vectors is denoted by $\mathcal{X} = \{x^F : F \in \mathcal{F}\}$ in the sequel; the latter set represents the image of all assignments in the space of criteria.

In this section, we study the problem of identifying a near-optimal assignment, i.e. an assignment F with a max regret $\mathrm{MR}(x^F, \mathcal{X}, \Omega) = \max_{x \in \mathcal{X}} \mathrm{PMR}(x^F, x, \Omega) = \max_{x \in \mathcal{X}} \max_{\omega \in \Omega} \{f_\omega(x^F) - f_\omega(x)\}$ that is below a given tolerance threshold $\delta \geq 0$. First, we propose a branch and bound algorithm that is able, given a set Ω of admissible weights, to detect that some partial assignments cannot be part of a near-optimal assignment. Then, we propose a query generation strategy that collects preference information during the search so as to ensure that the branch and bound algorithm returns a near-optimal assignment at the end of its execution.

5.2 Search for Possibly Near-Optimal Assignments

We now introduce a branch and bound procedure that enables to identify partial assignments that cannot be part of an assignment with a max regret below tolerance threshold δ (given our partial knowledge of the DM' preferences). Let d_e be the decision variable associated with edge $e \in E$, i.e. we have $d_e = 1$ if and only if

edge e is in the assignment. Nodes of the search tree represent partial instances of the decision variables d_e, $e \in E$. More precisely, each node η of the search tree is characterized by a pair of sets (E^0_η, E^1_η) which are defined by $E^k_\eta = \{e \in E : d_e = k\}$ for $k = 0, 1$. Let $E_\eta = E \setminus (E^0_\eta \cup E^1_\eta)$ be the set of edges in the graph that are associated with an undecided variable at node η. By construction, each node η is associated with a different region of the solution space which is defined as follows: assignment F is attached to node η if and only if we have $E^0_\eta \cap F = \emptyset$ and $E^1_\eta \subseteq F$. The set of cost vectors associated with the assignments attached to node η will be denoted by $\mathcal{S}_\eta$ hereafter. The main features of our branch and bound procedure are the following:

Initialization. Using a heuristic, a branch and bound procedure usually determines some set of solutions before performing the search in order to define an initial bound on the optimal cost. We propose here to construct an initial set $\mathcal{S}_0$ of cost vectors as follows. For every criterion $j \in \mathcal{Q}$, we first determine an assignment that is optimal when only considering criterion j and then we insert the associated cost vector in the set $\mathcal{S}_0$. Recall that each of these assignments can be computed in polynomial time since determining an optimal assignment is a polynomial time problem in the single objective case.

Evaluation and pruning. Let $\mathcal{S}$ be the set of cost vectors attached to the assignments found so far (initially $\mathcal{S} = \mathcal{S}_0$) and let $\mathcal{O}$ be the current set of nodes to be explored. Our pruning rule is based on the notion of setwise max regret defined as follows: the setwise max regret $\mathrm{SR}(X, Y, \Omega)$ of a set $X \subseteq \mathcal{X}$ with respect to a set $Y \subseteq \mathcal{X}$ is the worst-case loss when recommending the best cost vector in set X instead of the best cost vector in set Y. More formally, the setwise max regret is definded as follows:

$$\mathrm{SR}(X, Y, \Omega) = \max_{\omega \in \Omega} \left\{ \min_{x \in X} f_\omega(x) - \min_{y \in Y} f_\omega(y) \right\}$$

If we have $\mathrm{SR}(X, Y, \Omega) < -\delta$, then we know that set Y does not contain any near-optimal cost vector. It indeed induces that, for all solutions $y \in Y$ and for all weights $\omega \in \Omega$, there exists an assignment $x \in X$ such that $f_\omega(x) - f_\omega(y) < -\delta$ holds, i.e. $f_\omega(y) - f_\omega(x) > \delta$; therefore, we have $\mathrm{MR}(y, \mathcal{X}, \Omega') > \delta$ for all solutions $y \in Y$ and for all set of weights $\Omega' \subseteq \Omega$. Thus, we propose to prune a node $\eta \in \mathcal{O}$ if the setwise max regret $\mathrm{SR}(\mathcal{S}, \mathcal{S}_\eta, \Omega)$ is strictly below $-\delta$ (since this implies that $\mathcal{S}_\eta$ includes no near-optimal assignment). Note that the setwise max regret $\mathrm{SR}(X, Y, \Omega)$ of a set $X \subseteq \mathcal{X}$ with respect to a set $Y \subseteq \mathcal{X}$ can be rewritten as follows:

$$\mathrm{SR}(X, Y, \Omega) = \max_{y \in Y} \max_{\omega \in \Omega} \min_{x \in X} \left\{ f_\omega(x) - f_\omega(y) \right\}$$

Therefore, we have $\mathrm{SR}(\mathcal{S}, \mathcal{S}_\eta, \Omega) = \max_{y \in \mathcal{S}_\eta} \max_{\omega \in \Omega} \min_{x \in \mathcal{S}} \{f_\omega(x) - f_\omega(y)\}$ for all nodes $\eta \in \mathcal{O}$. This alternative formulation of setwise max regrets allows us to compute $\mathrm{SR}(\mathcal{S}, \mathcal{S}_\eta, \Omega)$ as the optimal value of the following mixed-integer quadratic program (denoted by MIQP_η hereafter):

$$\max\ r$$

$$\text{s.t.}\begin{cases} r \leq \sum_{j\in\mathcal{Q}} \omega_j x_j - \sum_{j\in\mathcal{Q}} \omega_j \sum_{e\in E} d_e x_j^e,\ \forall x \in \mathcal{S} & \text{(4a)}\\ d_e = k,\ \forall k \in \{0,1\}, \forall e \in E_\eta^k & \text{(4b)}\\ \sum_{(i,t)\in E} d_{(i,t)} = 1,\ \forall i \in N & \text{(4c)}\\ \sum_{(i,t)\in E} d_{(i,t)} = 1,\ \forall t \in T & \text{(4d)}\\ \sum_{j\in\mathcal{Q}} \omega_j = 1 & \text{(4e)}\\ \sum_{j\in\mathcal{Q}} \omega_j x_j \leq \sum_{j\in\mathcal{Q}} \omega_j y_j,\ \forall (x,y) \in \mathcal{P} & \text{(4f)}\\ r \in \mathbb{R};\ d_e \in \{0,1\},\ \forall e \in E;\ \omega_j \geq 0,\ \forall j \in \mathcal{Q} \end{cases}$$

In this program, Equations (4b-4d) ensure that variables $d_e, e \in E$, characterize a feasible assignment attached to node η, Equations (4e-4f) guarantee that ω is normalized and compatible with the available preference information and Equation (4a) introduces variable $r \in \mathbb{R}$ representing the smallest difference of costs between a cost vector $x \in \mathcal{S}$ and the cost vector associated with the assignment characterized by variables $d_e, e \in E$. Note that the constraints given in Equation (4a) include quadratic terms of type $\omega_j d_e$, $e \in E$, since the weights $\omega_j, j \in \mathcal{Q}$, are also variables of the optimization problem. In order to linearize theses constraints, we introduce positive variables $v_{je}, j \in \mathcal{Q}, e \in E$, representing the product $\omega_j d_e$ and Equation (4a) is replaced by the following constraints:

$$\text{s.t.}\begin{cases} r \leq \sum_{j\in\mathcal{Q}} \omega_j x_j - \sum_{j\in\mathcal{Q}} \sum_{e\in E} v_{je} x_j^e,\ \forall x \in \mathcal{S}\\ v_{je} \leq \omega_j,\ \forall e \in E, \forall j \in \mathcal{Q}\\ v_{je} \leq d_e,\ \forall e \in E, \forall j \in \mathcal{Q}\\ v_{je} - \omega_j \geq d_e - 1,\ \forall e \in E, \forall j \in \mathcal{Q} \end{cases}$$

The resulting mixed-integer linear program (denoted by MIP_η hereafter) can be further simplified by removing all the decision variables d_e with $e \in E_\eta^0 \cup E_\eta^1$ since we know their respective value (see Equation (4b)).

Branching. Setwise max regrets $\text{SR}(\mathcal{S}, \mathcal{S}_\eta, \Omega)$ available for all nodes $\eta \in \mathcal{O}$ are also used to select the next node to be explored. More precisely, we select here a node $\eta \in \mathcal{O}$ which maximizes $\text{SR}(\mathcal{S}, \mathcal{S}_\eta, \Omega)$. This node is then split in two by considering possible instantiations of a variable $d_e, e \in E_\eta$, chosen among the variable equal to 1 in the optimal solution of program MIP_η. This branching strategy aims to maximally improve the current solution set $\mathcal{S}$. The cost vector y of any optimal solution of MIP_η indeed maximizes the value $\min_{x\in\mathcal{S}}\{f_\omega(x)\} - f_\omega(y)$ over all weights $\omega \in \Omega$ and all feasible cost vectors $y' \in \bigcup_{\eta'\in\mathcal{O}} \mathcal{S}_{\eta'}$.

The algorithm implementing these principles is summarized by Algorithm 3.

Algorithm 3: Branch & Bound procedure for multicriteria assignment problems

Input: $G = (N \cup T, E)$; Ω; $\mathcal{S}_0$: initial cost vectors

$\mathcal{S} \leftarrow \mathcal{S}_0$
$\eta \leftarrow [\emptyset, \emptyset]$
$\mathcal{O} \leftarrow \{\eta\}$
while $\mathcal{O} \neq \emptyset$ **do**
  Select a node η in $\arg\max_{\eta' \in \mathcal{O}} \mathrm{SR}(\mathcal{S}, \mathcal{S}_{\eta'}, \Omega)$
  if $|\mathcal{S}_\eta| = 1$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_\eta$
  else if $\mathrm{SR}(\mathcal{S}, \mathcal{S}_\eta, \Omega) \geq -\delta$ **then**
    Select $e \in E_\eta$ such that $d_e = 1$ in the optimal solution of MIP_η
    Generate node $\eta^0 = [E^0_\eta \cup \{e\}, E^1_\eta]$ and node $\eta^1 = [E^0_\eta, E^1_\eta \cup \{e\}]$
    forall $\eta' \in \{\eta^0, \eta^1\}$ **do**
      if $\mathcal{S}_{\eta'} \neq \emptyset$ **and** $\mathrm{SR}(\mathcal{S}, \mathcal{S}_{\eta'}, \Omega) \geq -\delta$ **then**
        $\mathcal{O} \leftarrow \mathcal{O} \cup \{\eta'\}$
      end
    end
  end
  $\mathcal{O} \leftarrow \mathcal{O} \setminus \{\eta\}$
end
return $\mathcal{S}$

Recall that this regret-based branch and bound procedure is able to detect partial assignments that cannot be part of a near-optimal assignment (even if we further restrict the sets of admissible weights by collecting new preference statements from the agents, see Paragraph "evaluation and pruning").

5.3 Combining Incremental Elicitation and Search

The branch and bound procedure introduced in the previous subsection can be used to eliminate partial assignments that cannot be part of a near-optimal assignments. Nevertheless, it may be the case that there exist no near-optimal assignments for a given set Ω. The following proposition gives a sufficient condition to guanrantee the existence of a near-optimal solution:

Proposition 7. *If* $\mathrm{mMR}(\mathcal{S}, \Omega) \leq \delta$ *at the end of the search performed by Algorithm 3, then* $\mathrm{MR}(x^*, \mathcal{X}, \Omega) \leq \delta$ *for any cost vector* $x^* \in \arg\min_{x \in \mathcal{S}} \mathrm{MR}(x, \mathcal{S}, \Omega)$.

Proof. Let x^* be an element of $\arg\min_{x \in \mathcal{S}} \mathrm{MR}(x, \mathcal{S}, \Omega)$. To prove that we have $\mathrm{MR}(x^*, \mathcal{X}, \Omega) \leq \delta$, we need to prove that the inequality $\mathrm{PMR}(x^*, x, \Omega) \leq \delta$ holds for any cost vector $x \in \mathcal{X}$. If $x \in \mathcal{S}$, then we can directly infer the result since we have $\mathrm{PMR}(x^*, x, \Omega) \leq \mathrm{MR}(x^*, \mathcal{S}, \Omega) \leq \delta$ by hypothesis. Assume that $x \notin \mathcal{S}$. In that case, at some iteration step, Algorithm 3 pruned a node η such that $x \in S_\eta$ (see Paragraph "evaluation and pruning"). According to the pruning rule, there exists $\mathcal{S}' \subseteq \mathcal{S}$ such that $\mathrm{SR}(\mathcal{S}', \mathcal{S}_\eta, \Omega) < -\delta$. Hence, for all $\omega \in \Omega$, there exists $x' \in \mathcal{S}'$ such that $f_\omega(x') - f_\omega(x) < -\delta$ (by definition of SR). Moreover, since $\mathcal{S}' \subseteq \mathcal{S}$ and

$\mathrm{MR}(x^*, \mathcal{S}, \Omega) \leq \delta$, we have $f_\omega(x^*) - f_\omega(x') \leq \delta$ for all $x' \in \mathcal{S}'$ and for all $\omega \in \Omega$. Hence, $f_\omega(x^*) - f_\omega(x) < \delta - \delta \leq \delta$ for all $\omega \in \Omega$ and therefore $\mathrm{PMR}(x^*, x, \Omega) \leq \delta$. □

Thus, if we have $\mathrm{mMR}(\mathcal{S}, \Omega) \leq \delta$ at the end of the search performed by our branch and bound procedure, then any cost vector in $\arg\min_{x \in \mathcal{S}} \mathrm{MR}(x, \mathcal{S}, \Omega)$ is associated with a near-optimal assignment. Therefore, we now introduce a query generation strategy that collects preference information during the search performed by Algorithm 3 so as to ensure that this condition is satisfied at the end of the execution. This is made possible by the following simple result:

Proposition 8. *For all Ω, Ω' such that $\Omega' \subseteq \Omega$ and for all sets $X, Y \subseteq \mathcal{X}$:*

$$\mathrm{SR}(X, Y, \Omega) < -\delta \Rightarrow \mathrm{SR}(X, Y, \Omega') < -\delta$$

Hence, reducing Ω during the search do not invalidate any pruning operation made by Algorithm 3 while potentially increasing the number of pruning operations. In order to generate informative preference queries, we integrate the CSS strategy (presented in Section 2.2) to our search procedure as follows. Whenever the insertion of some cost vectors in $\mathcal{S}$ (see line 7) makes the value $\mathrm{mMR}(\mathcal{S}, \Omega)$ be strictly larger than δ, we repeatedly ask preference queries applying the CSS method on set $\mathcal{S}$ until this value is smaller than δ. By definition, this query generation strategy ensures that a near-optimal cost vector is returned by Algorithm 3 (see Proposition 7). The following example presents an execution of the proposed interactive branch and bound procedure:

Example 4. *Consider the multicriteria assignment problem with 4 agents, 2 criteria and $\delta = 0$, given in Figure 6.*

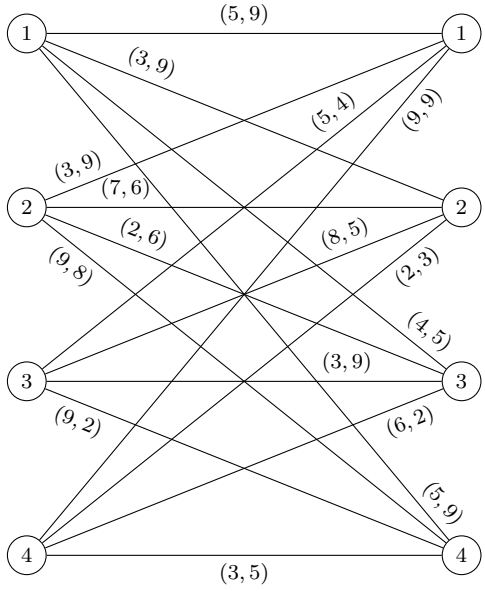


Fig. 6 An instance of bipartite graph $G = (N \cup T, E)$ with four agents and $\mathcal{Q} = \{1, 2\}$.

Initially, we have $\Omega = \{(\omega_1, \omega_2) \in int(\mathbb{R}^2_+) : \omega_1 + \omega_2 = 1\}$. *Note that* Ω *is completely characterized by the constraint* $0 < \omega_1 < 1$ *since* ω_2 *is implicitly defined by the normalization constraint. We assume here that the DM's preferences are represented by the weighting vector* $\omega_0 = (0.4, 0.6)$. *The assignments* $\{(1,2),(2,1),(3,3),(4,4)\}$ *and* $\{(1,1),(2,2),(3,4),(4,3)\}$ *are optimal when only considering the first and the second criterion respectively. These solutions are associated with the cost vectors* $(12, 32)$ *and* $(27, 19)$ *respectively. Hence, we set* $\mathcal{S} = \mathcal{S}_0 = \{(12,32),(27,19)\}$ *(see line 1). In the sequel, the nodes generated at iteration step* t *will be denoted by* η_t^0 *and* η_t^1 *(line 10).*

At the first iteration of the while loop, the node $\eta = [\emptyset, \emptyset]$ *is selected for branching. Here we have to solve* MIP_η *to be able to generate the new nodes (see lines 9-10). Since* $\{(1,4),(2,3),(3,1),(4,2)\}$ *is an optimal solution, we can choose* $e = (1,4)$ *at this iteration step and generate the following nodes:* $\eta_1^0 = [\{(1,4)\}, \emptyset]$ *and* $\eta_1^1 = [\emptyset, \{(1,4)\}]$. *Since* $\mathcal{S}_{\eta_1^0} \neq \emptyset$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_1^0}, \Omega) \approx 4.18 \geq -\delta = 0$, *node* η_1^0 *is inserted in* $\mathcal{O}$. *Moreover, since* $\mathcal{S}_{\eta_1^1} \neq \emptyset$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_1^1}, \Omega) \approx 4.43 \geq 0$, *node* η_1^1 *is also inserted in* $\mathcal{O}$. *Then, node* η *is removed from* $\mathcal{O}$ *(see line 17).*

At the second iteration step, we have $\mathcal{O} = \{\eta_1^0, \eta_1^1\}$ *where* $SR(\mathcal{S}, \mathcal{S}_{\eta_1^0}, \Omega) \approx 4.18$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_1^1}, \Omega) \approx 4.43$. *Since* $SR(\mathcal{S}, \mathcal{S}_{\eta_1^1}, \Omega) \geq SR(\mathcal{S}, \mathcal{S}_{\eta_1^0}, \Omega)$, *node* $\eta_1^1 = [\emptyset, \{(1,4)\}]$ *is selected for branching in line 5. Moreover, since assignment* $\{(1,4),(2,3),(3,1),(4,2)\}$ *is an optimal solution of* $MIP_{\eta_1^1}$, *we can select edge* $(2,3)$ *to generate the new nodes (see lines 9-10):* $\eta_2^0 = [\{(2,3)\}, \{(1,4)\}]$ *and* $\eta_2^1 = [\emptyset, \{(1,4),(2,3)\}]$. *Then, since* $\mathcal{S}_{\eta_2^0} \neq \emptyset$, $\mathcal{S}_{\eta_2^1} \neq \emptyset$, $SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx 0.79 \geq -\delta = 0$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_2^1}, \Omega) \approx 4.43 \geq 0$, *node* η_2^0 *and node* η_2^1 *are both inserted in* $\mathcal{O}$. *Finally, node* η_1^1 *is removed from* $\mathcal{O}$.

At the third iteration step, we have $\mathcal{O} = \{\eta_1^0, \eta_2^0, \eta_2^1\}$ *where* $SR(\mathcal{S}, \mathcal{S}_{\eta_1^0}, \Omega) \approx 4.18$, $SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx 0.79$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_2^1}, \Omega) \approx 4.43$. *Since we have* $SR(\mathcal{S}, \mathcal{S}_{\eta_2^1}, \Omega) \geq SR(\mathcal{S}, \mathcal{S}_{\eta_1^0}, \Omega) \geq SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega)$, *node* $\eta_2^1 = [\emptyset, \{(1,4),(2,3)\}]$ *is chosen in line 5. Moreover, since assignment* $\{(1,4),(2,3),(3,1),(4,2)\}$ *is an optimal solution of* $MIP_{\eta_2^1}$, *edge* $(3,1)$ *can be used to generate the new nodes:* $\eta_3^0 = [\{(3,1)\}, \{(1,4),(2,3)\}]$ *and* $\eta_3^1 = [\emptyset, \{(1,4),(2,3),(3,1)\}]$. *Then, since* $SR(\mathcal{S}, \mathcal{S}_{\eta_3^0}, \Omega) \approx -3.96 < -\delta = 0$, *node* η_3^0 *is not inserted in set* $\mathcal{O}$. *On the other hand, we have* $\mathcal{S}_{\eta_3^1} \neq \emptyset$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_3^1}, \Omega) \approx 4.43 \geq 0$; *therefore* η_3^1 *is inserted in* $\mathcal{O}$. *Finally, node* η_2^1 *is removed from* $\mathcal{O}$.

At the fourth iteration step, we have $\mathcal{O} = \{\eta_1^0, \eta_2^0, \eta_3^1\}$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_3^1}, \Omega) \approx 4.43 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_1^0}, \Omega) \approx 4.18 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx 0.79$. *Therefore* $\eta_3^1 = [\emptyset, \{(1,4),(2,3),(3,1)\}]$ *is here selected for branching. Since* $\{(1,4),(2,3),(3,1),(4,2)\}$ *is an optimal solution of* $MIP_{\eta_3^1}$, *the following nodes are generated:* $\eta_4^0 = [\{(4,2)\}, \{(1,4),(2,3),(3,1)\}]$ *and* $\eta_4^1 = [\emptyset, \{(1,4),(2,3),(3,1),(4,2)\}]$. *Then, node* η_4^1 *is inserted in set* $\mathcal{O}$ *since we have* $\mathcal{S}_{\eta_4^1} \neq \emptyset$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_4^1}, \Omega) \approx 4.43 \geq -\delta = 0$. *However, node* η_4^0 *is eliminated in line 12 since we obviously have* $\mathcal{S}_{\eta_4^0} = \emptyset$.

At the fifth iteration step, we have $\mathcal{O} = \{\eta_1^0, \eta_2^0, \eta_4^1\}$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_4^1}, \Omega) \approx 4.43 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_1^0}, \Omega) \approx 4.18 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx 0.79$. *As a consequence, at this step, node* $\eta_4^1 = [\emptyset, \{(1,4),(2,3),(3,1),(4,2)\}]$ *is selected for branching. Since we have* $|\mathcal{S}_{\eta_4^1}| = 1$, *we set* $\mathcal{S} = \mathcal{S} \cup \mathcal{S}_{\eta_4^1} = \{(12,32),(27,19)\} \cup \{(14,22)\}$ *in line 7. Then, our query generation strategy requires that we compute* $\mathrm{mMR}(\mathcal{S}, \Omega)$. *Here we have* $\mathrm{mMR}(\mathcal{S}, \Omega) = \mathrm{MR}((14,22), \mathcal{S}, \Omega) = \mathrm{PMR}((14,22),(27,19), \Omega) \approx 3 > \delta$. *Therefore, the DM is asked to compare cost vectors* $(14, 22)$ *and* $(27, 19)$ *(following the CSS). Since* $f_{\omega_0}(14,22) = 18.8 \leq f_{\omega_0}(27,19) = 22.2$, *the DM states that* $(14, 22)$ *is better than* $(27, 19)$ *and so we have to update the set* Ω *by inserting the linear constraint* $f_\omega(14,22) \leq f_\omega(27,19)$, *i.e.* $\omega_1 \geq \frac{3}{16}$. *Thus* $\Omega = \{(\omega_1, 1-\omega_1) : \frac{3}{16} \leq \omega_1 < 1\}$. *Now we have* $\mathrm{mMR}(\mathcal{S}, \Omega) =$

$\mathrm{MR}((14,22),\mathcal{S},\Omega) = \mathrm{PMR}((14,22),(12,32),\Omega) \approx 2 > \delta = 0$. *Hence the DM is now asked to compare* $(14,22)$ *and* $(12,32)$. *Since* $f_{\omega_0}(14,22) = 18.8 \leq f_{\omega_0}(12,32) = 24$, *the DM states that she prefers the first cost vector to the second one and so we have to update* Ω *by inserting the linear constraint* $f_\omega(14,22) \leq f_\omega(12,32)$, *i.e.* $\omega_1 \leq \frac{5}{6}$. *Thus,* $\Omega = \{(\omega_1, 1-\omega_1) : \frac{3}{16} \leq \omega_1 < \frac{5}{6}\}$ *and* $\mathrm{mMR}(\mathcal{S},\Omega) \leq \delta$.

At the sixth iteration step, we have $\mathcal{O} = \{\eta_1^0, \eta_2^0\}$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_1^0}, \Omega) \approx 1.69 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx -0.5$. *Therefore, node* $\eta_1^0 = [\{(1,4)\}, \emptyset]$ *is selected for branching. Since assignment* $\{(1,3),(2,1),(3,4),(4,2)\}$ *is an optimal solution of* $MIP_{\eta_1^0}$, *edge* $(1,3)$ *is selected and the following nodes are generated:* $\eta_6^0 = [\{(1,4),(1,3)\}, \emptyset]$ *and* $\eta_6^1 = [\{(1,4)\}, \{(1,3)\}]$. *Here we have* $\mathcal{S}_{\eta_6^0} \neq \emptyset$, $\mathcal{S}_{\eta_6^1} \neq \emptyset$, $SR(\mathcal{S}, \mathcal{S}_{\eta_6^0}, \Omega) \approx 0.88 \geq -\delta = 0$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_6^1}, \Omega) \approx 1.69 \geq -\delta = 0$. *Therefore both nodes* η_6^0 *and* η_6^1 *are inserted in* $\mathcal{O}$ *and then node* η_1^0 *is removed from* $\mathcal{O}$ *in line 17.*

At the seventh iteration step, we have $\mathcal{O} = \{\eta_2^0, \eta_6^0, \eta_6^1\}$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_6^1}, \Omega) \approx 1.69 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_6^0}, \Omega) \approx 0.88 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx -0.5$. *Therefore node* $\eta_6^1 = [\{(1,4)\}, \{(1,3)\}]$ *is selected for branching. Since assignment* $\{(1,3),(2,1),(3,4),(4,2)\}$ *is an optimal solution of* $MIP_{\eta_6^1}$, *edge* $(2,1)$ *can be selected here and then the following nodes are generated:* $\eta_7^0 = [\{(1,4),(2,1)\}, \{(1,3)\}]$ *and* $\eta_7^1 = [\{(1,4)\}, \{(1,3),(2,1)\}]$. *Moreover we have* $\mathcal{S}_{\eta_7^0} \neq \emptyset$, $\mathcal{S}_{\eta_7^1} \neq \emptyset$, $SR(\mathcal{S}, \mathcal{S}_{\eta_7^0}, \Omega) \approx 0.69 \geq -\delta = 0$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_7^1}, \Omega) \approx 1.69 \geq -\delta = 0$. *As a consequence, both nodes* η_6^0 *and* η_6^1 *are inserted in set* $\mathcal{O}$. *Finally, node* η_6^1 *is removed from* $\mathcal{O}$.

At the eighth step, we have $\mathcal{O} = \{\eta_2^0, \eta_6^0, \eta_7^0, \eta_7^1\}$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_7^1}, \Omega) \approx 1.69 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_6^0}, \Omega) \approx 0.88 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_7^0}, \Omega) \approx 0.69 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx -0.5$. *Therefore, we have to choose node* $\eta_7^1 = [\{(1,4)\}, \{(1,3),(2,1)\}]$ *for branching. Since assignment* $\{(1,3),(2,1),(3,4),(4,2)\}$ *is an optimal solution of* $MIP_{\eta_7^1}$, *we select* $(3,4)$ *and we generate the following nodes:* $\eta_8^0 = [\{(1,4),(3,4)\}, \{(1,3),(2,1)\}]$ *and* $\eta_8^1 = [\{(1,4)\}, \{(1,3),(2,1),(3,4)\}]$. *Since we have* $\mathcal{S}_{\eta_8^1} \neq \emptyset$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_8^1}, \Omega) \approx 1.69 \geq -\delta = 0$, *node* η_8^1 *is inserted in set* $\mathcal{O}$. *However,* η_8^0 *is eliminated by our pruning rule since we have* $SR(\mathcal{S}, \mathcal{S}_{\eta_8^0}, \Omega) \approx -2.37 < -\delta = 0$. *Then, node* η_7^1 *is removed from* $\mathcal{O}$.

At the ninth iteration step, we have $\mathcal{O} = \{\eta_2^0, \eta_6^0, \eta_7^0, \eta_8^1\}$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_8^1}, \Omega) \approx 1.69 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_6^0}, \Omega) \approx 0.88 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_7^0}, \Omega) \approx 0.69 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx -0.5$. *Hence node* $\eta_8^1 = [\{(1,4)\}, \{(1,3),(2,1),(3,4)\}]$ *is selected for branching at this step. Since assignment* $\{(1,3),(2,1),(3,4),(4,2)\}$ *is an optimal solution of* $MIP_{\eta_8^1}$, *we select edge* $(4,2)$ *and generate the following nodes:* $\eta_9^0 = [\{(1,4),(4,2)\}, \{(1,3),(2,1),(3,4)\}]$ *and* $\eta_9^1 = [\{(1,4)\}, \{(1,3),(2,1),(3,4),(4,2)\}]$. *Moreover, since we have* $\mathcal{S}_{\eta_9^1} \neq \emptyset$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_9^1}, \Omega) \approx 1.69 \geq -\delta = 0$, *node* η_8^1 *is inserted in set* $\mathcal{O}$. *However, node* η_9^0 *is here eliminated since* $\mathcal{S}_{\eta_9^0} = \emptyset$. *Finally, node* η_8^1 *is removed from* $\mathcal{O}$ *in line 17.*

At the tenth iteration step, we have $\mathcal{O} = \{\eta_2^0, \eta_6^0, \eta_7^0, \eta_9^1\}$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_9^1}, \Omega) \approx 1.69 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_6^0}, \Omega) \approx 0.88 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_7^0}, \Omega) \approx 0.69 \geq SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx -0.5$. *Hence node* $\eta_9^1 = [\{(1,4)\}, \{(1,3),(2,1),(3,4),(4,2)\}]$ *is here selected for branching. Since* $|\mathcal{S}_{\eta_9^1}| = 1$, *we set* $\mathcal{S} = \mathcal{S} \cup \mathcal{S}_{\eta_9^1} = \{(12,32),(27,19),(14,22))\} \cup \{(18,19)\}$. *Now, following our query generation strategy, we have to compute* $\mathrm{mMR}(\mathcal{S},\Omega)$. *Here we have* $\mathrm{mMR}(\mathcal{S},\Omega) = \mathrm{MR}((14,22),\mathcal{S},\Omega) = \mathrm{PMR}((14,22),(18,19),\Omega) \approx 1.69 > \delta = 0$. *Hence the DM is asked to compare the cost vectors* $(14,22)$ *and* $(18,19)$. *Since* $f_{\omega_0}(14,22) = 18.8 \geq f_{\omega_0}(18,19) = 18.6$, *the DM states that she prefers* $(18,19)$ *to* $(14,22)$ *and so we have to update* Ω *by inserting the linear constraint* $f_\omega(18,19) \leq f_\omega(14,22)$, *i.e.* $\omega_1 \leq \frac{3}{7}$. *Thus,* $\Omega = \{(\omega_1, 1-\omega_1) : \frac{3}{16} \leq \omega_1 < \frac{3}{7}\}$ *and* $\mathrm{mMR}(\mathcal{S},\Omega) = \mathrm{MR}((18,19),\mathcal{S},\Omega) \leq \delta$.

Now we have $\mathcal{O} = \{\eta_2^0, \eta_6^0, \eta_7^0\}$, $SR(\mathcal{S}, \mathcal{S}_{\eta_6^0}, \Omega) \approx -0.57$, $SR(\mathcal{S}, \mathcal{S}_{\eta_7^0}, \Omega) \approx -1$ *and* $SR(\mathcal{S}, \mathcal{S}_{\eta_2^0}, \Omega) \approx -2.56$. *Since all these values are strictly below* 0, *we know that the corresponding nodes will be deleted by our pruning rule in the following steps (see line 8). Therefore, our algorithm necessarily returns* $\mathcal{S} = \{(12, 32), (27, 19), (14, 22), (18, 19)\}$ *and since* $\text{MR}((18, 19), \mathcal{S}, \Omega) \leq \delta$, *we know that* $(18, 19)$ *is the cost vector of an optimal solution, namely* $\{(1, 3), (2, 1), (3, 4), (4, 2)\}$. *Note that only* 3 *queries were needed here to find the best choice between the twenty four feasible assignments.*

5.4 Numerical Tests

In this section, we report the results of numerical tests aiming to evaluate the performance of our regret-based branch and bound procedure (Algorithm 3) combined with our query generation strategy (presented in Section 5.3) in terms of both computation time and number of preference queries. In Table 4, the results are obtained by averaging over 30 runs and linear optimizations are performed using the Gurobi library of Java. In our experiments, we consider instances of weighted bipartite graphs with 50 nodes (with edge densities of $d = 0.1, 0.2$) randomly generated as follows: each agent (resp. item) is linked to exactly $d \times n$ items (resp. agents) and the associated cost vectors are randomly drawn in $\{1, \ldots, 1000\}^q$. To study the impact of tolerance threshold δ on the performance of our procedure, we ran tests with two distinct values: $\delta = 0.01$ and $\delta = 0.05$. Moreover, the number of criteria q varies from 2 to 5. In these tests, we start with an empty set of preferences statements and DMs answer to queries according to a linear scalarizing function f_ω where ω is randomly chosen in $\{\text{int}(\mathbb{R}_+^q) : \sum_{j=1}^q \omega_j = 1\}$.

		$\delta = 0.01$		$\delta = 0.05$	
q	d	time	queries	time	queries
2	0.1	0.02	3.5	0.02	2.8
2	0.2	0.97	4.3	0.92	2.9
3	0.1	0.04	7.4	0.05	5.3
3	0.2	79.01	11.8	94.43	7.0
5	0.1	0.09	13.5	0.10	10.1
5	0.2	540.12	20.4	701.27	15.7

Table 4 Performance of our incremental branch and bound method (times in minutes).

Not surprisingly, we observe here also that reducing δ tends to decrease the number of queries which impacts negatively on computation times.

6 Conclusion

We have shown how to implement an incremental preference elicitation method based on regret minimization within three different solution algorithms frequently used to address combinatorial optimization problems, namely dynamic programming, greedy search and branch and bound. The common feature in these three

algorithms is that they are constructive, i.e., the optimal solution is constructed step by step by sequencing elementary decisions until the obtention of a complete optimal solution. Due to this decomposition, the difficult problem of eliciting preferences over a combinatorial domain reduces to a sequence of simpler problems consisting in eliciting the best elementary decision at the current step, given the past decisions. With this sequential elicitation, we significantly reduce the combinatorial aspect of the problem and therefore the elicitation burden while better focusing the search on the most promising part of the Pareto set.

This approach is made possible by the use of a linear aggregation function (a weighted sum) to define the overall value of a solution. Due to the linearity of this aggregation, it is indeed possible to construct an optimal solution from locally optimal elementary decisions. When the DM's preferences are too complex to be captured by a weighted sum, local preferences on partial solutions are less informative and the elicitation must preferably be made using complete solutions. For example, we proposed an adaptation of this incremental approach to deal with preferences represented by a Choquet integral in multiobjective state space search (Benabbou and Perny, 2015a) but the number of queries used to solve the problem and the solution times are less good than with linear models. An alternative approach consists of directly working on the set of complete Pareto optimal solutions. This approach is illustrated in (Bourdache and Perny, 2017) where the solution optimizing an ordered weighted average is obtained by combining a ranking algorithm used to enumerate complete solutions with an incremental weight elicitation process.

Beside the difficulty of eliciting preferences represented by non-linear models, two important issues should be investigated in the future to extend the state of the art in incremental preference elicitation on combinatorial domains. The first one concerns the analysis of the query complexity of the algorithms (i.e., the number of preference queries necessary to obtain the optimum in the worst case). We need to obtain formal guarantees and possibly to revise the algorithms so as to obtain polynomial upper bounds on query complexity. This is yet possible when preference queries are carefully selected to divide the range of an interval at every step. An example is given in (Benabbou and Perny, 2017) for the incremental elicitation of a utility function in the context of sequential decision making under risk; it could probably be adapted in the context of weight elicitation for multicriteria decision making. The second issue concerns the sensitivity of the elicitation process to the DM's answers. Preference queries are asked here without any redundancy, in order to minimize the number of preference queries. Questionnaires are designed in such a way that the DM never has the opportunity to contradict herself. However, whenever the DM expresses some instability in answering to preference queries or if the underlying model does not perfectly fit to her preferences, it may be the case that some implicit inferences made on preferences are not valid. This may impact the quality of the final recommendation. For this reason, an alternative approach could be to regularly check the consistency of the model with the observable preferences and to let the possibility to revise the model if inconsistencies appear. However, this interesting line of research is significantly more expensive in terms of query complexity and probably more difficult to implement on combinatorial domains.

References

Benabbou N, Perny P (2015a) Combining preference elicitation and search in multiobjective state-space graphs. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, pp 297–303

Benabbou N, Perny P (2015b) Incremental weight elicitation for multiobjective state space search. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA., pp 1093–1099

Benabbou N, Perny P (2015c) On possibly optimal tradeoffs in multicriteria spanning tree problems. In: Algorithmic Decision Theory - 4th International Conference, ADT 2015, Lexington, KY, USA, September 27-30, 2015, pp 322–337

Benabbou N, Perny P (2016) Solving multi-agent knapsack problems using incremental approval voting. In: ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands, pp 1318–1326

Benabbou N, Perny P (2017) Adaptive elicitation of preferences under uncertainty in sequential decision making problems. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, pp 4566–4572

Benabbou N, Gonzales C, Perny P, Viappiani P (2015) Minimax regret approaches for preference elicitation with rank-dependent aggregators based on Choquet integral. EURO journal on Decision processes 3(1,2):29–64

Benabbou N, Diodoro SDSD, Perny P, Viappiani P (2016) Incremental preference elicitation in multi-attribute domains for choice and ranking with the borda count. In: Proceedings of the 10th International Conference on Scalable Uncertainty Management SUM, pp 81–95

Benabbou N, Perny P, Viappiani P (2017) Incremental elicitation of Choquet capacities for multicriteria choice, ranking and sorting problems. vol 246, pp 152–180

Bourdache N, Perny P (2017) Anytime algorithms for adaptive robust optimization with OWA and WOWA. In: Proc. of the 5th international conference on Algorithmic Decision Theory (ADT), p to appear

Boutilier C, Brafman RI, Domshlak C, Hoos HH, Poole D (2004) Preference-based constrained optimization with CP-nets. Computational Intelligence 20(2):137–157

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

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

Braziunas D (2011) Decision-theoretic elicitation of generalized additive utilities. PhD thesis, University of Toronto

Braziunas D, Boutilier C (2007) Minimax regret based elicitation of generalized additive utilities. In: UAI, pp 25–32

Chajewska U, Koller D, Parr R (2000) Making rational decisions using adaptive utility elicitation. In: AAAI, pp 363–369

Dery LN, Kalech M, Rokach L, Shapira B (2014) Reaching a joint decision with minimal elicitation of voter preferences. Information Sciences 278:466–487

Domshlak C, Hüllermeier E, Kaci S, Prade H (2011) Preferences in AI: an overview. Artif Intell 175(7-8):1037–1052

Drummond J, Boutilier C (2014) Preference elicitation and interview minimization in stable matchings. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada., pp 645–653

Ehrgott M (2005) Multicriteria Optimization (2. ed.). Springer

Gelain M, Pini MS, Rossi F, Venable KB, Walsh T (2010) Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. Artificial Intelligence 174(3):270–294

Gilbert H, Spanjaard O, Viappiani P, Weng P (2015) Reducing the number of queries in interactive value iteration. In: Proceedings of ADT'15, pp 139–152

Gilbert H, Benabbou N, Perny P, Spanjaard O, Viappiani P (2017) Incremental decision making under risk with the weighted expected utility model. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, pp 4588–4594

Ha V, Haddawy P (1997) Problem-focused incremental elicitation of multi-attribute utility models. In: UAI, pp 215–222

Hamacher H, Ruhe G (1994) On spanning tree problems with multiple objectives. Annals of Operations Research 52:209–230

Hansen P (1980) Bicriterion path problems. In: Multiple criteria decision making theory and application, Springer Berlin Heidelberg, pp 109–127

Hart PE, Nilsson NJ, 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

Hines G, Larson K (2010) Preference elicitation for risky prospects. In: AAMAS, pp 889–896

Kaddani S, Vanderpooten D, Vanpeperstraete JM, Aissi H (2017) Weighted sum model with partial preference information: application to multi-objective optimization. European Journal of Operational Research 260:665–679

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

Kouvelis P, Yu G (1997) Robust Discrete Optimization and Its Applications. Kluwer

Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 7:48–50

Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2:83–97

Lu T, Boutilier C (2011) Robust approximation and incremental elicitation in voting protocols. In: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, pp 287–293

Mandow L, De la Cruz JLP (2005) A new approach to multiobjective A* search. In: International Joint Conference on Artificial Intelligence, pp 218–223

Perny P, Viappiani P, Boukhatem A (2016) Incremental preference elicitation for decision making under risk with the rank-dependent utility model. In: Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI 2016, June 25-29, 2016, New York City, NY, USA

Prim RC (1957) Shortest connection networks and some generalizations. Bell System Technical Journal 36:1389–1401

Regan K, Boutilier C (2009) Regret-based reward elicitation for Markov decision processes. In: UAI, pp 444–451

Salo A, Hämäläinen RP (2001) Preference ratios in multiattribute evaluation (PRIME)–elicitation and decision procedures under incomplete information. IEEE Trans on Systems, Man and Cybernetics 31(6):533–545

Savage LJ (1954) The Foundations of Statistics. Wiley

Stewart BS, White III CC (1991) Multiobjective A*. J ACM 38(4):775–814

Wang T, Boutilier C (2003) Incremental Utility Elicitation with the Minimax Regret Decision Criterion. pp 309–316

Weng P, Zanuttini B (2013) Interactive Value Iteration for Markov Decision Processes with Unknown Rewards. In: International Joint Conference on Artificial Intelligence

White III CC, Sage AP, 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