Choquet-based optimisation in multiobjective shortest path and spanning tree problems

Lucie Galand, Patrice Perny, Olivier Spanjaard

Choquet-based optimisation in multiobjective shortest path and spanning tree problems

Lucie Galand, Patrice Perny and Olivier Spanjaard

LIP6, Université Pierre et Marie Curie, Paris, France.

Abstract

This paper is devoted to the search of Choquet-optimal solutions in finite graph problems with multiple objectives. The Choquet integral is one of the most sophisticated preference models used in decision theory for aggregating preferences on multiple objectives. We first present a condition on preferences (name hereafter *preference for interior points*) that characterizes preferences favouring compromise solutions, a natural attitude in various contexts such as multicriteria optimisation, robust optimisation and optimisation with multiple agents. Within Choquet expected utility theory, this condition amounts to using a submodular capacity and a convex utility function. Under these assumptions, we focus on the fast determination of Choquet-optimal paths and spanning trees. After investigating the complexity of these problems, we introduce a lower bound for the Choquet integral, computable in polynomial time. Then we propose different algorithms using this bound, either based on a controlled enumeration of solutions (ranking approach) or an implicit enumeration scheme (branch and bound). Finally, we provide numerical experiments that show the actual efficiency of the algorithms on multiple instances of different sizes.

Key words: Multiobjective discrete optimisation, Choquet integral, Shortest path problem, Minimum spanning tree problem, Submodular capacity.

1 Introduction

Most algorithmic works dealing with optimisation problems in graphs aim at developing efficient search procedures to determine one or several optimal solutions within a combinatorial set of possibilities. Classical optimisation problems such as shortest path problems or minimal spanning tree problems involve a single cost function, additively decomposable over the arcs (or edges). The effort is put on the design of constructive procedures taking advantage of this decomposability to construct an optimal solution from locally optimal sub-solutions. However, the growing complexity of real applications (e.g. optimisation of communication networks, transportation problems, planning) has led researchers to consider problems including

Email address: {lucie.galand,patrice.perny,olivier.spanjaard}@lip6.fr (Lucie Galand, Patrice Perny and Olivier Spanjaard).

additional sources of complexity such as uncertainty, multi-source information, multicriteria analysis, bringing new problems and new algorithmic challenges for computer scientists, mathematicians and operations researchers. For example, we consider here optimisation problems in multivalued graphs. Such problems appear when the value of a solution cannot be represented by a single decomposable cost-function, and must be assessed from several dimensions (criteria, agents or scenarios, depending on the problem). Such problems appear frequently in the field of multicriteria or multiagent optimisation, but also in the field of robust optimisation under risk or uncertainty. This explains the growing interest for multiobjective optimisation in the recent years [7,8].

As soon as multiple objectives are considered in the evaluation of a solution, the notion of optimality is not straightforward and various options are available in the literature on decision theory and multicriteria analysis to characterize the most-preferred solutions. Among them, the concept of Pareto optimality is one of the most widely used. A solution is said to be Pareto-optimal if it cannot be improved on one objective without being depreciated on another one. At first sight, Pareto optimality seems natural because it does not require much preference information from the Decision Maker and can be used as a preliminary filter to circumscribe the set of reasonable solutions. However, in combinatorial optimisation problems, the complete enumeration of the set of Pareto-optimal solutions may be practically intractable because the size of the Pareto set grows, in worst case, exponentially with the size of the instance (see [20] for multiobjective shortest path problems, and [9,19] for multiobjective spanning tree problems). Hence, the computation of the set of Pareto solutions may induce prohibitive response times and require a very large memory space. Knowing this difficulty, it is worth spending some time on preference elicitation so as to get a finest preference model, able to discriminate between Pareto optimal solutions and to narrow the area of interest within the Pareto set.

Once the preference model of the decision maker is known in a multiobjective optimisation problem, there is no need to enumerate the entire set of Pareto-optimal solutions. Instead, the search can be focused on good compromise solutions for the decision maker. This notion of compromise is natural in multiobjective decision support and has counterparts in other optimisation contexts involving several dimensions. For example, in the context of multi-agent optimisation problems, the notion of compromise solution refers to equity or fairness (see e.g. [29]). In the context of optimisation under uncertainty, compromise solutions refer to the idea of robustness (see e.g. [25,42,33]). The quality of the compromise achieved can be measured using an overall utility function refining Pareto dominance to better discriminate between the various possible solutions. In this paper we will resort to the Choquet expected utility model, which is one of the most sophisticated decision criteria used in decision analysis [5,35,17]. It provides both a generalisation of weighted means and weighted ordered averages [44], enhancing their descriptive and prescriptive possibilities. A similar study might be carried out using other scalarising functions, e.g. Tchebycheff distance to the ideal point which is classically used in interactive multiobjective optimisation [39]. A first step in this direction has been done to solve multiobjective shortest path problems in state space graphs [13].

In this paper, we adress the problem of finding optimal paths and spanning trees on graphs endowed with multiple cost functions, where optimality refers to the minimisation of a Choquet

expected disutility function. The paper is organized as follows: in Section 2, we recall basic elements linked to Choquet integrals. Then we formulate a condition characterizing preferences favouring compromise solutions and discuss its impact on the Choquet expected disutility model. This leads us to consider the problem of minimising a Choquet integral with a submodular capacity over a set of cost-vectors attached to feasible paths or trees. In Section 3 we discuss the complexity of these problems and establish preliminary results that will be used later in the algorithms. Then, we propose original algorithms to find Choquet-optimal paths and spanning trees in a multivalued graph. They are either based on a ranking approach performing a controlled explicit enumeration of solutions or a branch and bound algorithm performing an implicit enumeration. In Sections 4 and 5 we report numerical experiments showing the practical efficiency of the proposed algorithms.

2 Compromise search using a Choquet integral

2.1 Notations and definitions

Let $G = (V, E)$ be a given directed or undirected graph, where V denotes the set of vertices and E the set of arcs (or edges). A feasible solution is a subset $X \subseteq E$ satisfying a given property (here, we will only consider paths or spanning trees). The edges are weighted according to n objectives $f_i : E \to \mathbb{N}$, $i \in N = \{1, \ldots, n\}$ interpreted as cost functions to be minimised. Thus, each edge $e \in E$ is weighted by a vector $f(e) = (f_1(e), \ldots, f_n(e))$. We assume here that cost functions are additive. Hence, the cost of any feasible solution $X \subseteq E$ is defined by vector $f(X) = \sum_{e \in X} f(e)$. The set of all feasible cost vectors is denoted $\mathcal{X}$. Comparing feasible solutions amounts to comparing their respective cost vectors in $\mathcal{X}$. For example, Pareto dominance can be stated as follows: a solution $X \subseteq E$ *Pareto-dominates* a solution $Y \subseteq E$ whenever its cost vector $x = f(X)$ is at least as "good" as $y = f(Y)$ on every component, and strictly "better" on at least one component. Formally this writes $x_i \leq y_i$ for all $i \in N$, and $x_j < y_j$ for some $j \in N$.

As usual in multiobjective problems, the most preferred solutions belong to the set of Pareto-optimal solutions i.e. those solutions that are dominated by no other feasible solution. However Pareto optimality is generally not sufficient to discriminate between multiple feasible alternatives and we need a finer preference model to characterize the type of compromise sought in the Pareto set. As a decision criterion refining Pareto dominance, we use here the Choquet integral [5] which is a compromise operator that aggregates costs using a weighting function defined on every subset of criteria. Formally the weights of subsets are represented by a *capacity*.

Definition 1 *A capacity is a set function* $v : 2^N \to [0, 1]$ *such that:* $v(\emptyset) = 0$, $v(N) = 1$, *and* $\forall A, B \in 2^N$, $A \subseteq B \Rightarrow v(A) \leq v(B)$.

For any subset $A \subseteq N$, $v(A)$ represents the importance of coalition A. The Choquet integral of a vector $x \in \mathbb{N}^n$ with respect to capacity v is then defined by:

$$C_v(x) = \sum_{i=1}^{n} \Big[v(X_{(i)}) - v(X_{(i+1)}) \Big] x_{(i)} \quad (1)$$

$$= \sum_{i=1}^{n} \Big[x_{(i)} - x_{(i-1)} \Big] v(X_{(i)}) \quad (2)$$

where (.) represents a permutation on $\{1, \ldots, n\}$ such that $0 = x_{(0)} \leq x_{(1)} \leq \ldots \leq x_{(n)}$, $X_{(i)} = \{j \in N,\ x_j \geq x_{(i)}\} = \{(i),(i+1),\ \ldots,\ (n)\}$ for $i \leq n$ and $X_{(n+1)} = \emptyset$. Note that $X_{(i+1)} \subseteq X_{(i)}$, hence $v(X_{(i)}) \geq v(X_{(i+1)})$ for all i. The Choquet integral generalizes the classical notion of average with the following interpretation based on Equation (2): for a given vector $x = (x_1, \ldots, x_n)$, the cost is greater or equal to $x_{(1)}$ on all criteria belonging to $X_{(1)}$, which represents a weight of $v(X_{(1)}) = 1$; then the cost is greater or equal to $x_{(2)}$ on all criteria belonging to $X_{(2)}$ which represents an increment of $x_{(2)} - x_{(1)}$ with weight $v(X_{(2)})$. The same applies from $x_{(2)}$ to $x_{(3)}$ for all criteria belonging to $X_{(3)}$ which weights $v(X_{(3)})$, and so on... The overall integral is therefore obtained by aggregation of marginal increments $x_{(i)} - x_{(i-1)}$ weighted by $v(X_{(i)})$.

In Decision Theory, the Choquet integral is often used in maximisation problems under the form $C_v(u(x_1), \ldots, u(x_n))$ where u is a utility function defined on payoffs, to be maximised [35]. In our context where costs replace payoffs we need to reformulate the criterion using a *disutility* function to be minimised. Assuming we work with integer costs belonging to $[1, M]$ where M is a positive integer, we will use a strictly increasing function $w : [0, M] \to \mathbb{R}^+$, such that $w(x)$ represents the disutility of cost x. The *Choquet Expected Disutility model* (CED) is then defined from function w and capacity v by:

$$\psi_v^w(x) = C_v(w(x_1), \ldots, w(x_n)) \quad (3)$$

Note that there always exists a cost vector x minimising $\psi_v^w(x)$ in the Pareto set of $\mathcal{X}$. This is a direct consequence of the componentwise non-decreasingness of ψ_v^w (provided w is at least non-decreasing) [17]. The CED model includes the classical weighted average as a particular case. It is indeed sufficient to set $w(x) = x$ for all x and to use an additive capacity v, i.e. a capacity such that $v(A) = \sum_{i \in A} v_i$ for all $A \subseteq N$, where $v_i = v(\{i\})$. Then we have $v(X_{(i)}) - v(X_{(i+1)}) = v_{(i)}$ for all i and $\psi_v^w(x) = \sum_{i=1}^n v_{(i)} w(x_{(i)}) = \sum_{i=1}^n v_i x_i$. When used with a non-additive capacity and/or with a non-linear disutility function w, it offers additional descriptive possibilities. As an illustration, let us consider the following:

Example 1 *Consider the bi-valued graph represented on Figure 1 (resp. Figure 2) and assume that we are looking for an optimal spanning tree (resp. path from S to T) w.r.t. two criteria (or agents or scenarios). For simplicity, the instances given in Figure 1 and 2 are constructed in such a way that the image of the set of feasible solutions in the biobjective space is the same in both problems. It is pictured on the right part of Figure 3. We can easily see on this figure that the Pareto-optimal points are p_1, p_3, p_4, p_7, p_{10} and p_{12} (bold lines in the table given in the left part of Figure 3) representing the possible tradeoffs within the Pareto set. In order to discriminate between the Pareto-optimal solutions, we might use the CED model with a non-*

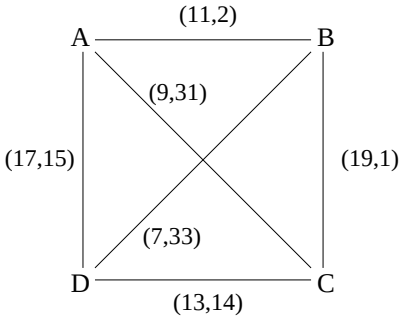

Fig. 1. Biobjective spanning tree.

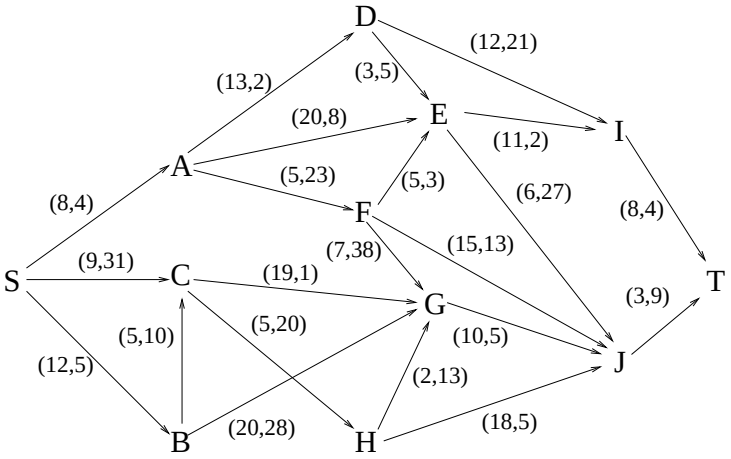

Fig. 2. Biobjective shortest path.

	Spanning trees	Paths	Cost vectors
X_1	**{AB,AC,BD}**	**⟨S,A,F,E,J,T⟩**	$\boldsymbol{p_1 = (27, 66)}$
X_2	{AC,BD,CD}	⟨S,C,H,G,J,T⟩	$p_2 = (29, 78)$
X_3	**{AB,BD,CD}**	**⟨S,A,F,J,T⟩**	$\boldsymbol{p_3 = (31, 49)}$
X_4	**{AB,AC,CD}**	**⟨S,A,D,E,J,T⟩**	$\boldsymbol{p_4 = (33, 47)}$
X_5	$\{AD, AC, BD\}$	$\langle S, A, F, G, J, , T\rangle$	$p_5 = (33, 79)$
X_6	$\{AC, BD, BC\}$	$\langle S, C, H, J, T\rangle$	$p_6 = (35, 65)$
X_7	**{AB,BC,BD}**	**⟨S,A,F,E,I,T⟩**	$\boldsymbol{p_7 = (37, 36)}$
X_8	$\{AB, AC, AD\}$	$\langle S, A, E, J, T\rangle$	$p_8 = (37, 48)$
X_9	$\{AD, BD, CD\}$	$\langle S, B, C, H, G, J, T\rangle$	$p_9 = (37, 62)$
X_{10}	**{AD,AB,CD}**	**⟨S,A,D,I,T⟩**	$\boldsymbol{p_{10} = (41, 31)}$
X_{11}	$\{AC, BC, CD\}$	$\langle S, C, G, J, T\rangle$	$p_{11} = (41, 46)$
X_{12}	**{AB,BC,CD}**	**⟨S,A,D,E,I,T⟩**	$\boldsymbol{p_{12} = (43, 17)}$
X_{13}	$\{AD, BC, BD\}$	$\langle S, B, C, H, J, T\rangle$	$p_{13} = (43, 49)$
X_{14}	$\{AD, AC, BC\}$	$\langle S, B, G, J, T\rangle$	$p_{14} = (45, 47)$
X_{15}	$\{AD, AB, BC\}$	$\langle S, A, E, I, T\rangle$	$p_{15} = (47, 18)$
X_{16}	$\{AD, BC, CD\}$	$\langle S, B, C, G, J, T\rangle$	$p_{16} = (49, 30)$

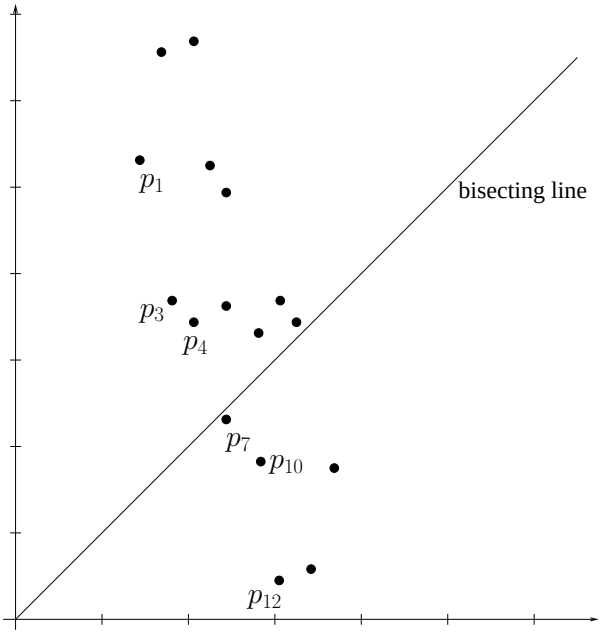

Fig. 3. Feasible solutions in Example 1.

additive capacity, e.g. $v(\{1\}) = 0.9$, $v(\{2\}) = 0.5$ *and* $v(\{1, 2\}) = 1$ *and a convex disutility function* $w(x) = x^2$ *which leads to the CED values given in Table 1.*

Table 1
CED values of Pareto-optimal solution (case 1).

Solutions	X_1	X_3	X_4	X_7	X_{10}	X_{12}
ψ_v^w	2361	1609	1593	1358	1573	1615

In this example we can see that the most preferred solution is X_7, then X_{10}, X_4, X_3, X_{12} and X_1. Clearly this model favours compromise solutions. Conversely, using a different capacity and a concave disutility, we might favour another type of compromise solutions, more contrasted and less consensual. For example, choosing $v(\{1\}) = 0.5$, $v(\{2\}) = 0.1$, $v(\{1, 2\}) = 1$ and $w(x) = 100\sqrt{x}$ leads to the CED values given in Table 2.

Table 2
CED values of Pareto-optimal solutions (case 2).

Solutions	X_1	X_3	X_4	X_7	X_{10}	X_{12}
ψ_v^w	549	571	586	604	599	534

We can see that X_{12} is now the optimal solution, then X_1, X_3 and so on. Tables 1 and 2 show that the optimal compromise solution found in the Pareto set depends on the choice of the capacity and the shape of the utility function. As mentioned in the introduction, we are interested here in determining fair compromise solutions in multicriteria decision problems (or robust solutions in multi-scenarios problems or consensual solutions in multi-agent problems) such as those put forward in Table 1. In order to formalize this idea, the next subsection introduces an axiom of preference for interior points and investigates its impact on the choice of both capacities and utilities in the CED model.

2.2 Preference for interior points

Preference for compromise solutions means intuitively that smoothing or averaging a cost vector makes the decision maker better off. This intuitive idea can be formalized using an axiom initially named *"preference for diversification"* [3] due to its interpretation in the context of portfolio management. This axiom can be reformulated in our framework as follows:

Definition 2 (Preference for interior points) *A preference relation $\succsim$ defined on cost vectors in $\mathbb{N}^n$ satisfies* preference for interior points *if, for any $x^1, \ldots, x^p \in \mathbb{N}^n$, and for all $\alpha_1, \ldots, \alpha_p \geq 0$ such that $\sum_{i=1}^p \alpha_i = 1$, we have:*

$$[x^1 \sim x^2 \sim \ldots \sim x^p] \Rightarrow \sum_{i=1}^{p} \alpha_i x^i \succsim x^k, \quad k = 1, \ldots, p \tag{4}$$

where $\sim$ is the symmetric part of $\succsim$ (indifference relation).

This axiom says that any compromise cost vector obtained by a convex combination of p indifferent cost vectors improves these vectors. Interestingly enough, Chateauneuf and Tallon have shown that, within the Choquet expected utility theory, this axiom on preference is equivalent to choosing a convex capacity v and a concave utility u [3], the convexity and concavity of a capacity being classically defined as follows:

Definition 3 *A capacity v is said to be* convex *(or supermodular) when $v(A \cup B) + v(A \cap B) \geq v(A) + v(B)$ for all $A, B \subseteq N$, and it is said to be* concave *(or submodular) when $v(A \cup B) + v(A \cap B) \leq v(A) + v(B)$ for all $A, B \subseteq N$.*

The direct counterpart of Chateauneuf and Tallon's result in the case of cost minimisation says that we should use the CED model with a *concave* capacity v and a *convex* disutility w to exhibit preference for interior points. For this reason, throughout the paper, we will assume that v is concave and w is convex (we recall that w is also assumed to be strictly increasing).

Table 1 provides examples of evaluations obtained from the CED model with a concave capacity and a convex disutility. We can see that solutions presenting a well-balanced profile receive a lower overall disutility. Another clear illustration of preference for interior points as defined in Equation (4) appears with a concave capacity such as $v(\{1\}) = 0.8$, $v(\{2\}) = 0.4$, $v(\{1,2\}) = 1$ and a convex disutility function such as $w(x) = x^2$. Here, the evaluations of solutions according to the CED model are the following (see Table 3).

Table 3
CED values of Pareto-optimal solutions (case 3).

Solutions	X_1	X_3	X_4	X_7	X_{10}	X_{12}
ψ_v^w	2180	1537	1537	1354	1537	1537

We can observe that points p_3, p_4, p_{10} and p_{12} are indifferent, with an overall disutility of 1537. Hence, any point inside the trapezoid formed by these points (see the grey area on Figure 4) is preferred (or indifferent) to these points. This is the case of p_7 that lies within the grey area. We can indeed see that p_7 is evaluated to 1354 which is a better overall disutility than 1537 obtained for p_3, p_4, p_{10} and p_{12}.

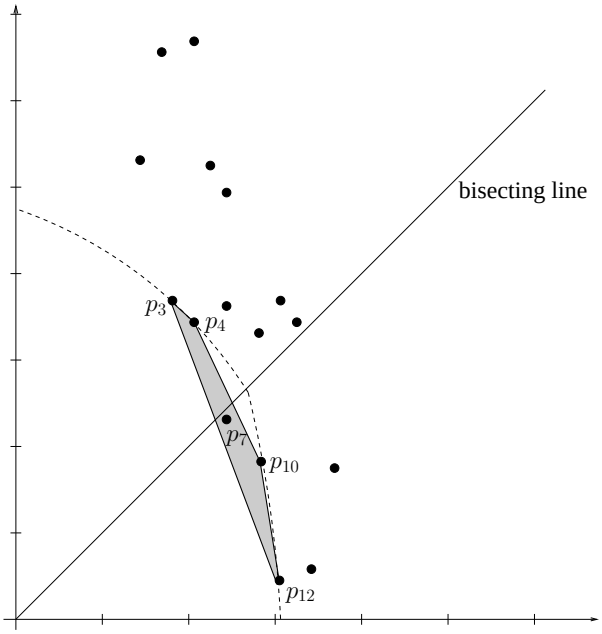

Fig. 4. Preference for interior points in Example 1.

3 Choquet optimisation in graph problems

3.1 Problems formulation and complexity issues

Assuming we use a concave capacity v and a convex disutility function w, we are now interested in finding Choquet-optimal spanning trees or paths in a multivalued graph. Choquet integral

can be seen as the Lovász extension of capacity v and as such, is convex if and only if v is concave [26]. Since w is also convex, ψ_v^w is convex for any concave capacity v. Under these hypotheses, the Choquet optimisation problems studied in this paper can be stated as follows:

CHOQUET-OPTIMAL SPANNING TREE PROBLEM (ψ_v^w-ST)

Input: a finite connected graph $G = (V, E)$, n integer-valued objectives f_i on E,

Goal: we want to determine a spanning tree X^* on G such that $\psi_v^w(f(X^*)) = \min_{X \in \mathcal{T}} \psi_v^w(f(X))$ where $\mathcal{T}$ is the set of all spanning trees on G.

CHOQUET-OPTIMAL PATH PROBLEM (ψ_v^w-P)

Input: a finite connected digraph $G = (V, E)$ with a source node s and a sink node t, n integer-valued objectives f_i on E,

Goal: we want to determine a path X^* such that $\psi_v^w(f(X^*)) = \min_{X \in \mathcal{P}} \psi_v^w(f(X))$, where $\mathcal{P}$ is the set of all paths from s to t in G.

When choosing $v(A) = 1$ for all non-empty $A \subseteq N$, we get a concave capacity and CED can be simplified as follows: $\psi_v^w(x) = \sum_{i=1}^{n} \left[w(x_{(i)}) - w(x_{(i-1)})\right] v(X_{(i)}) = w(x_{(n)}) = \max_{i \in N} w(x_i)$. Hence the determination of a Choquet-optimal solution reduces, in this case, to a min-max optimisation problem. Even when there are only two objective functions, the min-max spanning tree problem has been proved NP-hard by Hamacher and Ruhe [19] and the min-max shortest path problem has been proved NP-hard by Yu and Yang [45]. Consequently, the determination of a Choquet-optimal solution is NP-hard for both problems. However, there exist simple particular subclasses of problems that can be polynomially solved. For example, when $w(x) = x$ for all x and v is an additive capacity, then ψ_v^w is linear and ψ_v^w-ST (resp. ψ_v^w-P) boils down to the classical minimal spanning tree (resp. shortest path) problem.

When preferences over vectors are represented by the CED model, resorting to classical constructive approaches like dynamic programming (for ψ_v^w-P) or greedy search (for ψ_v^w-ST) is not easy. The two following examples illustrate the main difficulties to overcome when trying to adapt dynamic programming and greedy approaches to cope with the CED model in ψ_v^w-P and ψ_v^w-ST problems respectively.

Example 2 *Consider the graph of the biobjective shortest path problem given in Figure 2. In this graph, path $P_1 = \langle S, C\rangle$ and path $P_2 = \langle S, B, C\rangle$ are two different paths from S to C with cost vectors $(9, 31)$ for P_1 and $(17, 15)$ for P_2. With capacity $v(\{1\}) = 0.8$, $v(\{2\}) = 0.4$, $v(\{1, 2\}) = 1$ and convex disutility function $w(x) = x^2$, P_2 is preferred to P_1 w.r.t. ψ_v^w since $\psi_v^w(17, 15) = 276.2$ and $\psi_v^w(9, 31) = 433$. However consider now paths P_1', the cost vector of which is $(28, 32)$, and P_2', the cost vector of which is $(36, 16)$, obtained from P_1 and P_2, respectively, by adding arc (C, G). We can observe that the preference is reversed w.r.t. ψ_v^w since $\psi_v^w(28, 32) = 880$ and $\psi_v^w(36, 16) = 1088$, and hence P_1' is preferred to P_2'. Thus we see that an optimal path (here P_1') can include a non-optimal subpath (here P_1). In other terms, the Bellman principle does not hold.*

Example 3 *Consider the graph of the biobjective spanning tree problem given in Figure 1.*

In the spirit of the greedy approach, we might want to construct the optimal spanning tree by iteratively adding an edge e to a current subtree T such that $\psi_v^w(f(T \cup \{e\}))$ is minimal among all possible acyclic completions of T by one edge. Using the capacity and the disutility function given in Example 2, the best edge w.r.t. ψ_v^w is edge AB ($\psi_v^w(11,2) = 97.6$). The current tree is then $T = \{AB\}$. The optimal choice of the second edge is edge CD with cost $(13,14)$, since $\psi_v^w(24,16) = 512$ is minimal among all ψ_v^w-values of two-edge subtrees including edge AB. The current tree is then $T = \{AB, CD\}$. For the choice of the third edge, all remaining edges (i.e. edges BC, AC, AD or BD) are convenient since they lead to spanning trees with the same ψ_v^w-value (1537). For example, adding edge BC leads to spanning tree $T = \{AB, CD, BC\}$ with cost $(43,17)$, and $\psi_v^w(43,17) = 1537$. Nevertheless this tree is suboptimal. We can indeed consider tree $\{AB, BC, BD\}$, with cost $(37,36)$ that gives $\psi_v^w(37,36) = 1354.4 < 1537$. This shows that the greedy approach is not appropriate here.

The two previous examples show that we cannot simply extend the standard algorithms (Dijkstra or Bellman for paths, Kruskal or Prim for spanning trees) to determine an optimal solution w.r.t. ψ_v^w. In order to design exact solution methods for NP-hard problems, it is usual to proceed to a relaxation of the solution space so as to make the optimisation easier. This provides bounds on the value of the best solution. We proceed here differently. Instead of relaxing the solution space, we relax the non linear objective function to one which is easier to optimise. More precisely, we use linear scalarising functions that provide bounds using standard algorithms for shortest path and minimal spanning tree problems. This process is explained in the following subsections.

3.2 A default approximation of the Choquet integral

Before introducing the default approximation, we recall some definitions linked to the core of a capacity. Remark first that, to any capacity v, we can associate a *dual* capacity $\bar{v}$ defined by $\bar{v}(A) = 1 - v(N \setminus A)$ for all $A \subseteq N$. Note that the bidual capacity $\bar{\bar{v}}$ is equal to v. Moreover $\bar{v}$ is concave whenever v is convex and vice-versa. Finally, when v is concave, we have $v(A) + v(N \setminus A) \geq 1$, hence $\bar{v}(A) \leq v(A)$ and we can soundly define the *core* of capacity $\bar{v}$ as follows:

Definition 4 *The core of a capacity $\bar{v}$ is defined by:*

$$core(\bar{v}) = \{\lambda \in \mathcal{L} : \bar{v}(A) \leq \lambda(A) \leq v(A)\}$$

where $\mathcal{L}$ is the set of additive capacities defined on 2^N.

As shown by Shapley [38], when v is concave, the core of $\bar{v}$ is non-empty. Using this notion of non-emtpy core, Schmeidler [35,36] gives an intuitive interpretation of the Choquet expected utility as the minimum of a family of expected utilities to be maximised. This result is also related to the work of Edmonds in submodular optimisation [6,11], knowing that the Choquet integral appears as the value returned by a greedy algorithm used to maximise a linear function over a polymatroid. A useful by-product of these results on the CED model is the following:

Proposition 1 *Let v be a concave capacity. For all additive capacities $\lambda \in core(\bar{v})$ characterized by positive coefficients $(\lambda_1, \ldots, \lambda_n)$ such that $\lambda(A) = \sum_{i \in A} \lambda_i$ $(\forall A \subseteq N)$, we have $\psi_v^w(x) \geq \sum_{i=1}^n \lambda_i w(x_i)$. Moreover, if w is convex we have: $\psi_v^w(x) \geq w(\sum_{i=1}^n \lambda_i x_i)$.*

PROOF. $\psi_v^w(x) = \sum_{i=1}^n \left[w(x_{(i)}) - w(x_{(i-1)})\right] v(X_{(i)}) \geq \sum_{i=1}^n \left[w(x_{(i)}) - w(x_{(i-1)})\right] \lambda(X_{(i)})$ since inequality $v(X_{(i)}) \geq \lambda(X_{(i)})$ holds for all i ($\lambda \in core(\bar{v})$). Furthermore, $\sum_{i=1}^n \left[w(x_{(i)}) - w(x_{(i-1)})\right]$ $\lambda(X_{(i)}) = \sum_{i=1}^n [\lambda(X_{(i)}) - \lambda(X_{(i+1)})] w(x_{(i)}) = \sum_{i=1}^n \lambda_{(i)} w(x_{(i)})$ by definition of λ. Then we have $\sum_{i=1}^n \lambda_{(i)} w(x_{(i)}) = \sum_{i=1}^n \lambda_i w(x_i) \geq w(\sum_{i=1}^n \lambda_i x_i)$ provided w is convex. □

Hence, any weighting vector $\lambda \in \mathbb{R}_+^n$ characterizing an additive capacity in $core(\bar{v})$ can be used to produce a default approximation $w(\lambda x)$ of $\psi_v^w(x)$. Among the natural choices for λ, we can use $\phi = (\phi_1, \ldots, \phi_n)$, where ϕ_i is the Shapley value of criterion i. It represents the average marginal contribution of criterion i to coalitions [38]. Shapley values are positive coefficients adding-up to one and defined by:

$$\phi_i = \sum_{K \subseteq N \setminus \{i\}} \frac{(n - |K| - 1)!|K|!}{n!} (\bar{v}(K \cup \{i\}) - \bar{v}(K)).$$

Another possible choice for the weights is to determine a weighting vector $\lambda^* = (\lambda_1^*, \ldots, \lambda_n^*)$ with the maximal entropy, i.e. a solution to the following optimisation problem:

$$\begin{aligned} \max \quad & -\sum_{i=1}^n \lambda_i \log \lambda_i \\ \text{s.t.} \ & \sum_{i \in A} \lambda_i \leq v(A), \quad \forall A \subseteq N \\ & 0 \leq \lambda_i \leq 1, \quad i = 1, \ldots, n. \end{aligned} \tag{5}$$

As suggested by Jaffray [21], this optimisation problem can easily be solved by a greedy algorithm (see Algorithm 1 hereafter). An example of such admissible weights is given in Table 4 when $n = 3$. Starting with a concave capacity v we define the dual $\bar{v}$ and two additive capacities in the core of $\bar{v}$, respectively characterized by Shapley values $\phi = (0.5, 0.2, 0.3)$ and weights maximising entropy $\lambda^* = (0.4, 0.3, 0.3)$.

Table 4
Core of a capacity and associated weights

	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{2,3\}$	$\{1,3\}$	N
v	0	0.6	0.3	0.4	0.8	0.6	0.9	1
ϕ	0	0.5	0.2	0.3	0.7	0.5	0.8	1
λ^*	0	0.4	0.3	0.3	0.7	0.6	0.7	1
$\bar{v}$	0	0.4	0.1	0.2	0.6	0.4	0.7	1

We can easily check that for any set of criteria $A \subseteq N$ we have: $\bar{v}(A) \leq \phi(A) \leq v(A)$ and $\bar{v}(A) \leq \lambda^*(A) \leq v(A)$. In a combinatorial problem, the best solution according to additive capacity ϕ or λ^* is easy to determine as soon as the standard version of the problem is polynomially solvable. Thanks to Proposition 1, it provides a lower bound on the value of a ψ_v^w-optimal solution. In the following subsections, we present two ways to use this lower bound for Choquet optimisation, namely under a ranking approach or under an implicit enumeration procedure.

Algorithm 1: computing λ^* with max-entropy

Initialisation:
$A \leftarrow \emptyset$;
$B \leftarrow \emptyset$;
while $B \neq N$ **do**
  Select A in $\arg\min\{\frac{v(B\cup F)-v(B)}{|F|}, F \subseteq N\backslash B, F \neq \emptyset\}$;
  for all $i \in A$ **do**
    $\lambda_i^* \leftarrow \frac{v(B\cup A)-v(B)}{|A|}$;
  $B \leftarrow B \cup A$;
Output: $(\lambda_1^*, \ldots, \lambda_n^*)$

3.3 The ranking approach for Choquet optimisation

When used with a concave capacity and a convex disutility, the Choquet expected disutility function is convex but non-linear. In order to determine ψ_v^w-optimal spanning trees and paths we first propose adopting a ranking approach. The idea of the ranking approach is first to approximate function ψ_v^w by another function g which is easier to optimise, then to rank feasible solutions according to g until a stopping condition is met, ensuring that a ψ_v^w-optimal solution is found. This approach has been successfully used in similar contexts to optimise other non-linear functions over combinatorial domains, see [13–16,19,31]. It is based on a 3-steps procedure:

Step 1 [SCALARISATION] We consider the linear function $\varphi_\lambda(x) = \sum_{i=1}^n \lambda_i x_i$ where $(\lambda_1, \ldots, \lambda_n)$ are positive coefficients adding-up to one defining an additive capacity in $core(\bar{v})$. This function is useful to compute a lower bound on values $\psi_v^w(x)$. By Proposition 1 we know indeed that $\psi_v^w(x) \geq w(\varphi_\lambda(x))$ for all $x \in \mathbb{R}^n$. Furthermore, for any feasible cost vector $x = f(X)$ of a feasible solution X, the value $\varphi_\lambda(x)$ is nothing else but the value of X in graph G endowed with the scalar valuation $f_0(e) = \sum_i \lambda_i f_i(e)$. We have indeed $f_0(X) = \sum_{e\in X} f_0(e) = \sum_{e\in X} \sum_i \lambda_i f_i(e) = \sum_i \lambda_i \sum_{e\in X} f_i(e) = \sum_i \lambda_i f_i(X) = \varphi_\lambda(f(X))$.

Step 2 [RANKING] We rank feasible solutions in G with respect to f_0. More precisely, we generate a sequence $X^1, X^2, \ldots, X^k$ of feasible solutions in such a way that: $f_0(X^i) \leq f_0(X^j)$ whenever $i < j$. For problem ψ_v^w-P, we rank paths by increasing f_0-values using Jimenez and Marzal's algorithm [22] which is a lazy version of Eppstein's ranking algorithm [10]. For problem ψ_v^w-ST, we rank spanning trees by increasing f_0-values using Katoh, Ibaraki and Mine's algorithm [24] which is a sophistication of Gabow's algorithm [12]. By construction,

to each solution X^i generated during Step 2 corresponds a cost vector $x^i = f(X^i)$ such that $\varphi_\lambda(x^1) \leq \varphi_\lambda(x^2) \leq \ldots \leq \varphi_\lambda(x^k)$. We now have to identify a ψ_v^w-optimal x^i to stop the enumeration as soon as possible. This is achieved by the next step.

Step 3 [STOPPING CONDITION] The ranking procedure is stopped as soon as we reach a solution X^k with cost x^k such that $w(\varphi_\lambda(x^k)) \geq \psi_v^w(x^{\sigma(k)})$ where $\sigma(k)$ is the smallest integer such that $\psi_v^w(x^{\sigma(k)}) = \min_{i \in [\![1,k]\!]} \psi_v^w(x^i)$. This cut is justified by the following result:

Proposition 2 *Let $x^1, ..., x^k$ be the k-best cost vectors generated in Step 2, and $\sigma(k)$ the index of one minimising ψ_v^w among them (e.g. $\sigma(k)$ is the smallest integer such that $\psi_v^w(x^{\sigma(k)}) = \min_{i \in [\![1,k]\!]} \psi_v^w(x^i)$). If $w(\varphi_\lambda(x^k)) \geq \psi_v^w(x^{\sigma(k)})$ then $x^{\sigma(k)}$ is ψ_v^w-optimal, i.e. $\psi_v^w(x^{\sigma(k)}) = \min_{x \in \mathcal{X}} \psi_v^w(x)$.*

PROOF. Assume that condition $w(\varphi_\lambda(x^k)) \geq \psi_v^w(x^{\sigma(k)})$ holds. By construction we have an increasing sequence $\varphi_\lambda(x^1) \leq \varphi_\lambda(x^2) \leq \ldots \leq \varphi_\lambda(x^k)$. Moreover, for any $i > k$ we have $\varphi_\lambda(x^i) \geq \varphi_\lambda(x^k)$ and therefore $w(\varphi_\lambda(x^i)) \geq w(\varphi_\lambda(x^k))$ since w is strictly increasing. By proposition 1 we also have $\psi_v^w(x^i) \geq w(\varphi_\lambda(x^i))$. Hence we have $\psi_v^w(x^i) \geq w(\varphi_\lambda(x^i)) \geq w(\varphi_\lambda(x^k)) \geq \psi_v^w(x^{\sigma(k)})$. Consequently, for all $i > k$, $\psi_v^w(x^i) \geq \psi_v^w(x^{\sigma(k)})$ which shows that no solution found after step k in the ranking can improve the current best solution $X^{\sigma(k)}$ with cost vector $x^{\sigma(k)}$. □

Figure 5 illustrates the behaviour of the procedure. Assuming there exist p distinct feasible solutions, the white dots represent the increasing sequence $w(\varphi_\lambda(x^i))$, $i = 1, \ldots, p$ and the black dots represent the corresponding values $\psi_v^w(x^i)$, $i = 1, \ldots, p$. The stepwise increasing curve shows the evolution of $w(\varphi_\lambda(x^i))$ with x^i, $i = 1, \ldots, p$ while the stepwise decreasing curve shows the evolution of $\psi_v^w(x^{\sigma(i)})$, $i = 1, \ldots, p$. The stopping condition is fulfilled as soon as the two stepwise curves cross each other. This is achieved at step k.

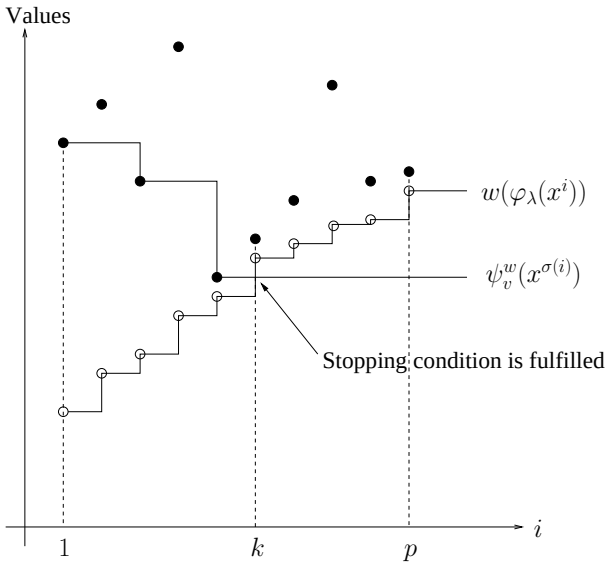

Fig. 5. Representation of the stopping condition

Algorithm 2 summarises the procedure. We denote by $\sigma(i)$ the index of the current best solution found at step i.

Algorithm 2: Choquet Optimisation

Determine the best solution X^1 w.r.t. f_0;
$\sigma(1) \leftarrow 1$; $best \leftarrow \psi_v^w(x^1)$; $i \leftarrow 1$;
repeat
  $i \leftarrow i+1$;
  Determine the i^{th} best solution X^i w.r.t f_0;
  if $\psi_v^w(x^i) < best$ **then**
    $\sigma(i) \leftarrow i$;
    $best \leftarrow \psi_v^w(x^i)$
  else
    $\sigma(i) \leftarrow \sigma(i-1)$
until $w(\varphi_\lambda(x^i)) \geq best$;
Output: solution $X^{\sigma(i)}$, its cost vector $x^{\sigma(i)}$ and value *best*

Note that, when stopping at iteration k, the complexity of Algorithm 2 is $O(|E|+|V|\log|V|+k)$ for ψ_v^w-P and $O(k|E|+\min(|V|^2, |E|\log\log|V|)$ for ψ_v^w-ST. These complexities are the ones of the ranking algorithms used, see [22,24].

Example 4 *We now illustrate the 3 steps of the ranking approach on the graphs of Figures 1 and 2 simultaneously, using ψ_v^w with a concave capacity $v(\{1\}) = 0.8$, $v(\{2\}) = 0.4$, $v(\{1,2\}) = 1$ and a convex disutility function $w(x) = x^2$.*

Step 1. *[Scalarisation] As scalarising function, we use φ_λ with $\lambda = (0.6, 0.4)$, obtained by Algorithm 1, to get an additive capacity in the core of $\bar{v}$. Hence, each edge e of the graph with cost $f(e) = (x_1, x_2)$ receives a scalar weight $f_0(e) = \varphi_\lambda(x_1, x_2) = 0.6x_1 + 0.4x_2$.*

Step 2 and 3. *[Algorithm 2] The behaviour of the algorithm may be visualised in the bidimensional space (see Figure 6) thanks to the following property: if two distinct points x and y both lie on the bisecting line, then $w(\varphi_\lambda(x)) > \psi_v^w(y)$ if and only if $x_i > y_i$, $i = 1, 2$ (strict Pareto-dominance). We have indeed $w(\varphi_\lambda(x)) = w(x_1) = w(x_2)$ since x belongs to the bisecting line; moreover $\psi_v^w(y) = w(y_1) = w(y_2)$ since y also belongs to the bisecting line; finally $w(x_i) > w(y_i)$ if and only if $x_i > y_i$, $i = 1, 2$ since w is strictly increasing.*

At iteration 1 of the ranking algorithm (see Figure 6), the best feasible cost vector (minimal with respect to φ_λ) is $p_{12} = (43, 17)$ with $\varphi_\lambda(43, 17) = 32.6$. Line Δ_λ represents all points having the same value than p_{12} with respect to φ_λ. Hence point N at the intersection of Δ_λ and the bisecting line is equivalent to p_{12} with respect to φ_λ. Moreover, point M is equivalent to p_{12} with respect to ψ_v^w (they are on the same isopreference curve). Since N strictly dominates M we can graphically observe that $w(\varphi_\lambda(p_{12})) < \psi_v^w(p_{12})$. Hence the stopping condition does not hold.

At iteration 2, we find the second best solution with respect to φ_λ. It is $p_{15} = (47, 18)$ with $\varphi_\lambda(47, 18) = 35.4$. Line Δ_λ has moved and represents now all points having the same value than p_{15} with respect to φ_λ. Hence point N' at the intersection of Δ_λ and the bisecting line is equivalent to p_{15} with respect to φ_λ. Moreover, point M (featuring variable best *in Algorithm 2)*

remains unchanged since p_{15} does not improve p_{12} in terms of ψ_v^w. Since N' strictly dominates M we can graphically observe that $w(\varphi_\lambda(p_{15})) < \psi_v^w(p_{12})$. Hence the stopping condition does not hold.

At iteration 3, we find the third best solution with respect to φ_λ. It is $p_7 = (37, 36)$ with $\varphi_\lambda(37, 36) = 36.6$. Line Δ_λ has moved and represents now all points having the same value as p_7 with respect to φ_λ. Hence point N'' at the intersection of Δ_λ and the bisecting line is equivalent to p_7 with respect to φ_λ. Moreover, point M is replaced by point M' since p_7 improves p_{12} in terms of ψ_v^w. Since N'' strictly dominates M' we can graphically observe that $w(\varphi_\lambda(p_7)) < \psi_v^w(p_7)$. Hence the stopping condition does not hold.

At iteration 4, we find the fourth best solution with respect to φ_λ. It is $p_{10} = (41, 31)$ with $\varphi_\lambda(41, 31) = 37$. Line Δ_λ has moved and represents now all points having the same value than p_{10} with respect to φ_λ. Hence point N''' at the intersection of Δ_λ and the bisecting line is equivalent to p_{10} with respect to φ_λ. Moreover, point M' remains unchanged since p_{10} does not improve p_7 in terms of ψ_v^w. Since N''' is strictly dominated by M' we can graphically observe that $w(\varphi_\lambda(p_{10})) > \psi_v^w(p_7)$. Hence the stopping condition holds which completes the ranking process.

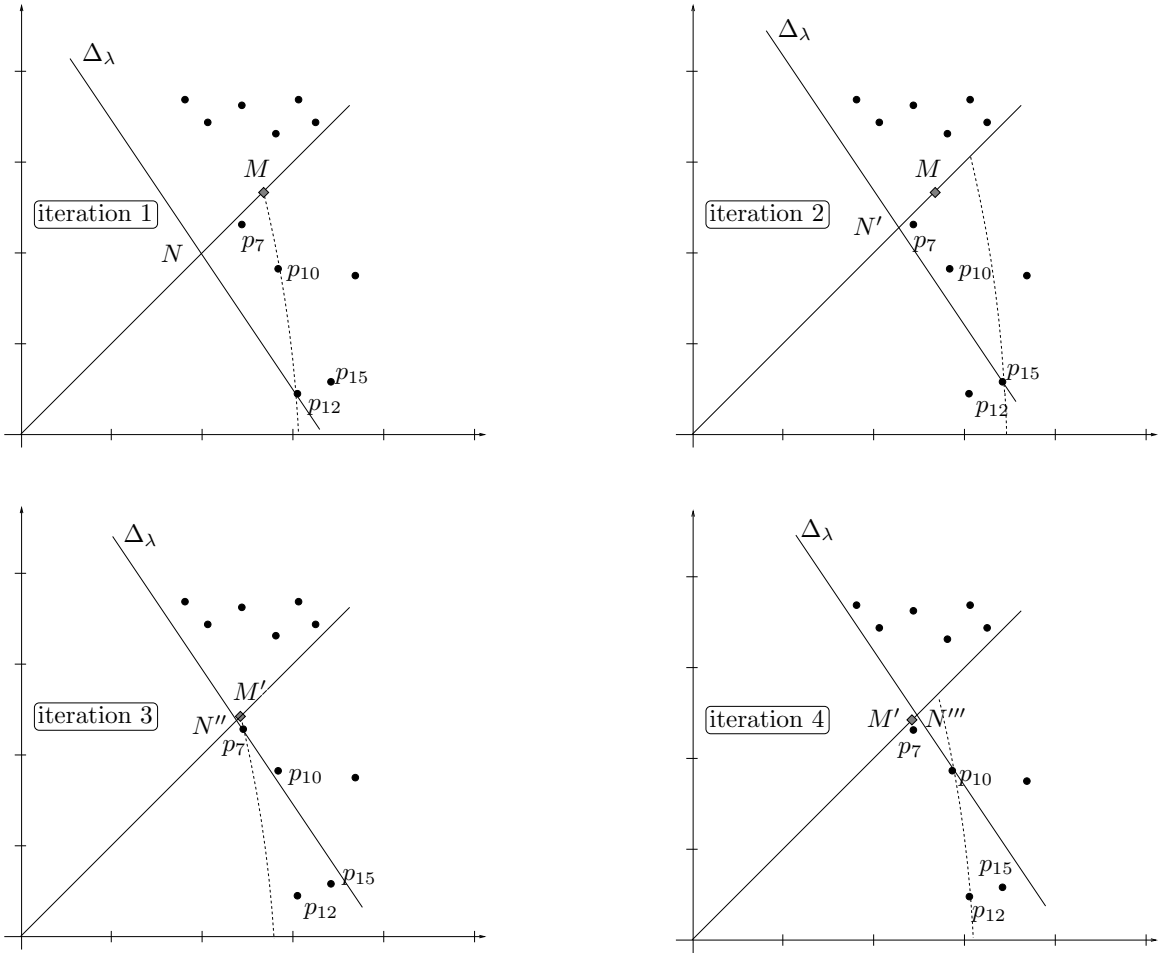

Fig. 6. Application of the ranking algorithm to Example 1.

3.4 Implicit enumeration

3.4.1 Branch and bound procedure for the Choquet-optimal spanning tree problem

The lower bound on $\psi_v^w(x)$ established in Proposition 1 can easily be used in the bounding phase of a branch and bound method. We propose here resorting to such a method, where the space of solutions is split into subspaces of solutions during the exploration. In the search tree, a node η is characterized by:

- $\text{IN}(\eta)$ a set of mandatory edges,
- $\text{OUT}(\eta)$ a set of forbidden edges.

Moreover, at each node η we define:

- $\mathcal{T}(n)$ the subset of spanning trees implicitly defined by $\text{IN}(\eta)$ and $\text{OUT}(\eta)$ (spanning trees X such that $\text{IN}(\eta) \subseteq X \subseteq E \setminus \text{OUT}(\eta)$),
- $T(\eta)$ a minimal spanning tree in $\mathcal{T}(\eta)$ for valuation f_0, i.e. $T(\eta) \in \arg\min_{X \in \mathcal{T}(\eta)} f_0(X)$,
- $\mathbf{LB}(\eta)$ a lower bound on ψ_v^w values at node η (detailed in Paragraph "Bounding" hereafter).

As usual, during the search, a node η is discarded when $\mathbf{LB}(\eta) \geq \mathbf{UB}$ where $\mathbf{UB}$ is the current incumbent (the best feasible ψ_v^w-value found so far). Let us now explain how the four classical phases (initialisation, branching, bounding and updating) of the branch and bound method are performed:

Initialisation. It is well known that a branch and bound method can be highly improved when a good solution is known before starting the search. In this concern, we propose to resort to an approximate version of the ranking approach previously presented (see Algorithm 2). It is indeed sufficient to stop the repeat loop as soon as the following relaxed condition hold: $(1+\varepsilon)w(\varphi_\lambda(x^i)) \geq best$. This is shown by the following:

Proposition 3 *If Algorithm 2 is stopped at step k such that $(1+\varepsilon)w(\varphi_\lambda(x^k)) \geq \psi_v^w(x^{\sigma(k)})$ then $(1+\varepsilon)\min_{x\in\mathcal{X}} \psi_v^w(x) \geq \psi_v^w(x^{\sigma(k)})$.*

PROOF. The proof can be derived from the one of Proposition 2 after modification of the stopping condition. By construction we have $\varphi_\lambda(x^i) \geq \varphi_\lambda(x^k)$ for all $i > k$ and therefore $w(\varphi_\lambda(x^i)) \geq w(\varphi_\lambda(x^k))$ since w is strictly increasing. Then we have, thanks to the relaxed stopping condition, $(1+\varepsilon)\psi_v^w(x^i) \geq (1+\varepsilon)w(\varphi_\lambda(x^i)) \geq (1+\varepsilon)w(\varphi_\lambda(x^k)) \geq \psi_v^w(x^{\sigma(k)})$. Thus, for all $i > k$, $(1+\varepsilon)\psi_v^w(x^i) \geq \psi_v^w(x^{\sigma(k)})$ which shows that no solution found after step k in the ranking can improve the current best solution with a ratio better than $(1+\varepsilon)$. □

This method makes it possible to find a near ψ_v^w-optimal spanning tree with an approximation ratio $1+\varepsilon$. Indeed, we have observed that such a solution is generally quickly computed even when ε is small. Computing this solution at the root of the search tree makes it possible to avoid the exploration of some subspaces in which a ψ_v^w-optimal spanning tree cannot be present.

Branching. At each node η of the search tree, edge e is put into $\text{IN}(\eta)$ or $\text{OUT}(\eta)$, where e is one of the best remaining edges (e minimises $\varphi_\lambda(f(e))$ over $E \setminus (\text{IN}(\eta) \cup \text{OUT}(\eta))$). Then the search space is split into two subspaces by creating two nodes η' and η'' such that:

- $\text{IN}(\eta') = \text{IN}(\eta) \cup \{e\}$ and $\text{OUT}(\eta') = \text{OUT}(\eta)$,
- $\text{IN}(\eta'') = \text{IN}(\eta)$ and $\text{OUT}(\eta'') = \text{OUT}(\eta) \cup \{e\}$.

Note that when edge e creates a cycle in $\text{IN}(\eta)$, node η'' is the only node to be created.

Bounding. We use here two complementary bounds. The first bound is obtained as follows. Let $f_i^*(\eta)$ be the minimum spanning tree in $\mathcal{T}(\eta)$ for valuation $f_i, i = 1, \ldots, n$ and $f^*(\eta) = (f_1^*(\eta), \ldots, f_n^*(\eta))$ the ideal cost vector in $\mathcal{T}(\eta)$. The components of this vector are easily obtained by n runs of a standard minimal spanning tree algorithm for valuations $f_i, i = 1, \ldots, n$ successively. By construction, for all $i = 1 \ldots n$ we have $f_i^*(\eta) \leq f_i(T)$ for any $T \in \mathcal{T}(\eta)$ and therefore $\psi_v^w(f^*(\eta)) \leq \psi_v^w(f(T))$ since ψ_v^w is componentwise non-decreasing. Hence $\psi_v^w(f^*(\eta))$ bounds the ψ_v^w-values at node η.

The second bound is obtained using valuation f_0. Using a standard minimal spanning tree algorithm, we determine a minimal spanning tree $T(\eta) \in \mathcal{T}(\eta)$ for valuation f_0. By proposition 1 we have indeed: $\forall T \in \mathcal{T}(\eta), \psi_v^w(f(T)) \geq w(f_0(T(\eta)))$ which gives the second bound on ψ_v^w-values. The two bounds make it possible to define the evaluation function $\mathbf{LB}(\eta)$ at a node η of the search-tree by: $\mathbf{LB}(\eta) = \max\{\psi_v^w(f^*(\eta)), w(f_0(T(\eta)))\}$ which requires to compute $n+1$ minimal spanning trees $T_0, \ldots, T_n$ for valuations $f_0, f_1, \ldots, f_n$ successively. The complexity of Kruskal's algorithm for determining a minimal spanning tree is $O(|E| \log |V|)$ when using a union-find data structure [40]. The overall complexity for the computation of $\mathbf{LB}(\eta)$ is therefore $O(n|E| \log |V|)$. Note that the $n+1$ sortings of edges according to $f_0, \ldots, f_n$ (in Kruskal's algorithm) are performed once and for all at the root of the search tree.

Updating the incumbent. At each node η the branch and bound algorithm checks whether $\min_{i \in \{0,\ldots,n\}} \psi_v^w(f(T_i))$ improves $\mathbf{UB}$ in which case $\mathbf{UB}$ is updated. We take here advantage of the fact that the lower bound is obtained from feasible solutions.

Algorithm 3 summarises the branch and bound procedure. First, note that the leaves of the search tree are characterised by $|\text{IN}(\eta)| = |V| - 1$ for any leaf η. Hence $\mathcal{T}(\eta)$ is a singleton containing tree $T(\eta)$ and $\mathbf{UB}$ is set to $\psi_v^w(f(T(\eta)))$. For the initial call of algorithm $BB(\eta, \mathbf{UB})$, root node η is characterized by empty sets IN and OUT, and $\mathbf{UB}$ (corresponding to the current optimal ψ_v^w-value) is initialised using Algorithm 2 with the relaxed stopping condition given in Proposition 3. For simplicity of presentation, we deliberately omitted the management of buffer variables storing the current ψ_v^w-optimal spanning tree and its cost vector.

3.4.2 Label setting algorithm for the Choquet-optimal path problem

We now present an implicit enumeration algorithm to compute a ψ_v^w-optimal path. This algorithm is based on a multiobjective extension of Dijkstra's algorithm proposed by Martins [27] to determine the set of Pareto-optimal paths from s to all other nodes. Before detailing the

Algorithm 3: $BB(\eta, \mathbf{UB})$

```
/* Bounding */
if LB(η) < UB then
    if |IN(η)| = |V| − 1 then
        UB ← ψ_v^w(f(T(η)));
        return UB
    else
        /* Updating the incumbent */
        for i = 0 to n do
            if ψ_v^w(f(T_i)) < UB then UB ← ψ_v^w(f(T_i))
        /* Branching */
        Select an edge e minimising f_0 in E\(IN(η) ∪ OUT(η))
        Create node η' such that: IN(η') = IN(η) ∪ {e} and OUT(η') = OUT(η);
        best ← BB(η', UB);
        UB ← min{UB, best};
        Create node η'' such that: IN(η'') = IN(η) and OUT(η'') = OUT(η) ∪ {e};
        best ← BB(η'', UB);
        return min{UB, best};
else
    return UB
```

way it can be taken out of its initial objective to focus on ψ_v^w-optimal paths, we first describe the original Martins procedure. It uses sets of labels for each node, representing Pareto-optimal cost vectors among detected subpaths. A label ℓ is a triplet $[v, P, f]$, where v is the node under consideration (i.e., to which ℓ is attached), P is a subpath from s to v, and f is the cost vector of P. In the following, node v to which ℓ is attached is denoted by v_ℓ, the corresponding subpath is denoted by P_ℓ, and its cost vector by f_ℓ. The algorithm proceeds by expanding labels, instead of nodes in Dijkstra's algorithm. In the multiobjective version of Dijkstra's algorithm, several labels can indeed be assigned to the same node, since several Pareto-optimal subpaths can reach this node. At each node v, the set of assigned labels, denoted by $\mathcal{L}(v)$, is divided into two disjoint subsets: the set of temporary labels (yet to be expanded), and the set of definitive labels (already expanded). We denote by TL the set of all temporary labels in the graph. In the procedure, the labels are compared according to their f-value. By abuse of language, we shall say that a label ℓ is Pareto-optimal in a set $\mathcal{L}$ if f_ℓ is Pareto-optimal in $\{f_\ell : \ell \in \mathcal{L}\}$. Following this convention, Martins procedure can be described as follows:

Step 1. Node s is labelled by $\ell_0 = [s, \langle s \rangle, (0, \ldots, 0)]$, $TL = \{\ell_0\}$ and $\mathcal{L}(v) = \emptyset$ for all $v \neq s$.

Step 2. At every iteration, a Pareto-optimal label ℓ in TL is expanded: for each outgoing arc $e = (v_\ell, v)$ a new label $\ell' = [v, \langle P_\ell, v \rangle, f_\ell + f(e)]$ is obtained (where $\langle P_\ell, v \rangle$ is the extension of path P_ℓ to node v). For each successor node v, only Pareto-optimal labels in $\mathcal{L}(v) \cup \{\ell'\}$ are kept (dominated labels are discarded). This set is denoted by $OPT(\mathcal{L}(v) \cup \{\ell'\})$. Label ℓ is then moved from TL to the set of definitive labels.

Step 3. When set TL becomes empty, the procedure stops. For all nodes v, $\{P_\ell : \ell \in \mathcal{L}(v)\}$ is the set of Pareto-optimal paths from s to v.

Actually we do not need to explicitly store the Pareto-optimal paths at each node; standard bookkeeping techniques are used instead to recover the Pareto-optimal paths at the end of the procedure. For simplicity, we deliberately omit the details here.

Martins' procedure is pseudopolynomial for integer costs and a fixed number n of objectives. An essential invariant of the algorithm is indeed that every expanded label at node v corresponds to an actual Pareto-optimal subpath from s to v [7] (similarly to the invariant in Dijkstra's algorithm which guarantees that the label of every expanded node corresponds to an actual shortest path). Futhermore, there can be at most $(M+1)^n$ labels of distinct value at each node, where M denotes an upper bound on the maximum value of a path for all criteria. The total number of expanded labels is therefore bounded above by $|V|(M+1)^n$, which proves the pseudopolynomiality of Martins' procedure.

As mentionned previously, there always exists a ψ_v^w-optimal path in the set of Pareto-optimal paths. For Choquet-based optimisation, one could therefore use a two stage approach, where one would first generate the set of Pareto-optimal paths, and second determine a ψ_v^w-optimal one among them. However, one can expect that many labels generated in the first stage do not contribute to any ψ_v^w-optimal path. For this reason, it would be useful to detect such labels as soon as they are created, in order to get a significant speedup in the label setting algorithm. This is precisely what we now describe. Similarly to the approach of Murthy and Her [28] for the Min-Max shortest path problem, the label setting algorithm we propose includes a pruning technique. This pruning technique is akin to the two complementary bounds used in the branch and bound procedure of the previous subsection.

For the first bound, let $h_i^*(v)$ be the value of a shortest path from v to t for valuation $f_i, i = 1,\ldots,n$ and $h^*(v) = (h_1^*(v),\ldots,h_n^*(v))$ the ideal cost vector at node v. The components of this vector are easily obtained by n runs of a standard shortest path algorithm to compute the shortest paths from t to all other nodes in the reverse graph (graph obtained by reversing all edges). By construction, for all ℓ in $\mathcal{L}(v)$, $\psi_v^w(f_\ell + h^*(v))$ bounds the ψ_v^w-value of any path extending P_ℓ.

The second bound is obtained thanks to valuation f_0. For all nodes v we determine $h_0(v)$, the value of the shortest path from v to t according to valuation f_0. This can be achieved easily by one run of Dijkstra's algorithm from t to all nodes in the reverse graph. By proposition 1 we have the following property: for all ℓ in $\mathcal{L}(v)$, $w(f_0(P_\ell) + h_0(v))$ is a lower bound on the ψ_v^w-values of all paths extending P_ℓ.

As in the branch and bound algorithm, these two bounds make it possible to define the evaluation function $\mathbf{LB}(\ell)$ of label ℓ by: $\mathbf{LB}(\ell) = \max\{\psi_v^w(f_\ell + h^*(v_\ell)), w(f_0(P_\ell) + h_0(v_\ell))\}$. This lower bound has a twofold advantage. It allows to prune any label ℓ whenever $\mathbf{UB} \leq \mathbf{LB}(\ell)$ where $\mathbf{UB}$ is the ψ_v^w-value of the best path from s to t found so far. Moreover it can be used as a heuristic to select the next node to be expanded, by giving priority to labels minimising $\mathbf{LB}(\ell)$. Algorithm 4 summarises the search procedure we use to determine a ψ_v^w-optimal path.

It is worth noting that Algorithm 4 preserves the pseudopolynomiality of Martin's procedure thanks to the following proposition:

Proposition 4 *If ψ_v^w is defined from a strictly increasing disutility function w and a strictly increasing capacity v with respect to set inclusion ($A \subset B \Rightarrow v(A) < v(B)$), then any label ℓ expanded by Algorithm 4 corresponds to a Pareto-optimal path from s to v_ℓ.*

PROOF. By contradiction, let us assume that a label ℓ corresponding to a dominated path from s to v_ℓ is expanded. There exists a Pareto-optimal path P' from s to v_ℓ such that $f(P')$ Pareto-dominates f_ℓ. This path necessarily includes a node with a label ℓ' in TL corresponding to a Pareto-optimal subpath of P' (otherwise label ℓ should have been discarded at node v_ℓ by the label associated to P').

Since $P_{\ell'}$ is a subpath of P' we know that $f_{\ell'} + h^*(v_{\ell'})$ is at least as good as $f(P') + h^*(v_\ell)$ on every component (i). Moreover, by assumption, $f(P')$ Pareto-dominates f_ℓ and therefore $f(P') + h^*(v_\ell)$ Pareto-dominates $f(\ell) + h^*(v_\ell)$ (ii). By combining (i) and (ii) we obtain that $f_{\ell'} + h^*(v_{\ell'})$ Pareto-dominates $f(\ell) + h^*(v_\ell)$ which implies that $\psi_v^w(f_{\ell'} + h^*(v_{\ell'})) < \psi_v^w(f(\ell) + h^*(v_\ell))$ (*) since ψ_v^w is componentwise increasing (by strict increasingness of v).

Similarly we have $f_0(P_{\ell'}) + h_0(v_{l'}) \leq f_0(P') + h_0(v_l)$ (iii) since $P_{\ell'}$ is a subpath of P'. Moreover, $f_0(P') < f_0(P_\ell)$ since $f(P')$ Pareto-dominates f_ℓ and f_0 is componentwise increasing (the components of the weighting vector λ defining an additive capacity in the core are necessarily strictly positive since v is strictly increasing). Hence $f_0(P') + h_0(v_l) < f_0(P_\ell) + h_0(v_l)$ (iv). From (iii) and (iv) we derive $f_0(P_{\ell'}) + h_0(v_{l'}) < f_0(P_\ell) + h_0(v_l)$ and then $w(f_0(P_{\ell'}) + h_0(v_{l'})) < w(f_0(P_\ell) + h_0(v_l))$ (**) since w is strictly increasing.

By (*) and (**) we get $\mathbf{LB}(\ell) > \mathbf{LB}(\ell')$ which is in contradiction with the fact that ℓ is expanded. □

4 Experimental results for the Choquet-optimal path problem

We first test the ranking approach (Algorithm 2) and the label-setting algorithm (Algorithm 4) on the ψ_v^w-P problem. These algorithms are implemented in C++ and the experimentations are performed on randomly drawn instances with an Intel Pentium Core 2 computer with 2.66Ghz. We have used a convex disutility function $w(x) = x^2$ and two types of concave and strictly increasing capacities:

- capacities defined by $v_1(A) = \sqrt{\sum_{i \in A} p_i}$ for all set $A \subseteq N$ where p_i's are strictly positive coefficients adding-up to one,
- capacities defined by $v_2(A) = 1 - \sum_{E \cap A = \emptyset} m(E)$ where sets $\{m(E) : E \subseteq N\}$ are stricly positive coefficients (Möbius masses) adding up to 1.

Note that this second type of capacities corresponds to plausibility measures since they are defined as the dual of a belief function constructed from positive Möbius masses (for more details, see Shafer [37]). We performed multiple tests using different capacities of both types, obtained by random draw of coefficients $p_i, i = 1, \ldots, n$ or masses $m(E), E \subseteq N$. It can easily be checked that such capacities are concave by construction and thus their dual capacities have

Algorithm 4: A label setting algorithm for the ψ_v^w-P problem

$\mathcal{L}(s) \leftarrow \{\ell_0\}$; **for** $v \neq s$ **do** $\mathcal{L}(v) \leftarrow \emptyset$;
create label $\ell_0 = [s, \langle s \rangle, (0, \ldots, 0)]$;
$TL \leftarrow \{\ell_0\}$; **UB** $\leftarrow +\infty$; $\ell \leftarrow \ell_0$;
while $[TL \neq \emptyset$ and **LB**$(\ell) <$ **UB**$]$ **do**
  begin
    remove ℓ from TL;
    if $n_\ell = t$ **then**
      $P^* \leftarrow P_\ell$;
      $x^* \leftarrow f_\ell$;
      UB $\leftarrow \psi_v^w(f_\ell)$;
    else
      for each node v such that $e = (v_\ell, v) \in E$ **do**
        begin
          create label $\ell' = [v, \langle P_\ell, v \rangle, f_\ell + f(e)]$;
          if LB$(\ell') <$ **UB then**
            $\mathcal{L}(v) \leftarrow OPT(\mathcal{L}(v) \cup \{\ell'\})$;
            if $\ell' \in \mathcal{L}(v)$ **then** $TL \leftarrow TL \cup \{\ell'\}$
          else discard label ℓ';
        end
    Select ℓ in arg $\min_{\ell' \in TL}$**LB**(ℓ') ;
  end
Output: solution P^* (a ψ_v^w-optimal path), its cost vector x^* and value **UB**

a non-empty core. To define φ_λ and therefore f_0, we use two types of additive measures in the core of $\bar{v}$ defined respectively by weighting vectors ϕ and λ^* as introduced in Subsection 3.2.

4.1 Ranking approach

The experimentations on the Choquet-optimal path problem were performed on randomly drawn graphs where the density is about 50% (the presence of every arc is randomly drawn with a probability of 0.5). The costs are randomly drawn between 1 and 100 for each arc and each objective. The number of objectives varies from 2 to 10. The number of nodes in the graph varies from 1.000 to 4.000. For each kind of instances (depending on the number of nodes and on the number of objectives), 50 different graphs are randomly drawn. Table 5 summarises the average execution times of the ranking approach on these instances.

The results show that most of the instances are solved within a second, even on instances with a large number of nodes. We observe that the average execution time grows slowly with the number of objectives and with the number of nodes. Furthermore, times are neither really sensitive to the type of capacity used nor to the choice of the additive capacity in the core (defined by λ^* or ϕ).

Table 5
Ranking approach for ψ_v^w-P: execution times (in seconds).

		2 obj.		3 obj.		5 obj.		10 obj.	
	$\|V\|$	λ_i^*	ϕ_i	λ_i^*	ϕ_i	λ_i^*	ϕ_i	λ_i^*	ϕ_i
	1000	0.05	0.05	0.06	0.06	0.06	0.07	0.09	0.11
v_1	2000	0.22	0.24	0.23	0.25	0.25	0.28	0.35	0.42
	3000	0.51	0.58	0.56	0.61	0.59	0.68	0.73	0.86
	4000	0.95	1.06	1.06	1.21	1.08	1.26	1.33	1.67
	1000	0.05	0.05	0.05	0.06	0.08	0.09	0.24	0.25
v_2	2000	0.22	0.24	0.23	0.23	0.31	0.31	0.71	0.68
	3000	0.51	0.56	0.54	0.58	0.6	0.66	0.9	1.05
	4000	0.96	1.06	1.16	1.31	0.94	1.05	2.43	2.56

4.2 Label setting algorithm

The experimentations for this algorithm were carried out with the same parameters than for the ranking approach. Table 6 summarises execution times obtained. Compared to the ranking approach, the label setting algorithm performs better when the number of objectives is small (2, 3 or 5). However, for a greater number of objectives (for example, 10 objectives), the label setting algorithm becomes less efficient than the ranking approach. A plausible explanation is that a lot of time is spent in dominance tests. Besides, the type of capacity drawn (v_1 or v_2) as well as the additive capacity chosen in the core (λ^* or ϕ) seem to have little impact on execution times, like in the previous approach.

Table 6
Label setting algorithm for ψ_v^w-P: execution times (in seconds).

		2 obj.		3 obj.		5 obj.		10 obj.	
	$\|V\|$	λ_i^*	ϕ_i	λ_i^*	ϕ_i	λ_i^*	ϕ_i	λ_i^*	ϕ_i
	1000	0.01	0.01	0.01	0.01	0.02	0.03	0.15	0.26
v_1	2000	0.02	0.02	0.02	0.03	0.04	0.08	0.44	0.77
	3000	0.04	0.04	0.04	0.06	0.07	0.14	0.58	1.04
	4000	0.06	0.07	0.07	0.12	0.1	0.21	0.73	2.13
	1000	0.01	0.01	0.01	0.01	0.04	0.05	0.84	0.84
v_2	2000	0.02	0.02	0.02	0.03	0.09	0.1	2.01	2.02
	3000	0.04	0.04	0.04	0.05	0.12	0.13	1.31	1.82
	4000	0.06	0.07	0.1	0.17	0.23	0.24	5.16	5.51

5 Experimental results for the Choquet-optimal spanning tree problem

Experimentations on Choquet-optimal spanning tree problems are performed on the same class of Choquet integrals as in the previous section (same construction of concave capacities v_1 and v_2, same convex disutility $w(x) = x^2$).

5.1 Ranking approach

The experimentations on the ψ_v^w-ST problem were performed on complete graphs (cliques). The components of cost vectors are randomly drawn between 1 and 100 on each edge. The number of objectives varies, here again, from 2 to 10, and the number of nodes from 10 to 30. For each kind of instances (depending on the number of nodes and on the number of objectives), 50 instances are randomly drawn. Table 7 summarises the average execution times of the ranking approach on these instances. Symbol "-" means that some executions could not terminate due to lack of memory space (more than 4GB required).

Table 7
Ranking approach for ψ_v^w-ST: execution times (in seconds).

		2 obj.		3 obj.		5 obj.		10 obj.	
	$\|V\|$	λ_i^*	ϕ_i	λ_i^*	ϕ_i	λ_i^*	ϕ_i	λ_i^*	ϕ_i
	10	0	0.02	0.01	0.04	0.02	0.19	0.7	2.16
	15	0	0.26	0.11	-	0.78	-	5.33	-
v_1	20	0.08	-	0.85	-	2.21	-	-	-
	25	0.22	-	1.11	-	-	-	-	-
	30	0.63	-	1.49	-	-	-	-	-
	10	0	0.01	0.01	0.03	0.06	0.57	4.92	5.27
	15	0	0.22	0.02	0.58	0.9	1.23	-	-
v_2	20	0.01	-	0.1	-	-	-	-	-
	25	0.19	-	0.56	-	-	-	-	-
	30	0.78	-	1.53	-	-	-	-	-

We can see that, on instances with about 30 nodes, an optimal solution is quickly determined. However, when the size of the instance grows, the memory space used by the ranking approach also quickly increases. Clearly, this will not be sufficient for large size instances. Besides, we can note that, contrary to what is observed for the Choquet-optimal path problem, the execution times are sensitive to the number of objectives.

5.2 Branch and bound procedure

The experimentations for this algorithm were carried out with the same parameters as for the ranking approach. Table 8 summarises the average execution times of the branch and bound procedure. Symbol ">1h" means that the average execution time is greater than one hour.

Table 8
Branch and bound approach for ψ_v^w-ST: execution times (in seconds).

		2 obj.		3 obj.		5 obj.		10 obj.	
	$\|V\|$	λ_i^*	ϕ_i	λ_i^*	ϕ_i	λ_i^*	ϕ_i	λ_i^*	ϕ_i
	10	0	0	0.01	0.03	0.06	0.29	2.21	6.2
	15	0.01	0.11	0.23	9.45	2.41	804	36.8	>1h
v_1	20	1.03	>1h	8.68	2726	31.4	>1h	>1h	>1h
	25	4.02	>1h	14.9	>1h	137.3	>1h	>1h	>1h
	30	13.4	>1h	60.7	>1h	>1h	>1h	>1h	>1h
	10	0	0	0.01	0.03	0.1	0.11	4.23	12
	15	0.01	0.16	0.1	9.63	2.36	3.04	1950	1987
v_2	20	0.48	40.13	0.86	63	72.1	>1h	>1h	>1h
	25	2.04	>1h	5.57	>1h	985.7	>1h	>1h	>1h
	30	5.11	>1h	48.6	>1h	3035	>1h	>1h	>1h

We can see that the average execution times are greater for the branch and bound procedure than for the ranking approach. However, the branch and bound procedure does not require as much memory space, and can therefore be applied to bigger instances. Nevertheless, as the size of the graph grows up, the average execution time significantly increases (beyond 30 minutes). The results also show that average execution times are sensitive to the number of objectives. Finally, here again, we observe that the execution time does not really depend on the type of capacity used (v_1 or v_2). The choice of an additive capacity inside the core (λ^* or ϕ) has a greater impact: the results obtained with λ^* are indeed significantly better than those obtained with ϕ.

5.3 Improving the quality of the lower bound

The lower bound introduced in Subsection 3.2 depends on a weighting vector λ defining an additive measure in the core of $\bar{v}$. We have proposed two possible weighting vectors, ϕ and λ^*, but there is no systematic argument to prefer one to the other and the best choice might really depend on the instance to deal with. One weakness of the bound we have introduced is that it relies on the choice of vector λ which is made a priori and *independently of the*

graph instance. However, all admissible weighting vectors λ do not provide the same value for the lower bound, and the best choice for these weights (i.e., providing the maximal lower bound) might depend on the trees under consideration. We want to select the most appropriate weighting vector λ for bounding the value of a ψ_v^w-optimal spanning tree at a given node η (corresponding to the set of spanning trees $\mathcal{T}(\eta)$) of the search tree. Let Y denote the image of $\mathcal{T}(\eta)$ in the multidimensional space (n criteria, scenarios or agents). Keeping in mind that w is strictly increasing, the optimal set of weights (i.e., providing the best lower bound according to Proposition 1) can be obtained by solving the following program:

$$\max_{\lambda \in \mathbb{R}^n} z(\lambda) = \min_{y \in Y} \sum_{i=1}^{n} \lambda_i y_i, \tag{6}$$

$$\text{s.t.} \sum_{i \in A} \lambda_i \leq v(A) \quad \forall A \subseteq N, \tag{7}$$

$$\sum_{i=1}^{n} \lambda_i = 1, \tag{8}$$

$$\lambda_i \geq 0 \quad \forall i = 1, \ldots, n. \tag{9}$$

Given that z is a concave piecewise linear function (since it is the lower envelope of a set of linear functions $\{\sum_{i=1}^{n} y_i \lambda_i : y \in Y\}$), we solve this program by using the SolvOpt library [23], which is an implementation of Shor's r-algorithm. This algorithm is indeed especially convenient for non-differentiable optimisation. To take into account constraints (7)-(9), the penalty function method is used, i.e. constraints are relaxed and one maximises $z(\lambda) - g(\lambda)$ instead of $z(\lambda)$, where g is a penalty function (that takes a positive value if a constraint is violated at point λ, and value 0 otherwise). Function g is defined as the sum of the residuals of the sets of violated constraints at point λ. More formally, $g(\lambda) = \sum_{A \subseteq N} \max\{r_A^\lambda, 0\}$ where $r_A^\lambda = \sum_{i \in A} \lambda - v(A)$.

Let us briefly explain the iterative optimisation process. For simplicity, we omit constraints (8) and (9) in the presentation. When performing a maximisation (as this is the case here), the basic principle of the algorithm is to build a sequence (λ^k) of points (a point is here a set of weights) by repeatedly making steps in the direction of a subgradient [1] $\bigtriangledown(z(\lambda^k) - g(\lambda^k)) = \bigtriangledown z(\lambda^k) - \bigtriangledown g(\lambda^k)$ at the current point (steepest ascent). However, unlike the standard subgradient method, every step is made in a dilated space, in a direction depending on the previous step.

More precisely, the minimisation of function $z(\lambda) - g(\lambda)$ at a given node η proceeds as follows. We chose an initial admissible weighting vector. Then, at iteration k of the procedure, we need to compute $\bigtriangledown z(\lambda^k) - \bigtriangledown g(\lambda^k)$. In this respect we first determine a minimum spanning tree T^k for valuation $f_0^k = \sum_{i=1}^{n} \lambda_i^k f_i$ to determine $z(\lambda^k)$. Hence the subgradient of z at point λ^k is nothing else but the vector $y^k = f(T^k)$ (the image of the minimum spanning tree computed at iteration k). Then, subgradient $\bigtriangledown g(\lambda^k)$ is equal to $(|C_1|, \ldots, |C_n|)$, where C_i denotes the set

[1] The subgradient plays here an analogous role to the one of the gradient in differentiable optimisation.

of violated constraints involving objective i. Then λ^{k+1} is computed by SolvOpt from λ^k and $\bigtriangledown z(\lambda^k) - \bigtriangledown g(\lambda^k)$ as indicated above. Since the set of feasible vectors is compact, the optimum is finite and the algorithm progressively converges to the optimum. Note that any set of weights that satisfies constraints (7)-(9) provides a lower bound. Hence, to save time, we can use only an approximation of the optimal solution and stop as soon as a given convergence threshold is achieved. The corresponding near-optimal value $z(\lambda^k)$ provides a bound for $\{\psi_v^w(y), y \in Y\}$.

Figure 7 presents a comparison between the average execution times obtained by the branch and bound procedure for the two kinds of lower bounds (using dynamic weights computed by Shor's r-algorithm or static weights computed *a priori* by Algorithm 1). Numerical tests were carried out on complete graphs of various sizes (from 10 to 30 nodes) using between 2 and 10 objectives, with concave capacities of type v_1 (see Section 4). We can see that the use of dynamic weights significantly improves the bounds and considerably decrease the computation times when the number of objectives is 5 or 10. When this number is smaller, the time spent at each node of the search-tree in Shor's r-algorithm is not compensated by the gain in the number of nodes. Consequently, the branch and bound procedure performs better with a static weights defined a priori.

Note that the use of dynamic weights to improve lower bounds could also be applied to the Choquet-optimal path problem. However, due to already satisfying numerical results for this problem, one can expect that the potential gain would not be as significant as for the minimum spanning tree problem.

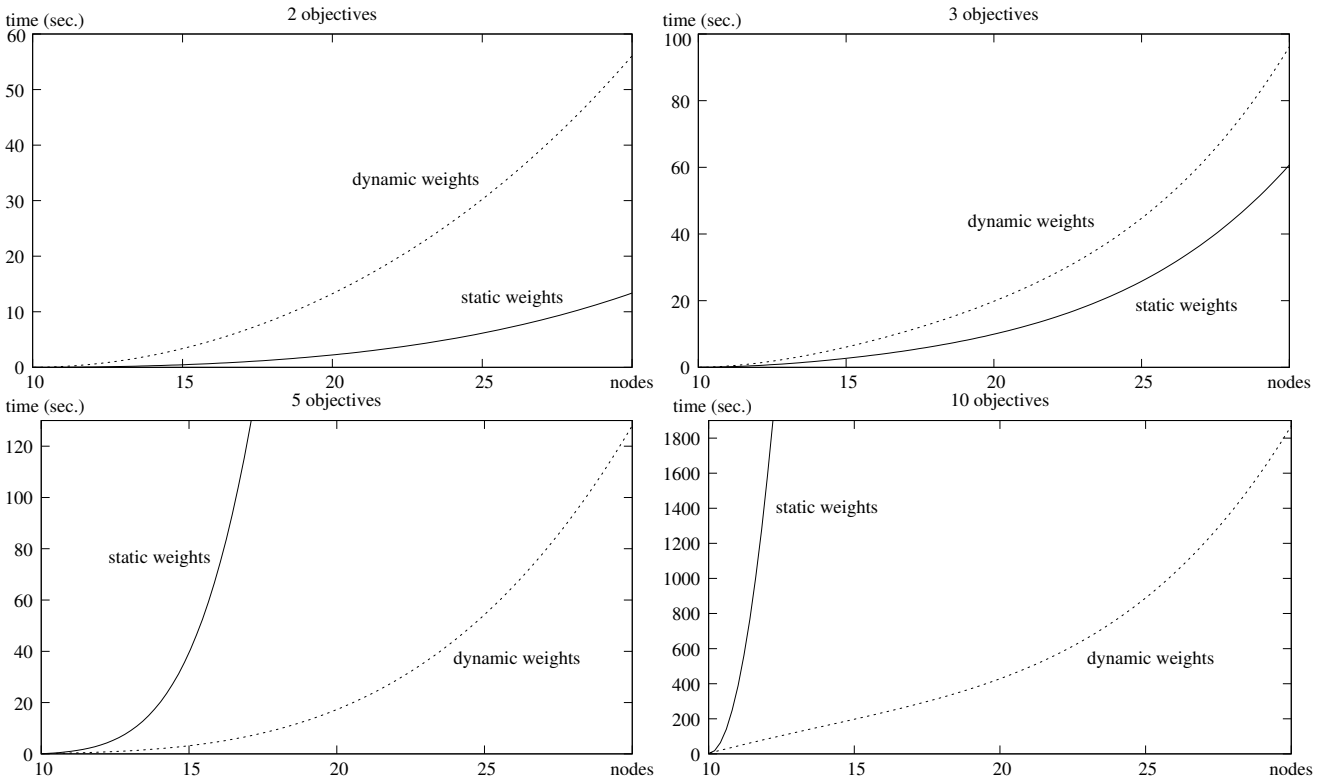

Fig. 7. Comparison between static and dynamic weights.

6 Conclusion

In this paper, we have studied Choquet-based optimisation when the capacity used to parameterize the Choquet integral is concave and the disutility is convex. We have presented various algorithms to compute Choquet-optimal solutions in multiobjective minimal spanning tree and shortest path problems. All of them rely on the use of a default approximation bounding Choquet expected disutilities.

The potential use of the bound we have introduced in the paper is not restricted to Choquet-optimal shortest path or spanning tree problems. As soon as a polynomial time algorithm is known for the standard version of a combinatorial problem, Proposition 1 provides a polynomially computable bound that can be used for the search of a Choquet-optimal solution in the multiobjective version. This bound can either be used in a branch and bound procedure or in a ranking approach, the latter requiring an efficient k-best solution procedure. For example, this is the case of matching problems [4], scheduling problems [2] and network flow problems [18]. The bound also holds for the OWA (Ordered Weighted Average) operator [44], its weighted extension WOWA [41], Yaari's model [43] and RDU (Rank Dependent Utility) [34], each of them being a particular case of Choquet expected utility.

For future works, it would be interesting to study a complementary type of bounds by relaxing the solution space rather than the objective function. For this purpose, the optimisation of a Choquet integral on a convex polyhedron is worth investigating. In this respect, a first step in this direction is the work by Ogryczak [30] on WOWA optimisation, that studies different LP reformulations of the problem. Another research direction would be to design polynomial time approximation algorithms (with performance guarantee) for ψ_v^w-ST and ψ_v^w-SP. Note that fully polynomial time approximation schemes (FPTAS's) already exist to determine the set of Pareto-optimal spanning trees and paths [32]. There also exist FPTAS's for the min-max versions of both problems (when the number of objective functions is bounded by a constant) [1]. It is likely that these approximation schemes can be extended to obtain FPTAS's for ψ_v^w-ST and ψ_v^w-P.

References

[1] H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

[2] P.J. Brucker and H.W. Hamacher. k-optimal solution sets for some polynomially solvable scheduling problems. *European Journal of Operational Research*, 41:194–202, 1989.

[3] A. Chateauneuf and J.M. Tallon. Diversification, convex preferences and non-empty core in the Choquet expected utlity model. *Economic Theory*, 19:509–523, 2002.

[4] C.K. Chegireddy and H.W. Hamacher. Algorithms for finding k-best perfect matchings. *Discrete Applied Mathematics*, 18:155–165, 1987.

[5] G. Choquet. Theory of capacities. *Annales de l'Institut Fourier*, 5:131–295, 1953.

[6] J. Edmonds. *Submodular functions, matroids and certain polyhedra*, pages 69–87. International conference on combinatorics. 1970.

[7] M. Ehrgott. *Multicriteria Optimization*, volume 491 of *Lecture Notes in Economics and Mathematical Systems*, pages 153–222. Springer Verlag, Berlin, 2000.

[8] M. Ehrgott and X. Gandibleux. *Multiple Criteria Optimization: State of the Art Surveys*. Kluwer Academic Publishers, 2002.

[9] V.A. Emelichev and V.A. Perepelitsa. Multiobjective problems on the spanning trees of a graph. *Soviet Math. Dokl.*, 37(1):114–117, 1988.

[10] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.

[11] S. Fujishige. *Submodular functions and Optimization*, volume Annals of discrete mathematics, 58. Elsevier Science Publishers, 2005.

[12] H. N. Gabow. Two algorithms for generating weighted spanning trees in order. *SIAM Journal on Computing*, 6(1):139–150, 1977.

[13] L. Galand and P. Perny. Search for compromise solutions in multiobjective state space graphs. In *17th European Conference on Artificial Intelligence*, pages 93–97, 2006.

[14] L. Galand and P. Perny. Search for Choquet-optimal paths under uncertainty. In *Proceedings of the 23rd conference on Uncertainty in Artificial Intelligence*, pages 125–132, Vancouver, Canada, 7 2007. AAAI Press.

[15] L. Galand and O. Spanjaard. OWA-based search in state space graphs with multiple cost functions. In *20th International Florida Artificial Intelligence Research Society Conference*, pages 86–91. AAAI Press, 2007.

[16] C. Gonzales, P. Perny, and S. Queiroz. GAI-networks: Optimization, ranking and collective choice in combinatorial domains. *Foundations of computing and decision sciences*, 32(4):3–24, 2008.

[17] M. Grabisch. The application of fuzzy integrals in multicriteria decision making. *European Journal of Operational Research*, 89(3):445–456, 1996.

[18] H.W. Hamacher. A note on k-best network flows. *Annals of Operations Research*, 57:65–72, 1995.

[19] H.W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52:209–230, 1994.

[20] P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multicriteria Decision Making*, 1980.

[21] J.-Y. Jaffray. On the maximum probability which is consistent with a convex capacity. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 3(1):27–34, 1995.

[22] V. M. Jiménez and A. Marzal. A lazy version of Eppstein's shortest paths algorithm. In *Experimental and Efficient Algorithms, Second International Workshop, WEA'03*, pages 179–190, 2003.

[23] F. Kappel and A.V. Kuntsevich. An implementation of Shor's r-algorithm. *Computational Optimization and Applications*, 15:193–205, 2000.

[24] N. Katoh, T. Ibaraki, and H. Mine. An algorithm for finding k minimum spanning trees. *SIAM Journal on Computing*, 10(2):247–255, 1981.

[25] P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*. Kluwer Academic Publisher, 1997.

[26] L. Lovász. Submodular functions and convexity. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming – The State of the Art.*, pages 235–257, 1983.

[27] E.Q.V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16:236–245, 1984.

[28] I. Murthy and S.S. Her. Solving min-max shortest-path problems on a network. *Naval Research Logistics*, 39:669–683, 1992.

[29] W. Ogryczak. Inequality measures and equitable approaches to location problems. *European Journal of Operational Research*, 122:374–391, 2000.

[30] W. Ogryczak and T. Sliwinski. On optimization of the importance weighted OWA aggregation of multiple criteria. In *Computational Science and Its Applications – ICCSA*, volume 4705 of *Lecture Notes in Computer Science*, pages 804–817, 2007.

[31] J. M. P. Paixão, E.Q.V. Martins, M. S. Rosa, and J. L. E. Santos. The determination of the path with minimum-cost norm value. *Networks*, 41(4):184–196, 2003.

[32] C.H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *IEEE Symposium on Foundations of Computer Science*, pages 86–92, 2000.

[33] P. Perny, O. Spanjaard, and L. X. Storme. A decision-theoretic approach to robust optimization in multivalued graphs. *Annals of Operations Research*, 147(1):317–341, 2006.

[34] J. Quiggin. *Generalized expected utility theory - The rank-dependent model*. Kluwer Academic, Dordrecht, 1993.

[35] D. Schmeidler. Integral representation without additivity. *Proceedings of the American Mathematical Society*, 97(2):255–261, 1986.

[36] D. Schmeidler. Subjective probability and expected utility without additivity. *Econmetrica*, 57:571–587, 1989.

[37] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[38] L.S. Shapley. Cores of convex games. *International Journal of Game Theory*, 1:11–22, 1971.

[39] R.E. Steuer and E.-U. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26:326–344, 1983.

[40] R.E. Tarjan. *Data structures and network algorithms.* Society for Industrial and Applied Mathematics, 1983.

[41] V. Torra. The weighted OWA operator. *International Journal of Intelligent Systems*, 12:153–166, 1997.

[42] P. Vincke. Robust solutions and methods in decision-aid. *Journal of Multicriteria Decision Analysis*, 8:181–187, 1999.

[43] M.E. Yaari. The dual theory of choice under risk. *Econometrica*, 55:95–115, 1987.

[44] R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. In *IEEE Trans. Systems, Man and Cybern.*, volume 18, pages 183–190, 1988.

[45] G. Yu and J. Yang. On the robust shortest path problem. *Computers and Operations Research*, 25(6):457–468, 1998.