Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines — UMR 8530 CNRS

Recherche Opérationnelle et Informatique

Note de Recherche

n° 11 / 2004 / LAMIH-ROI

Martins' algorithm revisited for multi-objective shortest path problems with a MaxMin cost function

Xavier Gandibleux, Frédéric Beugnies, Sabine Randriamasy

Submitted January 27, 2004
to 4OR, Quarterly Journal of the Belgian, French and Italian Operations Research Societies

UNIVERSITÉ DE VALENCIENNES ET DU HAINAUT-CAMBRÉSIS
LE MONT HOUY — F-59313 VALENCIENNES CEDEX 9 — FRANCE
Secrétariat : Mme Aureggi – phone: +33 (0)3 27 51 19 41
`www.univ-valenciennes.fr/LAMIH`

4OR manuscript No.
(will be inserted by the editor)

Martins' algorithm revisited for multi-objective shortest path problems with a MaxMin cost function

Xavier Gandibleux[1], **Frédéric Beugnies**[1], **Sabine Randriamasy**[2]

[1] LAMIH-ROI – UMR CNRS 8530, Université de Valenciennes
Campus "Le Mont Houy", F-59313 Valenciennes cedex 9 – France
e-mail: {Xavier.Gandibleux,Frederic.Beugnies}@univ-valenciennes.fr

[2] ALCATEL R&I - ISR project
Route de Nozay F-91460 Marcoussis – France
e-mail: Sabine.Randriamasy@alcatel.fr

January 27, 2004

Abstract This paper presents a straight extension of the label setting algorithm proposed by Martins in 1985 for a shortest path problem with multiple objectives. This extended version computes all the efficient paths from a given source vertex, to all the other vertices of the network. The algorithm can cope with problems in which the "cost values" associated with the network arcs are all positive, and it "optimizes" multiple objectives simultaneously. The proposed extension handles such objectives as the bottleneck function, as well as several linear functions. The main modications to Martins' algorithm for multi-objective shortest path problems touch upon the dominance test and the procedure for identifying efficient paths. The algorithmic features are described and a didactic example is provided to illustrate the working principle. The results of numerical experiments concerning the number of efficient solutions produced and the CPU time consumed for several configurations of objectives, on a set of randomly generated networks, are also provided.

Keywords: multi-objective combinatorial optimization – shortest path problems – labelling algorithms.

1 Introduction

Multi-Objective Combinatorial Optimization (MOCO) is a branch of multi-objective programming that touches many real-world problems in a broad range of areas, including finance, telecommunications, and transportation. Among MOCO problems, the Multi-Objective Shortest Path (MOSP) problem is well-known. As noted in the recent comprehensive survey of MOCO problems (Ehrgott and Gandibleux, 2002), MOSP appears to be one of the most intensively studied problems. Two types of objectives formulated by a linear function or a max-min function are usually considered in MOSP. These objectives correspond respectively to the "sum" problem and the "bottleneck" problem, which are denoted in the general case as (σ–S | μ–M), meaning that a situation concerns respectively σ objectives of the first type and μ objectives of the second. For example, $(2-\text{S} \mid 1-\text{M})$ denotes a situation with two linear objectives and one bottleneck. A brief look at the literature shows MOSP research has often concentrated on bi-objective situations and/or linear functions. Few papers deal with problems that have more than two objectives of different types, despite the importance of such problems in the real world.

There are two principal classes of algorithms for solving multi-objective shortest path problems with linear functions: labelling-based algorithms and ranking path-based algorithms. The former may be split into two main families: label setting algorithms (see Hansen, 1979; Martins, 1984) in which one label is set permanently at each iteration, and label correcting algorithms (see Brumbaugh-Smith and Shier, 1989; Mote et al., 1991; Skriver and Andersen, 2000) in which all the labels become permanent only in the last iteration. Ranking-based techniques, like those proposed in Climaco and Martins (1982), use a k-shortest path routine to solve the problem. In addition, specific network problems may be solved by using dynamic programming-based algorithms (see Henig, 1985; Kostreva and Wiecek, 1993; Sniedovich, 1988).

When all the objectives lead to sum problems (σ–S) only, the set of efficient paths can, under assumptions concerning the cost values, be computed by applying Martins' label setting algorithm, a multiple objective extension of the Dijkstra algorithm. The Martins algorithm is a direct modification of the classic shortest path algorithm in which the 'min' operator is replaced by a dominance test. This modification is valid since the Bellman principle of optimality can be applied to the multiple objective path case. Indeed, for the networks being considered, any subpath of an efficient path is an efficient subpath. Martins' algorithm ensures the computation of the maximal complete set of efficient paths from one vertex to all the other vertices of a network.

Unless mentioned otherwise, S-type objectives seek to minimize (cost, delay) and M-type ones seek to maximize (quality, bandwidth) in the continuation. In this paper, we propose an extension of Martins' algorithm for solving ($\sigma - \mathrm{S} \mid 1 - \mathrm{M}$)-class multi-objective shortest path problems. Several real-world contexts are addressed by this ($\sigma - \mathrm{S} \mid 1 - \mathrm{M}$) problem. For example, in the world of computer networks, Internet traffic routing could be enhanced if based on a multi-objective routing procedure which would prevent network congestion. Multi-objective shortest paths between one router and all the others routers of the network must be computed in real-time, by simultaneously optimizing (σ–S | 1–M) (Randriamasy et al., 2002; Randriamasy and Gandibleux, 2003). The routing operation includes a procedure for computing the multi-objective shortest paths. In the context of vehicle routing, selecting a route in a transportation network, according to the objectives of capacity, distance or time, is the same type of problem (Moore et al., 1978).

This paper is organized as follows. Section 2 provides the mathematical background, i.e., definitions and notations. Section 3 presents an algorithmic description of the Martins label setting algorithm. Section 4 describes our extension of this algorithm for ($\sigma - \mathrm{S} \mid 1 - \mathrm{M}$) shortest path problems. In section 5, the results of numerical experiments on set of randomly generated networks are reported and discussed. Finally, section 6 presents our conclusions and suggests several directions for future research.

2 Mathematical background

2.1 Fundamental definitions

Let us consider the following notation. A *network* is defined as a directed and connected *graph* $G = (V, A)$, where $V = \{1, \dots, n\}$ is the set of *vertices* with cardinality $\mid V \mid= n$ and $A = \{(i_1, j_1), \dots, (i_m, j_m)\}$ is the set of *arcs* with cardinality $\mid A \mid= m$. The directed arc linking vertices i and j is denoted by (i, j), and the vector $(c^1(i, j), \dots, c^K(i, j))$ represents the "cost values" associated with the arc (i, j). C refers to the *cost matrix* for all arcs (i, j) in G. In the set V, we identify a *source vertex* s and a *sink vertex* t. A *path* r from s to t in G is a sequence of arcs and vertices from s to t, where the tail vertex of a given arc coincides with the head vertex of the next arc in the path; $R_{s,t}$ denotes the set of all paths from s to t in G, and $R_{s,\bullet}$, the set of all the paths from s to all others vertices $V \setminus \{s\}$ in G. Let $z^p(r)$ denote the value of a path r with respect to the criterion p, for $p = 1, \dots, K$, with $K = \sigma + \mu$, the number of objectives. Hence, the vector $z(r) = (z^1(r), \dots, z^K(r))$ represents the *evaluation* or *performance vector* of path $r \in R_{s,t}$. The formulation of the objectives in question is $z^p(r) = \sum_{(i,j)\in r} c^p(i, j)$ for the linear function, and $z^p(r) = \min_{(i,j)\in r} c^p(i, j)$ for the max-min function. These functions correspond respectively to the (1–S) problem $\min \sum_{(i,j)\in r} c^p(i, j)$, and the (1–M) problem $\max \min_{(i,j)\in r} c^p(i, j)$. When all the objectives are minimized, a path r_e is said to be *efficient* if and only if, there is no path r in $R_{s,t}$, such that $z^p(r) \leq z^p(r_e)$, for all $p = 1, \ldots, K$, with at least one strict inequality. $z(r_e)$ is a non-dominated point or vector. A path r_{we} is said to be *weakly efficient* if and only if, there is no path r in $R_{s,t}$, such that $z(r) < z(r_{we})$. $z(r_{we})$ is a weakly non-dominated point or vector.

2.2 Multi-objective shortest paths

The multi-objective shortest path problem is a well-known NP-hard (Serafini, 1986) MOCO problem. The version considered in this paper can be summarized as:

$$\text{“}\min_{r \in R_{s,t}}\text{”}(z^1(r), \ldots, z^K(r)) \quad \forall t \in V \setminus \{s\} \tag{MOSP}$$

Let $E(R_{s,\bullet})$ denote the *maximal complete set* of efficient paths in $R_{s,\bullet}$ for a given source vertex s, i.e. the set of all paths of $R_{s,\bullet}$ in which each path corresponds to a non-dominated point. Several distinct efficient paths r_1^*, r_2^*, r_3^* can correspond to the same non-dominated point $z(r_1^*) = z(r_2^*) = z(r_3^*)$ in the objective space. The paths r_1^*, r_2^*, r_3^* are said to be *equivalent* in the objective space. The *minimal complete set* of efficient paths is a subset of $E(R_{s,\bullet})$ that contains no equivalent solutions, and for any $r \in E(R_{s,\bullet})$, there exists r' in the minimal complete set, such that r and r' are equivalent.

Hansen (1979) studied ten shortest path problems with two objectives. In this paper, the author reported the following important results :

- Minimizing or maximizing a $(1-\mathrm{S})$ problem is equivalent. In the same way, a $(1-\mathrm{M})$ problem can be considered either as max min or min max.
- For the $(2-\mathrm{S})$ problem, the number of solutions $|R_{s,t}|$ is equal to $2^{|A|}$ for some specific networks.
- An algorithm yielding to a minimal complete set of efficient paths $E(R_{s,t})$ for a given t for the $(1-\mathrm{S} \mid 1-\mathrm{M})$ problem in $\mathrm{O}(m^2 \log n)$ operations.

In terms of complexity, the enumeration of all efficient paths in a multi-objective shortest path problem may not be tractable in polynomial time. However, the example considered in Hansen makes use of exponential arc values. In practice, arc values are far from that specific case, and the number of efficient paths is generally not exponential. Section 5 provides the experimental results concerning the number of efficient paths collected for randomly generated networks.

Martins (1984) proposed a label setting algorithm for computing the maximal complete set of efficient paths when all the cost values are non negative ($c^p(i,j) \geq 0$ for all $(i,j) \in A$, and $p = 1...K$) with at least one strict inequality hold for some value of p. It is a generalisation for $(\sigma-\mathrm{S})$ objectives of the algorithm presented by Hansen in 1979.

3 Martins label setting algorithm

The main idea of the Martins' algorithm is quite simple. At each iteration, for each vertex, there are two different sets of labels: *permanent* labels and *temporary* labels. The algorithm selects the minimum lexicographic label from all the sets of temporary labels, converts it to a permanent label, and propagates the information contained in this label to all the temporary labels of its successors. The procedure stops when there are no more temporary labels. A label l_i can be represented as follows:

$$l_i = [\ z^1, \ldots, z^K,\ j,\ h\ ]$$

where $(z^1, \ldots, z^K)$ is the performance vector extracted using the operator $\texttt{perf}(l_i)$; j is the adjacent vertex from which it was possible to label the vertex i; and h is the position of the label in the list of labels on vertex j. The Martins algorithm is described in algorithm 1.

The following example illustrates the algorithm. Let us examine the network in Figure 1 with the goal of computing all the shortest paths for $(3-\mathrm{S})$ from vertex $s = 1$ to vertices $t = \{2, 3, 4\}$, with all the objectives being minimized.

1. The temporary label $[0, 0, 0, \perp, \perp]$ is assigned to vertex $s = 1$. This first label is selected and made permanent. Vertex 1's successors are labelled, yielding two temporary labels : $[5, 5, 5, 1, 1]$ on vertex 2 and $[3, 4, 6, 1, 1]$ on vertex 3.
2. Of the temporary labels, $[3, 4, 6, 1, 1]$ is lexicographically the smallest. This label is selected and becomes permanent. Vertex 3's successors are in turn labelled, yielding two new temporary labels : $[5, 5, 16, 3, 1]$ on vertex 2 and $[6, 6, 10, 3, 1]$ on vertex 4. The new temporary label on vertex 2 is dominated by the previous one, $[5, 5, 5, 1, 1]$, and is consequently deleted.
3. Of the now existing temporary labels, $[5, 5, 5, 1, 1]$ is lexicographically the smallest. Once selected, this label becomes permanent. Vertex 2's successors are labelled, again yielding two temporary labels : $[6, 7, 15, 2, 1]$ on vertex 3 and $[6, 6, 10, 2, 1]$ on vertex 4. The new temporary label on vertex 3 is dominated by $[3, 4, 6, 1, 1]$, and is consequently is deleted.
4. The temporary labels $[6, 6, 10, 2, 1]$ and $[6, 6, 10, 3, 1]$ are equally lexicographically. If $[6, 6, 10, 2, 1]$ is selected and becomes permanent, vertex 4 has no successor, and thus no new temporary label is created.
5. The single temporary label remaining in the list, $[6, 6, 10, 3, 1]$, is in turn selected and becomes permanent. Because vertex 4 has no successor, no new temporary label is created. The list of temporary labels now being empty, the labelling phase is concluded.

Algorithm 1 Martins' label setting algorithm for MOSP problem

Require: $G = (V, A)$, and C, the cost matrix for all arcs $(i, j) \in A$
Ensure: All efficient paths from s to all vertices $i \in V \backslash \{s\}$

l_i : is a label of vertex i
lt_i : is the entire list of temporary labels of vertex i
lp_i : is the entire list of permanent labels of vertex i
$z^p_{q,h}$: is the p^{th} performance of a permanent label of vertex q in position h
Δ : is the dominance relation (if $z \Delta z'$ then z is dominated by z')

--| Initialization
$lt_i, lp_i \leftarrow \emptyset, \forall i \in V$
$lt_s \leftarrow \{\ [\ 0, \ldots, 0,\ \perp,\ \perp\ ]\ \}$

--| Iteration
while $(\bigcup_{i \in V} lt_i \neq \emptyset)$ **do**
  --| Find the minimum lexicographic label in $lt_i, \forall i \in V$
  $l_q \leftarrow$ min lex$\{\bigcup_{i \in V} lt_i\}$
  --| Move the selected label from the 'temporary' list to the 'permanent' list
  $lt_q \leftarrow lt_q \setminus \{l_q\}$; $lp_q \leftarrow lp_q \cup \{l_q\}$
  --| Store the position of label l_q from list lp_q
  $h \leftarrow$ card(lp_q)
  --| Label all the successors of q
  for all $j \in V \mid (q, j) \in A$ **do**
    --| Compute l_j, the current label of vertex j
    $l_j \leftarrow [\ z^1_{q,h} + c^1(q, j), \ldots, z^K_{q,h} + c^K(q, j),\ q,\ h\ ]$
    --| Verify that there is no performance of vertex j labels dominating $\texttt{perf}(l_j)$)
    if $(\nexists l'_j \in \{lt_j \cup lp_j\} \mid \texttt{perf}(l_j)\, \Delta\, \texttt{perf}(l'_j))$ **then**
      --| Store the label l_j of vertex j as temporary
      $lt_j \leftarrow lt_j \cup \{l_j\}$
      --| Delete all temporary labels of vertex j dominated by l_j
      $lt_j \leftarrow lt_j \setminus \{l'_j \in lt_j \mid \texttt{perf}(l'_j)\, \Delta\, \texttt{perf}(l_j)\}$
    end if
  end for
end while

Each permanent label corresponds to a unique efficient path. To determine any of these paths, choose one permanent label on a vertex q and extract the values corresponding to j and h for this label. So, j is the vertex just before q in the efficient path. To determine the label on vertex j that has produced this current path, the value of h is needed. This value indicates the hth label on vertex j that has produced the current path. By moving backwards, the first vertex of the path (s) will be reached. Table 1 summarizes all the efficient paths computed from $s = 1$. A valid proof concerning this algorithm is described in Ehrgott (2000).

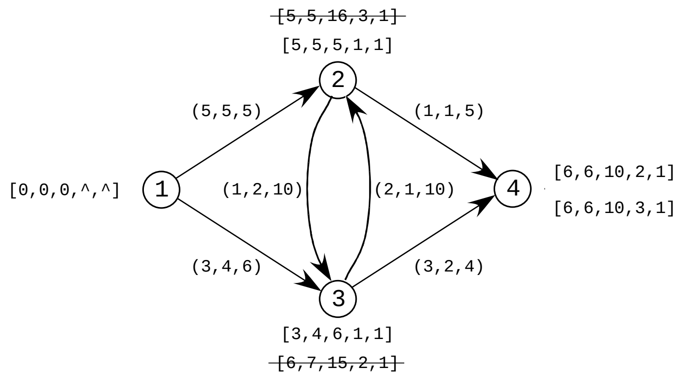

Fig. 1. Example for (3−S). The deleted labels are crossed out.

from $s = 1$ to	paths $r_e \in E(R_{s,\bullet})$ are	performances $z(r_e)$
$t = 2$	$1 \rightarrow 2$	(5, 5, 5)
$t = 3$	$1 \rightarrow 3$	(3, 4, 6)
$t = 4$	$1 \rightarrow 2 \rightarrow 4$	(6, 6, 10)
	$1 \rightarrow 3 \rightarrow 4$	(6, 6, 10)

Table 1. Shortest paths concerning the (3−S) example

Real world problems are often complex, and the objectives that must be considered in the optimization problem can depend on dynamic factors. For example, in a communication network, depending on the circumstances and management policy, computing the optimal path may or may not involve S-type objectives, such as transmission time or administrative cost. However, another objective is fundamental: the available bandwidth of the links in the network, which is a M-type objective. Given these parameters, the optimization procedure must be both flexible in term of the number of S-type objectives and capable of optimizing both S- and M-type of objectives simultaneously. In addition, the minimal complete set of efficient paths is not sufficient; all equivalent paths must be available as backup, ready to be used if the initially selected path cannot be implemented due to the sudden breakdown of a router in the network, for example.

Martins' algorithm both computes the maximal complete set of efficient paths and is able to deal with several S-type objectives. These are the characteristics that motivated us to extend this algorithm to simultaneously optimise S-type and M-type objectives. The revised version presented in the next section deals with (σ–S | 1–M) in which $\sigma \geq 1$ and $\mu = 1$, and computes the maximal complete set of efficient paths $\{r_e \mid r_e \in R_{s,\bullet}\}$.

4 The revised Martins algorithm for $(\sigma - \mathrm{S} \,|\, 1 - \mathrm{M})$

The main change in our version of the Martins algorithm concerns the dominance test Δ for ensuring that the maximal complete set of efficient paths is computed for $(\sigma - \mathrm{S} \,|\, 1 - \mathrm{M})$. For reasons of simplicity, all objectives are minimized in the following description and by abuse of notation, the indices noted as $1...\sigma$ refer to the S-type objectives and μ refers to the M-type objective. Let us focus on a vertex q connected to a successor vertex j (Figure 2).

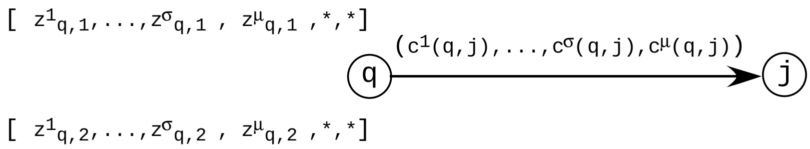

Fig. 2. View of two labels on a vertex

The costs of the arc (q, j) are given by $(c^1(q, j), \ldots, c^\sigma(q, j), c^\mu(q, j))$. Consider two temporary labels $[\, z^1_{q,h}, \ldots, z^\sigma_{q,h}, z^\mu_{q,h}, \ *, \ * \,]$ where $h = 1, 2$ on the vertex q. When $z^p_{q,1} = z^p_{q,2}$ $\forall p = 1, ..., \sigma$ and $z^\mu_{q,1} < z^\mu_{q,2}$ (resp. $z^\mu_{q,1} > z^\mu_{q,2}$), label 2 (resp. 1) is weakly non-dominated by label 1 (resp. 2). According to the dominance test in Martins algorithm, label 2 (resp. 1) must be deleted. Now, consider these two labels on the vertex j according to the value of $c^\mu(q, j)$. If $c^\mu(q, j) \geq \max(z^\mu_{q,1}, z^\mu_{q,2})$, then the two labels become equivalent on vertex j and thus, label 2 must be kept; otherwise, label 2 (resp. 1) remains weakly non-dominated on vertex j. Thus, the dominance test used in the original version of the algorithm cannot find all the efficient paths for this configuration of objectives. To do so, it must be modified as follows :

A) No labels are deleted when the two labels are non-dominated :

$$\exists p, p' \in \{1, ..., \sigma\} \cup \{\mu\}, \ p \neq p', \text{ such that } z^p_{q,1} \ < z^p_{q,2} \text{ and } z^{p'}_{q,1} > z^{p'}_{q,2}$$

or the two labels are equivalent :

$$z^p_{q,1} = z^p_{q,2} \ \forall p \in \{1, ..., \sigma\} \cup \{\mu\}$$

B) Label 2 (resp. 1) is deleted when label 2 (resp. 1) is dominated by the label 1 (resp. 2) :

$$z^p_{q,1} < z^p_{q,2} \text{ (resp. } z^p_{q,1} > z^p_{q,2}) \ \forall p \in \{1, ..., \sigma\} \cup \{\mu\}$$

C) Label 2 (resp. 1) is weakly non-dominated by label 1 (resp. 2), i.e. $z^p_{q,1} \leq z^p_{q,2}$ (resp. $z^p_{q,1} \geq z^p_{q,2}$) $\forall p \in \{1, ..., \sigma\} \cup \{\mu\}$ with at least one strict inequality :

C1) if there is at least one $p' \in \{1, ..., \sigma\}$, such that $z^{p'}_{q,1} < z^{p'}_{q,2}$ (resp. $z^{p'}_{q,1} > z^{p'}_{q,2}$), then label 2 (resp. 1) is deleted because, under the assumptions (non-negative costs), these two labels cannot become equivalent in the next iterations.

C2) otherwise $z^{p}_{q,1} = z^{p}_{q,2}$ $\forall p = 1, ..., \sigma$ and then neither label is deleted because, according the value of $c^{\mu}(q, j)$, these two labels can become equivalent on vertex j.

Because some weakly non-dominated labels have been made permanent, not every permanent label corresponds to an efficient path when the labelling phase is completed. But these weakly non-dominated labels cannot be deleted because they participate in the determination of the efficient paths. Such a label can correpond to an intermediate vertex for an efficient path; however, they cannot be a sink vertex of an efficient path because, by definition, there is another label on this vertex with a better performance. Weakly non-dominated labels (that have been crossed out twice in Figure 3) are thus hidden in the list of permanent labels that will be used for the construction procedure as a sink vertex of an efficient path. The others remain visible and correspond to individual efficient paths. Other than this particularity, a path is determined in the same way in our revised algorithm as in Martins algorithm (see section 3).

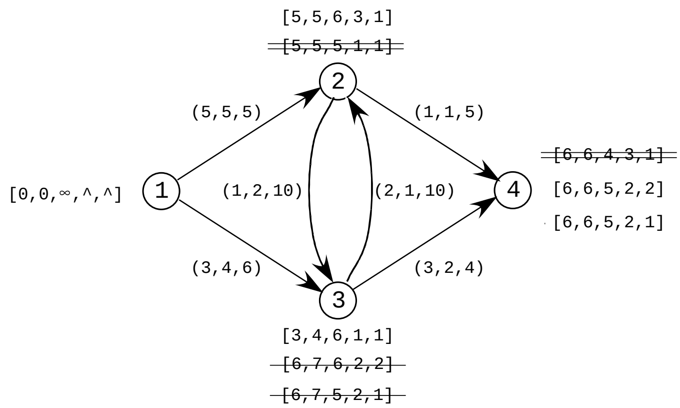

Fig. 3. Example of $(2-\mathrm{S}\,|\,1-\mathrm{M})$. The deleted labels are crossed out. The labels that have been crossed out twice correspond to the hidden labels for the determination of a sink vertex of an efficient paths

In order to illustrate the revised dominance test, the example in Figure 3 is used to compute the shortest paths for $(2-\mathrm{S}\,|\,1-\mathrm{M})$ from source vertex

$s = 1$ to sink vertices $t = \{2, 3, 4\}$. The two first objectives are minimized; the last one is maximized.

1. The temporary label $[0, 0, \infty, \perp, \perp]$, assigned to vertex $s = 1$, is selected and becomes permanent. Two temporary labels are computed : $[5, 5, 5, 1, 1]$ for vertex 2 and $[3, 4, 6, 1, 1]$ for vertex 3.
2. Of these temporary labels, the smallest lexicographically is $[3, 4, 6, 1, 1]$. It is selected and becomes permanent. The vertex 3 successors are labelled, yielding temporary labels $[5, 5, 6, 3, 1]$ for vertex 2 and $[6, 6, 4, 3, 1]$ for vertex 4. Label $[5, 5, 5, 1, 1]$ on vertex 2 is weakly non-dominated by $[5, 5, 6, 3, 1]$, producing a situation in which performances on the sum objectives are equal. Therefore, this label is not deleted.
3. $[5, 5, 6, 3, 1]$ is the next temporary label selected. In its turn, it becomes permanent. The vertex 2 successors are labelled, yielding temporary labels : $[6, 7, 6, 2, 2]$ for vertex 3 and $[6, 6, 5, 2, 2]$ for vertex 4. The new temporary label for vertex 3 is weakly non-dominated by permanent label $[3, 4, 6, 1, 1]$. This situation corresponds to the case C1 presented earlier. Consequently, label $[6, 7, 6, 2, 2]$ is deleted.
4. In the next iteration, $[5, 5, 5, 1, 1]$ is the temporary label selected and made permanent. The vertex 2 successors are labelled, yielding two temporary labels : $[6, 7, 5, 2, 1]$ on vertex 3 and $[6, 6, 5, 2, 1]$ on vertex 4. The new temporary label on vertex 3 is again weakly non-dominated by $[3, 4, 6, 1, 1]$, and consequently, is deleted.
5. $[6, 6, 5, 2, 1]$ and $[6, 6, 5, 2, 2]$ are lexicographically the smallest temporary labels. Assuming that $[6, 6, 5, 2, 1]$ is selected and becomes permanent, vertex 4 has no successor, and thus no new temporary label is created.
6. The list now contains only a single temporary label, $[6, 6, 5, 2, 2]$. This label is selected and becomes permanent. Again, vertex 4 has no successor, and no new temporary label is created. The list of temporary labels now being empty, the labelling phase is concluded.

from $s = 1$ to	paths $r_e \in E(R_{s,\bullet})$ are	performances $z(r_e)$
$t = 2$	$1 \to 2$	(5, 5, 6)
$t = 3$	$1 \to 3$	(3, 4, 6)
$t = 4$	$1 \to 2 \to 4$	(6, 6, 5)
	$1 \to 3 \to 2 \to 4$	(6, 6, 5)

Table 2. Shortest paths in the (2−S|1−M) example

Table 2 presents all the efficient paths computed. To summarize, our revision touches three aspects of the original algorithm. First, it replaces the value 0 corresponding to a min−max objective in the initial label with

∞ for a max−min objective. Second, it extends the dominance test on labels, and third, it disables the weakly non-dominated permanent labels for the determination of efficient paths at the end of the labelling phase. The following section presents the results of several numerical experiments. The aim of these experiments was to provide feedback concerning the number of efficient solutions observed and the CPUt required in practice, for selected configurations of randomly generated networks.

5 Numerical experiments

5.1 Experimental conditions

In these experiments, the networks were generated for a given number of vertices and a given density of arcs. In all networks, each node can be reached from the source node. The network topology was built as follows : first a rooted tree was elaborated, and then, arcs were added randomly to the tree in order to reach the desired density (Figure 4).

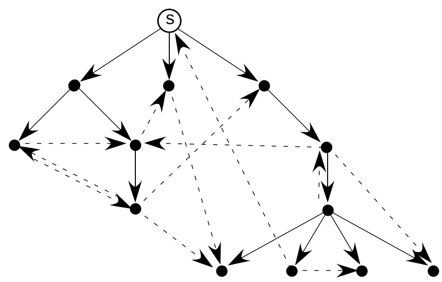

Fig. 4. Example of a topology. Plain arrows were included in the oriented tree. Dashed arrows are added randomly.

Twenty-seven parameter configurations were selected for generating the random networks : (i) the number of vertices : 50, 100, and 200 vertices; (ii) the density of the network : 5%, 10%, and 20%; (iii) the cost values $c^p(i,j)$, for all $(i,j) \in A$:

- range 1: [1 , 100] for p=1,2,3
- range 2: [1 , 1000] for p=1,2,3
- range 3: [1 , 255] for p=1; [1 , 50000] for p=2 ; [1 , 1000000] for p=3

Ten numerical instances were generated for each configuration. Four configurations of objectives were considered for computing the efficient paths : (2−S), (1−S | 1−M), (3−S), and (2−S | 1−M), with all S-type objectives minimized and the M-type maximized. For each configuration of objectives, the maximal complete set of the shortest paths from node $s = 1$ to all other network vertices was computed. The experiments yielded results concerning : (i) the CPUt (minimum/avg/maximum), (ii) the number of efficient paths observed for each network configuration (avg), (iii) the number of efficient paths observed for each network configuration (minimum avg), given the node with the minimum number of solutions, and (iv) the number of efficient paths observed for each network configuration (maximum avg), given the node with the maximum number of solutions. Results for the last two categories concerned extreme cases.

The computer used for the experiments was a laptop equipped with a PowerPC G4 1GHz processor, 512 Mb of RAM memory installed, running under the MacOSX 10.3.2 operating system. The algorithms were implemented in the language C. The binary code was obtained using the gcc-3.3 compiler without the optimizer option.

5.2 Results and discussion

Figures 5 and 6 present, respectively, the number of efficient paths and the CPU time observed for the (2–S | 1–M) configuration. We found no significant difference between the three value range sets used in the experiment. Our algorithm does not seem to be sensitive to the ranges.

The number of efficient paths is larger for higher density and network size values, than the equivalent networks with lower values for one of these parameters. In general, the average value of the minimum number of efficient paths is low. Thus, there are apparently some vertices in the network where few efficient backup paths are available from s to these vertices. On the other hand, the maximum number is generally high, almost twice the average value. Thus, there are also several vertices with a huge number of backup paths. Consequently, it is not surprising to observe that more CPU time is needed in large networks with a high density of arcs.

The expected use of this algorithm in real world problems related to computer communication, involves a target configuration for the network of 100 vertices with a density of less than 20 %, and typical cost values in "range 3". To be used in this environment, the algorithm must be able to produce solutions in less than one second, using a conventional computer. Given that the most pessimistic measurements are far under this temporal limit, the results shown in Figure 6 appear very promising.

Figures 7 and 8 respectively synthetise the average values of the efficient paths and the CPU time observed for the four configurations of objectives. Remember that all S-type objectives were minimized and the M-type objective was maximized in these experiments. For the configurations with two or three objectives, plus one M-type objective enabled in the labelling procedure, the number of efficient paths is very often greater than the configurations with only S-type objectives. In examining the 1080 resolutions performed, it appears that this is true for 96,30 % of the configurations.

Compared to the (2−S | 1−M) configuration, all the other configurations need less CPU time. Thus, the algorithm satisfies the ressource constraints of the real world problem, for all of the configurations in the experiment.

6 Conclusion

We have presented a revised version of the Martins algorithm for multi-objective shortest path problems with (σ−S | 1−M) objectives. Our revision concerns the initial label, the dominance test for labels during the labelling procedure, and the extraction procedure for the labelled efficient paths. Numerical experiments allowed us to verify that the number of efficient paths generated is not exponential in practice, even when simultaneously optimizing these S- and M- type objectives. In addition, the computational effort required by the revised algorithm is comparable to Martins version although more labels are handled in the revised version.

By combining two types of objectives, this revised version provides a flexible algorithmic solution for dealing with real world problems in which the configuration of objectives to be considered depends on external factors. It can be used in a real-time environment, such as the computer communication problem mentioned in this paper. Finally, although this paper examined the configuration with only one M-type objective, our revised version of the Martins algorithm is capable of handling more than one of these objectives.

Acknowledgements The authors would like to thank José Figueira, guest researcher at the University of Valenciennes for a month in 2001, for many fruitful discussions concerning shortest path problems with multiple objectives.

References

J. Brumbaugh-Smith and D. Shier. An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operational Research*, 43(2):216–224, 1989.

J.C.M. Climaco and E.Q.V. Martins. A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11:399–404, 1982.

M. Ehrgott. *Multicriteria Optimization*. Volume 491 of *Lecture Notes in Economics and Mathematical Systems* Springer, Berlin, 2000.

M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization. In Matthias Ehrgott and Xavier Gandibleux, editors, *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Survey*, volume Volume 52 of *International Series in Operations Research and Management Science*, chapter 8, pages 369–444. Kluwer Academic Publishers, 2002.

P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making Theory and Application*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 109–127. Springer Verlag, Berlin, 1979.

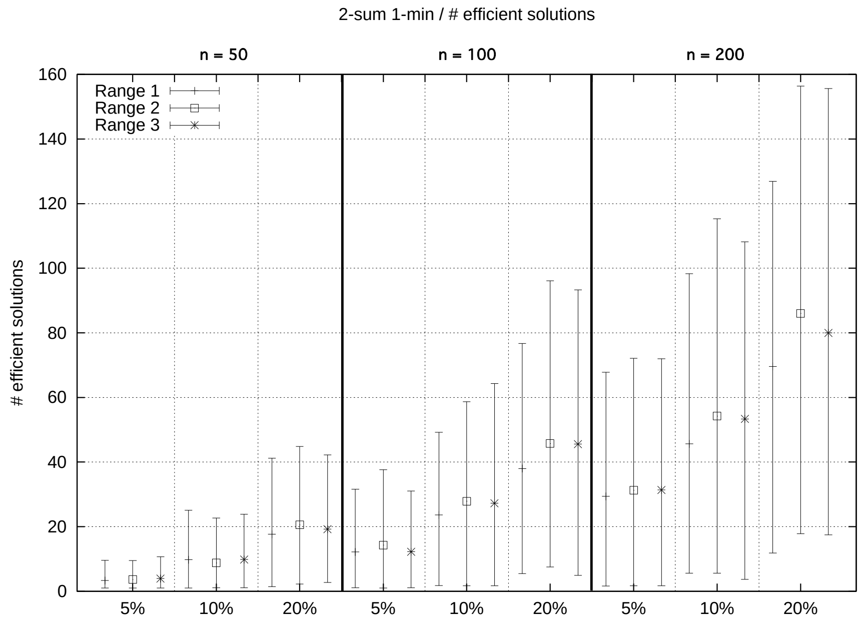

Fig. 5. Number of efficient paths observed for the (2−S|1−M) configuration.

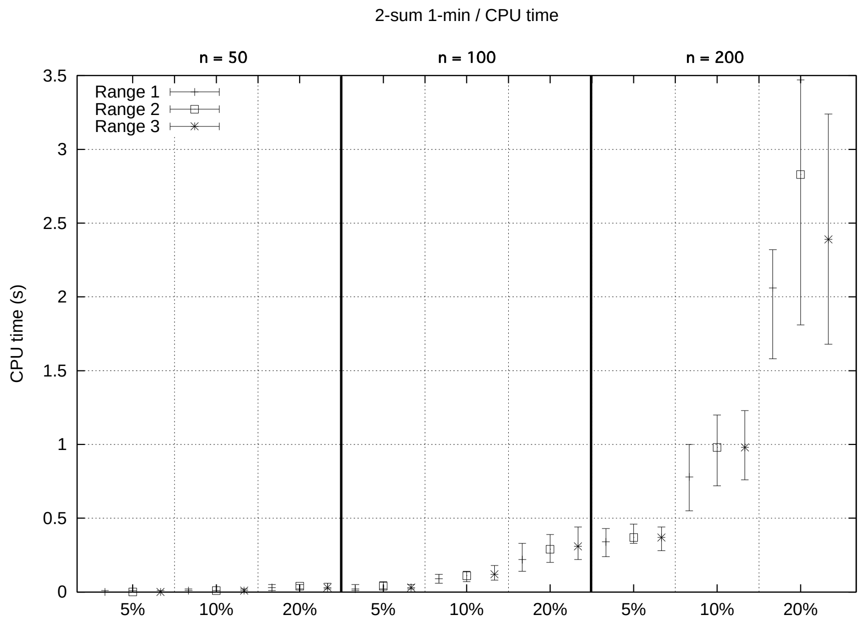

Fig. 6. CPUtime comsumed for the (2−S|1−M) configuration.

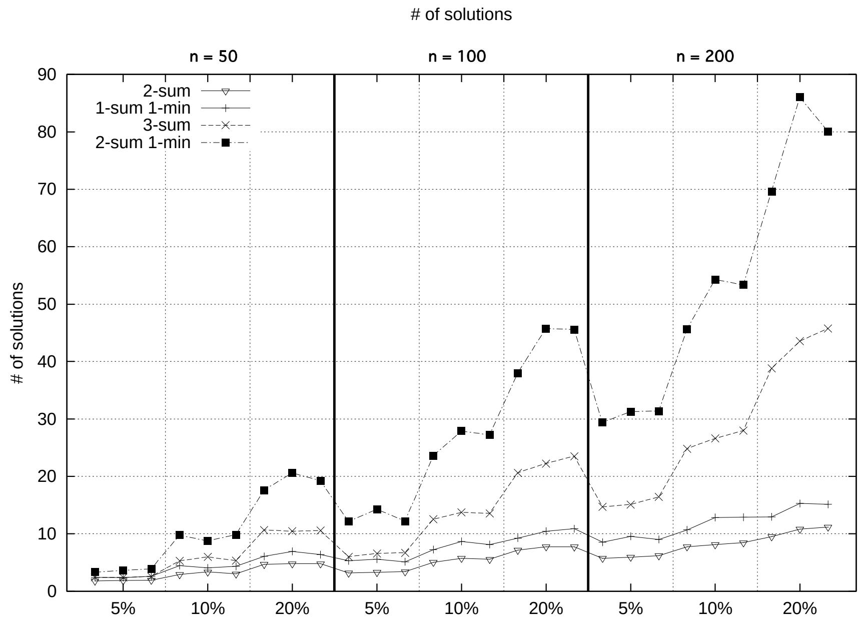

Fig. 7. Average number of efficient paths observed for all configurations.

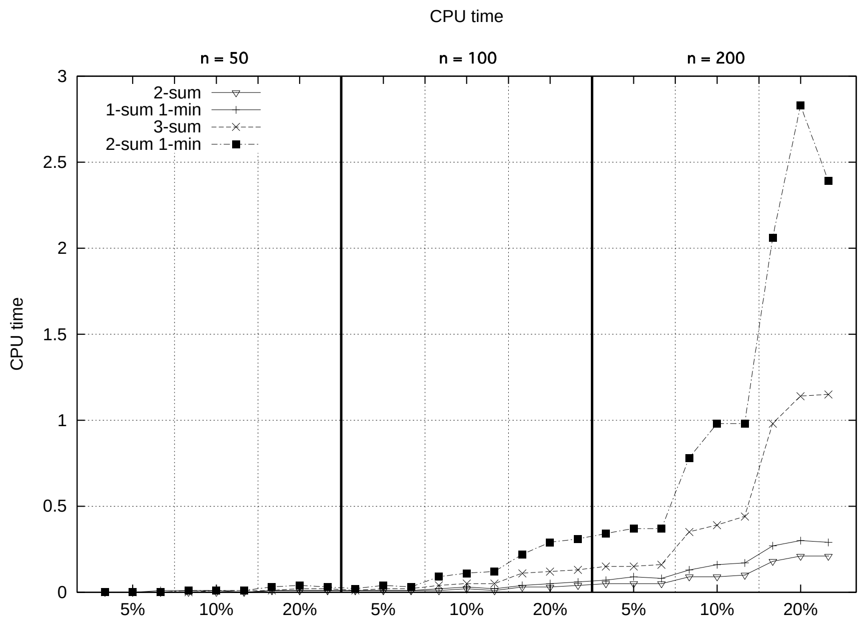

Fig. 8. Average CPUtime for computing all solutions for all configurations.

M.I. Henig. The shortest path problem with two objective functions. *European Journal of Operational Research*, 25:281–291, 1985.

M.M. Kostreva and M.M. Wiecek. Time dependency in multiple objective dynamic programming. *Journal of Mathematical Analysis and Applications*, 173(1):289–307, 1993.

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

L.J. Moore, B.W. Taylor, and S.M. Lee. Analysis of a transshipment problem with multiple conflicting objectives. *Computers and Operations Research*, 5:39–46, 1978.

J. Mote, I. Murthy and DL. Olson. A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53:81–92, 1991.

S. Randriamasy, X. Gandibleux, J. Figueira, and Ph. Thomin. Fiche brevet n°03291744.5-2416 intitulée 'Dispositif et procédé de détermination de chemins de routage dans un réseau de communications, en présence dattributs de sélection', Fiche déposée le 15 Juillet 2002.

S. Randriamasy and X. Gandibleux. Routage multiobjectif dans les réseaux IP. *ROADEF'03*, 26-28 février 2003, Avignon, France.

P. Serafini. Some considerations about computational complexity for multi objective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent advances and historical development of vector optimization*, volume 294 of *Lecture Notes in Economics and Mathematical Systems*. Springer Verlag, Berlin, 1986.

A.J.V. Skriver and K.A. Andersen. A label correcting approach for solving bicriterion shortest path problems. *Computers and Operations Research*, 27(6):507–524, 2000.

M. Sniedovich. A multi-objective routing problem revisited. *Engineering Optimization*, 13:99–108, 1988.