CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

New Labeling Algorithms for Minimum-Hop Bicriteria Path Problems

Marta M.B. Pascoal

December 2008

CIRRELT-2008-57

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

New Labeling Algorithms for Minimum-Hop Bicriteria Path Problems

Marta M.B. Pascoal[1,2]

[1] Invited researcher, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and Departamento de Matemática da Universidade de Coimbra, Apartado 3008, EC Universidade, 3001-454 Coimbra, Portugal

[2] Institute for Systems and Computers Engineering, Coimbra (INESCC), rua Antero de Quental, 199; 3000 – 033 Coimbra, Portugal

Abstract. The number of hops (or arcs) of a path is a frequent objective function with applications to problems where network resources utilization is to be minimized. In this paper we solve bicriteria path problems involving this objective function and two other common metrics, the path cost and the path capacity. We introduce labeling algorithms using a breadth-first search tree in order to compute the maximal and the minimal sets of non-dominated paths. The properties of this data structure are explored to better suit the number of hops objective function and thus simplify the labeling process. Computational experiments on randomly generated test instances are presented and discussed. Results show a significant speed-up over generic standard methods.

Keywords. Hop, cost, capacity, bicriteria path problems, labeling algorithms.

Acknowledgements. This work was partially funded by the FCT Portuguese Foundation of Science and Technology (Fundação para a Ciência e a Tecnologia) under grant SFRH/BSAB/830/2008. Thanks are also due to Gilbert Laporte for comments on a preliminary version of this manuscript.

---

* Corresponding author: marta@mat.uc.pt

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2008

1 Introduction

Optimal path problems arise in several contexts and with different types of objective functions. However, very often a single objective function cannot completely characterize a problem. Two of the most common objective functions used in these problems are the path cost, given by an additive function, and the path capacity, which is a bottleneck function. In 1980 Hansen [6] presented a list of several bicriteria path problems, studied their complexity, and adapted labeling algorithms to solve them. These problems include the MinSum-MinSum path problem (which minimizes two cost functions) and the MinSum-MaxMin path problem (which minimizes the cost and maximizes the capacity). Later Martins [7] generalized labeling algorithms for the MinSum path problem with more than two objective functions, and in [8] the same author studied the MinSum-MaxMin case and developed both an algorithm for finding the maximal set of non-dominated paths and another algorithm for finding simply the minimum set of non-dominated paths, thus computing the non-dominated objective values. Recently the algorithm presented by Martins [7] was extended by Gandibleux, Beugnies and Randriamasy [4] to cope with more than a single cost function and one bottleneck function. Other labeling algorithms [1, 10] have also been presented for the MinSum-MinSum case.

The number of arcs in a path, or the number of hops borrowing from the telecommunications terminology, is another common objective function with particular interest to telecommunications. In such problems it is frequent to look for a route using the maximum available bandwidth, or having the minimum cost (sometimes also related with the arcs bandwidth). When combining the objective function number of hops with the cost or with the bandwidth, the MinHop-MinSum or the MinHop-MaxMin path problems are obtained. These are particular cases of the MinSum-MinSum or the MinSum-MaxMin path problems, since the number of hops can be seen as an additive function where all arcs have cost equal to 1. Some problems related to these two variants can be found in [2, 5, 9]. Still, despite their large number of potential applications, to our knowledge no specific methods have been developed to deal with them.

This paper focuses on labeling algorithms for the MinHop-MinSum and the MinHop-MaxMin path problems, aiming to determine the minimal and the maximal sets of non-dominated paths. It is known that breadth-search allows to scan the nodes of a tree by order of the level they belong to, and therefore the proposed methods use a queue to manage the labels associated with each generated path. This results in a simplification of the labeling procedure, namely regarding the dominance test and in the decrease of the CPU time.

The remainder of the paper is organized as follows. Section 2 introduces notation, preliminary concepts and the problems. Section 3 proposes labeling algorithms for the minimum hop-shortest path problem from two points of view, the determination of all non-dominated paths or simply of those with distinct objective values, while Section 4 is devoted to the minimum hop-maximum capacity path problem. Section 5 presents the results of computational experiments and conclusions follow in Section 6.

2 Some bicriteria path problems

Let $(\mathcal{N}, \mathcal{A})$ be a directed network consisting of a set $\mathcal{N} = \{1, \ldots, n\}$ of nodes and of a set $\mathcal{A} = \{1, \ldots, m\}$ of arcs. Let s, the initial node, and t, the terminal node, be two distinct nodes in $(\mathcal{N}, \mathcal{A})$. A path from s to t in $(\mathcal{N}, \mathcal{A})$ is a sequence of the form $p = \langle v_1, \ldots, v_\ell \rangle$ where $v_1 = s$, $v_\ell = t$, $v_i \in \mathcal{N}$, $i = 1, \ldots, \ell$, and $(v_i, v_{i+1}) \in \mathcal{A}$, $i = 1, \ldots, \ell - 1$. For simplicity we write $(i, j) \in p$ if i and j are consecutive nodes and i precedes j in p. Let $\mathcal{P}$ denote the set of all paths from s to t in $(\mathcal{N}, \mathcal{A})$. With each arc $(i, j) \in \mathcal{A}$ are associated a cost $c_{ij} \in \mathbb{R}$ and a capacity $u_{ij} \in \mathbb{R}^+$. Then the cost of

path p is

$$c(p) = \sum_{(i,j)\in p} c_{ij},$$

its capacity is

$$u(p) = \min_{(i,j)\in p} \{u_{ij}\},$$

and its number of arcs is

$$h(p) = \ell - 1.$$

Usually the functions c and h are minimized, whereas u is maximized, and these functions are combined in some bicriteria path problems. The following sections focus on the MinHop-MinSum path problem, where h and c are minimized, and on the MinHop-MaxMin, where h is minimized and u is maximized.

In general, the two objective functions of a bicriteria problem are not correlated and there is no solution optimizing them simultaneously. Instead, the set of non-dominated paths, for which there is no other solution that improves one of the objectives without worsening the other, is computed. In the following definitions we borrow some terminology used by Gandibleux *et al.* [4] for multicriteria path problems. As we focus on the optimization of different objective functions we present a general definition of dominance.

Definition 1. *Let p_1, p_2 be two paths between the same pair of nodes in $(\mathcal{N}, \mathcal{A})$ and f_1, f_2 two functions defined for any path.*

1. *Path p_1 dominates path p_2 (denoted $p_1 {}_D p_2$) if and only if $f_i(p_1)$ is better than or equal to $f_i(p_2)$, $i = 1, 2$, and it is strictly better for at least one of the objective functions. It can also be said that $(f_1(p_1), f_2(p_1))$ dominates $(f_1(p_2), f_2(p_2))$ (denoted $(f_1(p_1), f_2(p_1)) {}_D (f_1(p_2), f_2(p_2))$).*

2. *Path p_1 strictly dominates path p_2 if and only if it is better than p_2 for f_1 and f_2. p_1, p_2 are equivalent when $f_i(p_1) = f_i(p_2)$, $i = 1, 2$.*

Definition 2. *A path $p \in \mathcal{P}$ is non-dominated, or efficient, if and only if it is not dominated by any other. If there is no path that strictly dominates p, then p is said to be weakly non-dominated, or weakly efficient.*

Definition 3. *Let $\mathcal{P}_D$ be the subset of dominated paths in $\mathcal{P}$. Then $\mathcal{P}_N = \mathcal{P} - \mathcal{P}_D$ denotes the maximal complete set of non-dominated paths in $\mathcal{P}$, while $\bar{\mathcal{P}}_N$, the minimal complete set of non-dominated paths denotes the largest subset of $\mathcal{P}_N$ that contains no equivalent solutions.*

3 MinHop-MinSum path problem

In labeling algorithms for bicriteria path problems several labels can be assigned to a network node. Each label corresponds to a path in the network starting from s, and thus a tree of paths rooted at s can be constructed. Some branches of this tree can be pruned by testing the dominance of new nodes and of those that are already in the tree, corresponding to paths. Like for the single criteria case, label setting and label correcting algorithms for these problems differ on the strategy they use to pick the next label to scan. If the lexicographically smallest label is taken, in the first case, then it is permanent, that is, non-dominated, after being scanned, while with label correcting algorithms non-dominated paths can only be known when all labels have been scanned.

For the MinHop-MinSum path problem a label associated with a node x in the search tree of paths from s to other nodes has the form $l_x = [\pi_x^h, \pi_x^c, \xi_x, \beta_x]$, where

- π_x^h denotes the number of arcs in the path from the root node to x,

- π_x^c denotes its cost,
- ξ_x is the node preceding x in that tree, and
- β_x is the network node that corresponds to x.

Given $\beta_x = i \in \mathcal{N}$ and $(i,j) \in \mathcal{A}$, a new node y can be considered in the tree, with the label

$$l_y = [\pi_x^h + 1, \pi_x^c + c_{ij}, x, j].$$

Denote by X the set of labels that are eligible to be scanned.

Given two labels l_x, l_y corresponding to a path ending at the same network node, we say that l_x is dominated by l_y if and only if

$$(\pi_x^h > \pi_y^h \text{ and } \pi_x^c \geq \pi_y^c) \text{ or } (\pi_x^h \geq \pi_y^h \text{ and } \pi_x^c > \pi_y^c). \tag{1}$$

Therefore, a new label l_y can be discarded if there is another node x already in the tree such that $\beta_x = \beta_y$ and

$$(\pi_x^h < \pi_y^h \text{ and } \pi_x^c \leq \pi_y^c) \text{ or } (\pi_x^h \leq \pi_y^h \text{ and } \pi_x^c < \pi_y^c). \tag{2}$$

On the other hand, whenever (1) holds for some label l_x in the search tree, then l_x can be replaced by l_y.

Now, considering the computation of the maximal set of non-dominated paths and assuming X is manipulated as a First In First Out list (FIFO), the number of arcs of the analyzed paths, i.e. the π_x^h values, forms a non-decreasing sequence [3]. Thus for a new label l_y condition $\pi_x^h \leq \pi_y^h$ always holds, so that the first part of (1) is never satisfied and (1) can be replaced by

$$\pi_x^h = \pi_y^h \text{ and } \pi_x^c > \pi_y^c. \tag{3}$$

Furthermore the labels associated with a certain node have particular properties, as shown in Lemma 1.

Lemma 1. *If X is a FIFO, then for each level of the tree of paths rooted at s a tree node is associated with several labels, if and only if all have the same objective function values.*

Proof. Let l_x and l_y be two labels at the same level in X, that correspond to the same node, i.e. $\beta_x = \beta_y$ and $\pi_x^h = \pi_y^h$. By contradiction, assuming that $\pi_x^c \neq \pi_y^c$, then:

1. either $\pi_x^c < \pi_y^c$, which implies $l_x D l_y$, so y should not belong to X,
2. or else $\pi_x^c > \pi_y^c$, therefore $l_y D l_x$, so x should not belong to X.

Moreover, if a network node has several non-dominated labels with the same objective values, then they share the same value of h and belong to the same paths tree level. □

This result yields a simplification of the acceptance condition of a new label l_y, which can be restated as

$$(\pi_x^h < \pi_y^h \text{ and } \pi_x^c > \pi_y^c) \text{ or } (\pi_x^h = \pi_y^h \text{ and } \pi_x^c \geq \pi_y^c), \tag{4}$$

for any $x \in X$ such that $\beta_x = \beta_y$, while if (3) holds for some node $x \in X$, this node can be removed from the tree since its label is dominated. Using the FIFO data structure to represent X, and conditions (3) and (4) as dominance rules, the following result holds.

Corollary 1. *If X is a FIFO, then the sequence of labels extracted from X and associated with a network node is in lexicographic order.*

Another consequence of Lemma 1 and Corollary 1 is that each label picked in X is non-dominated.

Corollary 2. *If X is a FIFO, then a label chosen in X is permanent.*

Two conclusions can be drawn from this result. First, non-dominated paths from s to t can be known since labels that correspond to t are chosen in X. Second, in order to check the dominance of a new label it is sufficient to compare it with the last label inserted in X associated with the same node. In the pseudo-code presented below this information is maintained in the n elements array *Last*.

Although X is manipulated as a FIFO in the sense that new elements are inserted at the end of the queue while the first one is scanned, an adaptation might have to be done when a new label l_y dominates a previous one, l_x. If l_x is the only label at level π_x^h, it is sufficient to replace l_x by l_y. However, if there are multiple labels associated with β_x at level π_x^h, then they are all dominated and should be deleted. An alternative is to include an additional comparison between labels that are picked in X and the correspondent *Last*, in order to check their dominance.

The pseudo-code of the method just described is presented in Algorithm 1.

Algorithm 1. ***MinHop-MinSum maximal set of paths determination***

```
For i ∈ 𝒩 Do Last_i ← 0
Last_s ← 1; nX ← 1; l_nX ← [0, 0, −, s]; X ← {1}; 𝒫_N ← ∅
While X ≠ ∅ Do
   x ← first node in X; X ⟵ X − {x}; i ← β_x
   If i = t Then 𝒫_N ← 𝒫_N ∪ {x}
   For j ∈ 𝒩 such that (i, j) ∈ A Do
      y ← Last_j
      If (y = 0) or (y ≠ 0 and π_x^h + 1 > π_y^h and π_x^c + c_ij < π_y^c) Then
         nX ← nX + 1; l_nX ← [π_x^h + 1, π_x^c + c_ij, x, j]; Insert nX at the end of X
         Last_j ← nX
      Else If π_x^h + 1 = π_y^h and π_x^c + c_ij = π_y^c Then
                nX ← nX + 1; l_nX ← [π_y^h, π_y^c, x, j]; Insert nX at the end of X
           Else If π_x^h + 1 = π_y^h and π_x^c + c_ij < π_y^c Then
                     l_y ← [π_y^h, π_x^c + c_ij, x, j]
                     Remove from X other labels associated with j with the objective values of x
                EndIf
   EndFor
EndWhile
```

If the goal is to find the non-dominated objective values, i.e. the minimal set of non-dominated paths, it is sufficient to compute a single path for each objective values pair, which allows to make certain modifications to the method above. The main difference now is that in such a case at most one label is used for each node and each number of arcs, so Lemma 1 is now replaced by the following result.

Lemma 2. *If X is a FIFO, then at most one label per level is associated with a network node.*

Moreover, Corollaries 1 and 2 are still valid. Thus, concerning the minimal set of paths computation a new label l_y should be inserted in X if and only if

$$(\pi_x^h < \pi_y^h \text{ and } \pi_x^c > \pi_y^c) \text{ or } (\pi_x^h = \pi_y^h \text{ and } \pi_x^c > \pi_y^c), \tag{5}$$

for any $x \in X$ such that $\beta_x = \beta_y$, and in the second case, i.e., if

$$\pi_x^h = \pi_y^h \text{ and } \pi_x^c > \pi_y^c, \tag{6}$$

l_x can be discarded. The fact that there is a single label associated with each pair of objective values also allows to replace labels, with no need for deletions, if they become dominated by others. The pseudo-code of Algorithm 2 is a simplified version of Algorithm 1 for finding the minimal set of paths.

Algorithm 2. ***MinHop-MinSum minimal set of paths determination***

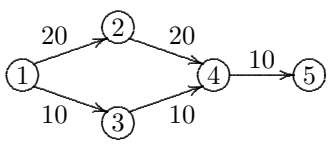
```
For i ∈ 𝒩 Do Last_i ← 0
Last_s ← 1; nX ← 1; l_nX ← [0, 0, −, s]; X ← {1}; P̄_N ← ∅
While X ≠ ∅ Do
    x ← first node in X; X ⟵ X − {x}; i ← β_x
    If i = t Then P̄_N ← P̄_N ∪ {x}
    For j ∈ 𝒩 such that (i, j) ∈ A Do
        y ← Last_j
        If (y = 0) or (y ≠ 0 and π^h_x + 1 > π^h_y and π^c_x + c_ij < π^c_y) Then
            nX ← nX + 1; l_nX ← [π^h_x + 1, π^c_x + c_ij, x, j]; Insert nX at the end of X
            Last_j ← nX
        Else If π^h_x + 1 = π^h_y and π^c_x + c_ij < π^c_y Then l_y ← [π^h_y, π^c_x + c_ij, x, j]
    EndFor
EndWhile
```

The MinHop-MinSum path problem has up to $n(n-2)$ non-dominated pairs of objective values, if there is a path with every number of arcs between 1 and $n-2$ from s to any other node i. Therefore the tree of paths obtained by Algorithm 2 can have at most $n-2$ levels. In the worst case, every of the m network arcs has to be scanned once for each of those levels. Analyzing an arc implies the insertion and deletion of an element in X, which can be done in constant time. Therefore the worst-case time complexity of Algorithm 2 is $\mathcal{O}(nm)$.

As for Algorithm 1, its theoretical complexity also depends on the number of levels the paths tree can have and on the total number of non-dominated labels. Even though the tree can have up to $n-2$ levels, because there may be multiple labels with the same objective values, the number of labels cannot be polynomially bounded.

4 MinHop-MaxMin path problem

The MinHop-MaxMin path problem can be viewed as a special case of the problems solved by Martins and by Gandibleux *et al.* [4, 7] considering a single cost function and $c_{ij} = 1$ for any $(i,j) \in \mathcal{A}$. Like in the previous section, labeling algorithms can also be designed for the MinSum-MaxMin path problem, by including the number of hops and the capacity values in each label and modifying the label dominance test. However, non-dominated paths may contain weakly non-dominated subpaths [4]. Therefore weakly non-dominated labels can be necessary to determine the maximal set of paths. As an example, the paths $\langle 1,2,4,5\rangle$ and $\langle 1,3,4,5\rangle$ between nodes 1 and 5 in the network depicted in Figure 1 are both non-dominated with respect to the MinHop-MaxMin path problem, although $\langle 1,2,4\rangle_D\langle 1,3,4\rangle$.



p	$(h(p),u(p))$	p	$(h(p),u(p))$
$\langle 1,2,4\rangle$	$(2,20)$	$\langle 1,2,4,5\rangle$	$(3,10)$
$\langle 1,3,4\rangle$	$(2,10)$	$\langle 1,3,4,5\rangle$	$(3,10)$

Figure 1: Efficient solutions formed by weakly efficient subpaths

Maintaining the previous notation a label associated with a node x in the search tree of paths from s to other nodes has now the form $l_x = [\pi_x^h, \pi_x^u, \xi_x, \beta_x]$, where π_x^u denotes the path capacity. Furthermore, if $\beta_x = i \in \mathcal{N}$ and $(i,j) \in \mathcal{A}$, a new label associated with node y can be created, such that

$$l_y = [\pi_x^h + 1, \min\{\pi_x^u, u_{ij}\}, x, j].$$

Now consider that instead of minimizing a cost function a capacity function is maximized. Given l_x, l_y two labels correspondent to a path starting at s and ending at the same network node, we say l_x is dominated by l_y if and only if

$$(\pi_x^h > \pi_y^h \text{ and } \pi_x^u \leq \pi_y^u) \text{ or } (\pi_x^h \geq \pi_y^h \text{ and } \pi_x^u < \pi_y^u). \tag{7}$$

Taking into account that in this problem weakly non-dominated labels can be used to obtain non-dominated solutions, namely solutions with the same number of hops but different capacities, a new label l_y should only be discarded if there is another x in X such that $\beta_x = \beta_y$ and

$$\pi_x^h < \pi_y^h \text{ and } \pi_x^u \geq \pi_y^u. \tag{8}$$

On the other hand, l_x can be replaced by l_y whenever

$$\pi_x^h > \pi_y^h \text{ and } \pi_x^u \leq \pi_y^u. \tag{9}$$

Back to the determination of the maximal set of non-dominated paths using a FIFO, as the number of arcs of the scanned paths is non-decreasing (9) never holds, which means no label should be deleted and the dominance test whenever a new label is formed can be replaced simply by (8).

The acceptance of weakly non-dominated labels that may be dominated implies that Corollaries 1 and 2 now fail. For this reason the set $\mathcal{P}_N$ is only known after the labeling process is over. Moreover, it is not enough to compare a new label with the last one observed for that node, and two cases should be distinguished:

1. $\pi_x^h = \pi_y^h$ for some x in X, then node β_x already has a label at level π_y^h and neither l_x strictly dominates l_y nor l_y strictly dominates l_x, therefore y is inserted in X;

2. $\pi_x^h < \pi_y^h$ for every x in X, then l_y belongs to a different level than l_x and it should be inserted if and only if $\pi_x^u < \pi_y^u$, that is, if and only if

$$\max\{\pi_x^u : x \in X \text{ and } \pi_x^h < \pi_y^h\} < \pi_y^u. \tag{10}$$

In short, a candidate label will only be accepted if its capacity improves the capacity of the labels that have less arcs. An auxiliary array storing the best capacity value found for each network node until the latest scanned level, $Best$, is used in Algorithm 3, which summarizes the pseudo-code for finding the maximal set of MinHop-MaxMin paths.

Algorithm 3. ***MinHop-MaxMin maximal set of paths determination***

```
For i ∈ 𝒩 Do
    Best_i ← 0; Last_i ← 0; π_i^u ← 0
EndFor
Last_s ← 1; nX ← 1; l_nX ← [0, +∞, −, s]; X ← {1}
While X ≠ ∅ Do
    x ← first node in X; X ⟵ X − {x}; i ← β_x
    For j ∈ 𝒩 such that (i, j) ∈ 𝒜 Do
        If min{π_x^u, u_ij} > Best_j Then
```

$y \leftarrow Last_j$
If $(y = 0)$ or $(y \neq 0$ and $\pi_x^h + 1 > \pi_y^h)$ Then
$Best_j \leftarrow \pi^u_{Last_j}$
If $\min\{\pi_x^u, u_{ij}\} > Best_j$ Then
$nX \leftarrow nX + 1$; $l_{nX} \leftarrow [\pi_x^h + 1, \min\{\pi_x^u, u_{ij}\}, x, j]$; Insert nX at the end of X
$Last_j \leftarrow nX$
EndIf
Else If $\pi_x^h + 1 = \pi_y^h$ Then
$nX \leftarrow nX + 1$; $l_{nX} \leftarrow [\pi_y^h, \min\{\pi_x^u, u_{ij}\}, x, j]$; Insert nX at the end of X
If $\min\{\pi_x^u, u_{ij}\} > \pi_y^u$ Then $Last_j \leftarrow nX$
EndIf
EndFor
EndWhile
$\mathcal{P}_N \leftarrow \{$*non-dominated paths from* s *to* $x \in X$ *where* $\beta_x = t\}$

Even though weakly non-dominated subpaths have to be generated, node t can be treated differently from the others because only non-dominated t labels are necessary (otherwise the label corresponds to a solution that contains a loop and is dominated). A different dominance test can then be applied, because a new label l_y, $\beta_y = t$, should be discarded if and only if there is another one l_x, $\beta_x = t$, such that

$$(\pi_x^h < \pi_y^h \text{ and } \pi_x^u \geq \pi_y^u) \text{ or } (\pi_x^h = \pi_y^h \text{ and } \pi_x^u > \pi_y^u), \tag{11}$$

and it should replace l_x if

$$\pi_x^h = \pi_y^h \text{ and } \pi_x^u < \pi_y^u. \tag{12}$$

This variation of Algorithm 3 ensures that t labels are non-dominated as soon as they are chosen in X, thus allowing the generation of non-dominated paths between s and t along the labeling process.

If again the aim is the determination of the minimal set of non-dominated paths at most one label is stored for each node in a tree level, therefore no dominated subpaths need to occur in non-dominated solutions and the following result follows.

Proposition 1. *Let* $p^* \in \mathcal{P}_N$ *for the MinHop-MaxMin path problem, then there is* $p \in \mathcal{P}$ *formed by non-dominated subpaths from* s *to any node such that* $(h(p), u(p)) = (h(p^*), u(p^*))$.

This result can be used to tighten the dominance test, as a new label l_y can be discarded if there is l_x such that $\beta_x = \beta_y$ and

$$(\pi_x^h < \pi_y^h \text{ and } \pi_x^u \geq \pi_y^u) \text{ or } (\pi_x^h \leq \pi_y^h \text{ and } \pi_x^u > \pi_y^u),$$

that is, as X is manipulated as a FIFO, if

$$\pi_x^u \geq \pi_y^u. \tag{13}$$

The same label will replace l_x whenever

$$\pi_x^h = \pi_y^h \text{ and } \pi_x^u < \pi_y^u. \tag{14}$$

Moreover, in this case the labels associated with a network node i correspond to a non-dominated path from s to i, and thus a non-dominated path from s to t is obtained whenever a label associated with t is chosen in X. The resulting method is very similar to Algorithm 2, but its pseudo-code is shown in Algorithm 4 for the sake of completeness.

Algorithm 4. ***MinHop-MaxMin minimal set of paths determination***

```
For i ∈ 𝒩 Do Last_i ← 0
Last_s ← 1; nX ← 1; X ← {1}; l_nX ← [0, +∞, −, s]; 𝒫_N ← ∅
While X ≠ ∅ Do
    x ← first node in X; X ⟵ X − {x}; i ← β_x
    If i = t Then 𝒫_N ← 𝒫_N ∪ {x}
    For j ∈ 𝒩 such that (i, j) ∈ A Do
        y ← Last_j
        If (y = 0) or (y ≠ 0 and π^h_x + 1 > π^h_y and min{π^u_x, u_ij} > π^u_y) Then
            nX ← nX + 1; l_nX ← [π^h_x + 1, min{π^u_x, u_ij}, x, j]; Insert nX at the end of X
            Last_j ← nX
        Else If π^h_x + 1 = π^h_y and min{π^u_x, u_ij} > π^u_y Then l_y ← [π^h_y, min{π^u_x, u_ij}, x, j]
    EndFor
EndWhile
```

The complexity of Algorithms 3 and 4 can be determined as for the MinHop-MinSum path problem.

5 Computational results

Computational experiments were carried out to evaluate the empirical performance of the methods described in the previous sections. To this end, a set of random networks with 1 000, 3 000, 5 000 and 7 000 nodes, dn arcs, for densities $d = 5, 10, 20, 30$, and uniformly integer cost (capacity) values generated in $[1, M]$, with $M = 100$ and $M = 10\,000$, was considered. The results presented in the following were obtained on 30 different instances generated for each dimension of this data set. Tests were executed on a Dual Core AMD Opteron at 2 GHz, with 4 Gb of RAM running over SUSE Linux 10.3.

5.1 MinHop-MinSum path problem

In order to evaluate the methods proposed for finding the maximal set of MinHop-MinSum paths two programs were coded in C, namely a labeling algorithm where X is as FIFO list with a standard dominance test, `F1`, and Algorithm 1, `A1`. Similarly, when concerning only the determination of the minimal set of paths a standard labeling algorithm using a FIFO, `F2`, and Algorithm 2, `A2`, have been implemented.

	n			
	1000	3000	5000	7000
Minimum	3	3	2	2
Average	4.7	4.3	3.7	4.6
Maximum	6	6	6	7

Table 1: Number of MinHop-MinSum non-dominated paths with $M = 100$ and $d = 30$

Table 1 presents the minimum, average and maximum number of non-dominated paths from s to t, regarding the case of costs in $[1, 100]$ and $d = 30$. Even though t can have $n - 2$ different non-dominated labels in the worst-case, the results show the actual number of labels to be much smaller. In the entire test set the registered average number of $\mathcal{P}_N$ elements is always between 2 and 5, increasing slowly with density. When $M = 10\,000$ the results are very similar in general, only with some higher minimum and average number of non-dominated paths.

100×(F1-A1)/F1				
	d			
n	5	10	20	30
1000	0.0	58.3	35.0	25.9
3000	37.5	32.4	40.0	41.7
5000	8.3	34.4	47.6	44.2
7000	20.9	43.0	39.4	44.5

100×(F2-A2)/F2				
	d			
n	5	10	20	30
1000	100.0	47.1	32.0	31.7
3000	14.3	36.4	45.5	50.5
5000	57.1	39.4	46.0	50.2
7000	41.4	43.7	44.3	47.1

Table 2: Percentage average CPU times improvement of MinHop-MinSum non-dominated paths with $M = 100$

A comparison between the average running times of standard versions of a labeling algorithm, F1 and F2, and the methods introduced for finding the maximal and the minimal sets of paths, A1 and A2, is presented in Table 2, with average values of the improvement obtained by the new algorithms. In most cases the CPU time is approximately reduced by half, although in some small size instances (with low density when finding the minimal set) the results are better for the standard versions. However, for these dimensions all running times are very close to 0 second. The algorithmic performance is very similar for the two cost ranges, although the times are slightly greater for the wider range. It is still worth noting that the minimal set determination is easier than that of the maximal set. The difference seems to increase with the instances dimension. The CPU times taken by any of the implemented algorithms were very small for small size instances. Figures 2 and 3 show the average running times for the A1 and A2 codes. The first two plots concern the variation depending on the number of network nodes, while the other two depend on the network density. There is an increase of running times with n as well as with d, both for A1 and A2 as the theoretical complexity bound suggests. The determination of the maximal, and of the minimal, sets of MinHop-MinSum paths in the considered data set was made in short times. A1 and A2 were able to solve problems with 7 000 nodes and 210 000 arcs in less than 0.043 seconds.

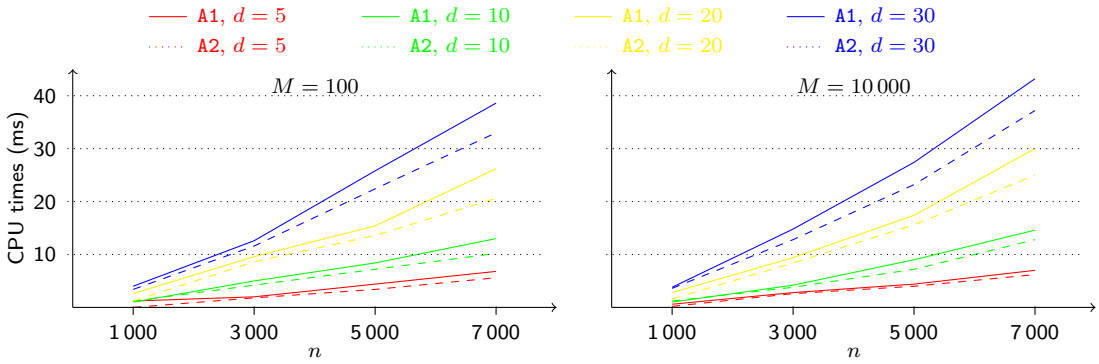


Figure 2: Average CPU times MinHop-MinSum non-dominated paths versus n

5.2 MinHop-MaxMin path problem

The procedure followed for this problem is analogous to the one used in the previous section. Two standard labeling algorithm using a FIFO were coded: one for finding the maximal and the other for finding the minimal sets of MinHop-MaxMin paths, F3 and F4, as well as Algorithms 3 and 4, A3 and A4, respectively.

A sample of the minimum, average and maximum number of solutions in the tested instances with capacity values in $[1, 100]$ and $d = 30$ is provided in Table 3. These values are greater than

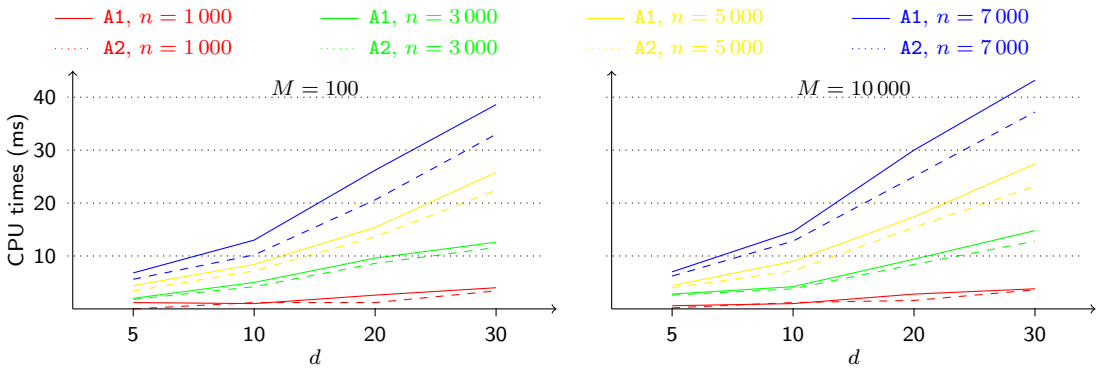


Figure 3: Average CPU times MinHop-MinSum non-dominated paths versus d

for the latter problem. On average the maximal set of paths has between 4 and 8 non-dominated solutions, but are still far from the upperbound $n-2$. For instances with $M = 10\,000$ the average and the maximum values can increase by one to two solutions.

	n			
	1000	3000	5000	7000
Minimum	4	2	3	3
Average	7.1	7.9	8.5	9.5
Maximum	15	14	15	18

Table 3: Number of MinHop-MaxMin non-dominated paths with $M = 100$ and $d = 30$

Table 4 shows the relation between the original algorithms, `F3` and `F4`, and the introduced methods, `A3` and `A4`, for the minimal and the maximal sets determination. The improvement on running times is around 75% for finding all non-dominated paths and between 75% and 100% for finding the non-dominated objective values, regardless of the instance size. As expected these values are greater than the observed for the MinHop-MaxSum path problem, as much more weakly non-dominated labels now have to be stored. The general tendancy is the same when $M = 10\,000$, but the improvement is around 80% for codes `F3` and `A3`, and around 70% for `F4` and `A4`.

100×(F3-A3)/F3				
	d			
n	5	10	20	30
1000	69.1	73.5	78.2	83.0
3000	67.9	75.6	83.1	84.6
5000	69.0	80.1	85.0	87.2
7000	71.9	82.5	87.2	89.5

100×(F4-A4)/F4				
	d			
n	5	10	20	30
1000	83.3	90.2	96.2	97.0
3000	86.7	93.1	96.5	97.5
5000	90.2	95.1	97.4	99.1
7000	92.2	95.4	97.8	98.5

Table 4: Percentage average CPU times improvement of MinHop-MaxMin non-dominated paths with $M = 100$

The difference in performance when determining the maximal and the minimum sets of paths is also evident on the plots in Figures 4 and 5, which show the running times of `A3` and `A4` variation with n and with d, respectively. Solid lines, depicting `A3` results, grow much faster than dashed lines, for `A4`. The growth seems to present a linear behaviour with n and like in the previous section, quadratic with d. For larger instances it took in average 0.374 seconds to find the maximal set of paths and 0.076 seconds to find the minimal set of non-dominated paths.

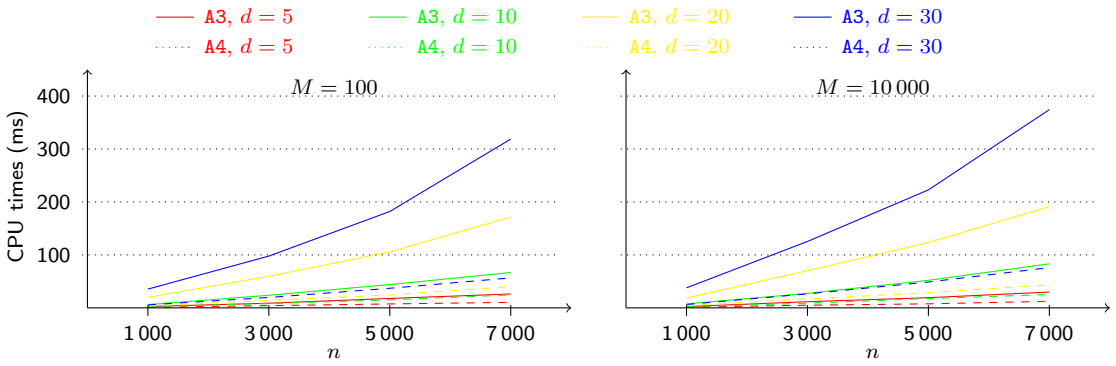


Figure 4: Average CPU times MinHop-MaxMin non-dominated paths versus n

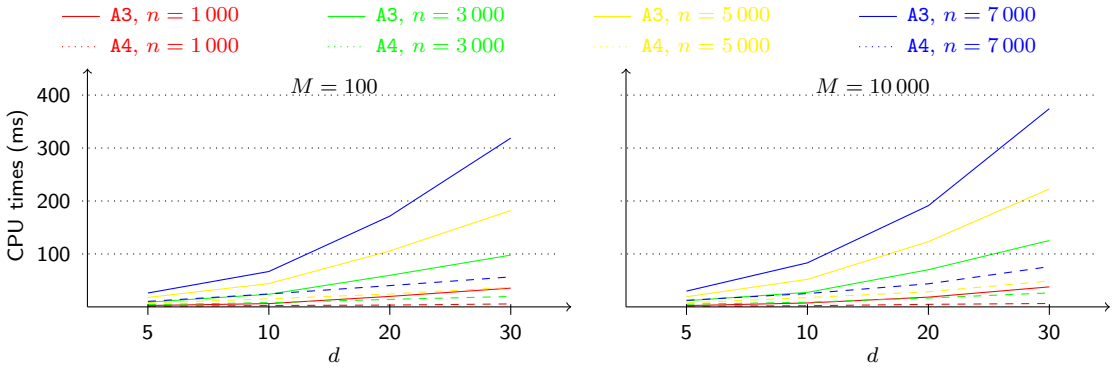


Figure 5: Average CPU times MinHop-MaxMin non-dominated paths versus d

6 Conclusions

Labeling algorithms for bicriteria path problems minimizing the number of hops and either the path cost or the path capacity have been described, aiming the maximal and the minimal sets of non-dominated paths computation. These methods make use of a breadth-first search tree by managing the set of labels as a FIFO and list node labels by non-decreasing order of the number of hops. Tuning the dominance test according with this structure leads to an improvement of about 50% for the MinHop-MinSum path problem running times and between 75% and 100% for the MinHop-MaxMin path problem over randomly generated instances. Numerical results indicate that only 0.043 seconds are necessary to find the whole set of non-dominated paths in the first case, and 0.037 seconds are needed in the second case, for instances with 7 000 nodes and 210 000 arcs. The determination of the non-dominated objective values was completed with 0.374 seconds for the MinHop-MinSum and within 0.076 seconds for the MinHop-MaxMin path problems over the same test bed.

Acknowledgments This work was partially funded by the FCT Portuguese Foundation of Science and Technology (Fundação para a Ciência e a Tecnologia) under grant SFRH/BSAB/830/2008. Thanks are also due to Gilbert Laporte for comments on a preliminary version of this manuscript.

References

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

[2] G. Cheng and N. Ansari. Finding a least hop(s) path subject to multiple additive constraints. *Computer Communications*, 29:392–401, 2006.

[3] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 2001.

[4] X. Gandibleux, F. Beugnies, and S. Randriamasy. Multi-objective shortest path problems with a maxmin cost function. *4OR – Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 4:47–59, 2006.

[5] R. Guerin and A. Orda. Computing shortest paths for any number of hops. *IEEE/ACM Trans. Netw.*, 10:613–620, 2002.

[6] P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making: Theory and Applications, Lectures Notes in Economics and Mathematical Systems*, 177, pages 109–127. Springer-Verlag, Berlin, 1980.

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

[8] E. Martins. On a special class of bicriterion path problems. *European Journal of Operational Research*, 17:85–94, 1984.

[9] S. Randriamasy, L Fourni, and D Hong. Distributed adaptive multi-criteria load balancing: analysis and end to end simulation. In *INFOCOM Poster and Demo Session*, Barcelona, April 2006.

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