Computers & Operations Research 38 (2011) 165–173

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Minimum cost path problems with relays

Gilbert Laporte[a,b], Marta M.B. Pascoal[c,d,*]

[a] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada
[b] Canada Research Chair in Distribution Management, HEC, Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
[c] Departamento de Matemática da Universidade de Coimbra, Apartado 3008, EC Universidade, 3001-454 Coimbra, Portugal
[d] Institute for Systems and Computers Engineering–Coimbra (INESCC), Portugal

ARTICLE INFO

Available online 29 April 2010

Keywords:
Relays
Shortest path problem
Bicriteria optimization
Labeling algorithms

ABSTRACT

The *minimum cost path problem with relays* (MCPPR) consists of finding a minimum cost path from a source to a destination, along which relay nodes are located at a certain cost, subject to a weight constraint. This paper first models the MCPPR as a particular bicriteria path problem involving an aggregated function of the path and relay costs, as well as a weight function. A variant of this problem which takes into account all three functions separately is then considered. Formulating the MCPPR as a part of a bicriteria path problem allows the development of labeling algorithms in which the bound on the weight of paths controls the number of node labels. The algorithm for this constrained single objective function version of the problem has a time complexity of $\mathcal{O}(Wm+Wn\log(\max\{W,n\}))$, where n is the number of nodes, m is the number of arcs and W is the weight upper bound. Computational results on random instances with up to 10 000 nodes and 100 000 arcs, are reported.

1. Introduction

The purpose of this paper is to describe exact algorithms for the single criterion and the bicriteria *minimum cost path problem with relays* (MCPPR). The goal of this problem is to determine a least cost route between two nodes of a network, subject to a resource weight constraint. In order to satisfy this constraint the route nodes can be used as relays. This resets the transported weight to 0 but generates a node dependent cost. The optimal solution thus comprises a route as well as a set of relay nodes along it. The MCPPR is defined on a directed network $(\mathcal{N},\mathcal{A})$ with a set $\mathcal{N}=\{1,\dots,n\}$ of nodes and a set $\mathcal{A}=\{1,\dots,m\}$ of arcs. Given two nodes x and y a path from x to y is represented by a sequence $p=\langle v_1,v_2,\dots,v_\ell\rangle$, where $v_1=x$, $v_\ell=y$, $v_i\in\mathcal{N}$, $i=1,\dots,\ell$, and $(v_i,v_{i+1})\in\mathcal{A}$, $i=1,\dots,\ell-1$. For simplicity we write $i\in p$ if i is a node in the sequence p, and $(i,j)\in p$ if i immediately precedes j in p. Two nodes are distinguished in $\mathcal{N}$: an origin s and a destination t. We denote by $\mathcal{P}$ the set of paths from s to t in $(\mathcal{N},\mathcal{A})$. Given $i,j\in p$, the subpath of p between nodes i and j is denoted by $\sigma_p(i,j)$. If a path p_1 ends at a node where another path p_2 starts, then their concatenation is denoted by $p_1\diamond p_2$.

With each arc $(i,j)\in\mathcal{A}$ are associated non-negative parameters, a cost $c_{ij}\in\mathbb{R}_0^+$ and a weight $w_{ij}\in\mathbb{R}_0^+$. Also, using a network node $i\in\mathcal{N}$ as a relay generates a fixed cost $r_i\in\mathbb{R}_0^+$. If $r_i=0$, this means that adding a relay at node i does not carry any extra cost. This is the case for s and t, which are assumed to have relays with $r_s=r_t=0$. The MCPPR aims to determine a least cost path and the locations of relays under a weight constraint. We will write $\hat{i}$ to distinguish a relay node $i\in\mathcal{N}-\{s,t\}$ of a path from the remaining nodes that are not relays. As an example, in the network $(\mathcal{N}_1,\mathcal{A}_1)$ of Fig. 1, $q=\langle 1,2,4\rangle$ and $q'=\langle 1,\hat{2},4\rangle$ are paths from 1 to 4, and thus two possible solutions of the MCPPR: path q contains no relay besides s and t, whereas q' has a relay at node 2. The objective function of each of them depends on the arc and the relay costs, whereas the weight of any subpath is the accumulated weight after the last relay (or the first node if no relays are located). A formal definition of the functions involved is presented in the following.

The objective function $f(p)$ is the sum of the path cost

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

and of the cost of the selected relays,

$$r(p)=\sum_{\hat{i}\in p}r_i.$$

In addition, the weight of any subpath between two consecutive relays cannot exceed a given upper bound $W>0$. The weight of a path q from x to y, without relay nodes except possibly at nodes x

* Corresponding author at: Departamento de Matemática da Universidade de Coimbra, Apartado 3008, EC Universidade, 3001-454 Coimbra, Portugal. Tel.: +351 239 791150; fax: +351 239 832568.
E-mail addresses: Gilbert.Laporte@cirrelt.ca (G. Laporte), marta@mat.uc.pt (M.M. Pascoal).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.04.010

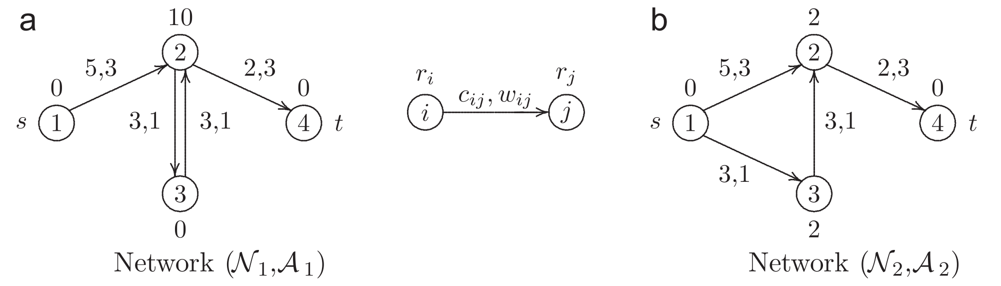

Fig. 1. Two instances of the MCPPR.

or y, is given by

$$w(q) = \sum_{(i,j) \in q} w_{ij}.$$

This function can be extended to another function $\tilde{w}$, which takes the same value as $w(q)$ for a path q from x to y with no relay nodes. This extension will be used later to compare the weights of intermediate paths between s and t, so that non-efficient solutions can be discarded at an early stage of the algorithm. Because the weight is reset to 0 whenever a relay node is located, $\tilde{w}(q)$ is the weight of the subpath of q from its last relay node until y. Thus, if q has no relay nodes

$$\tilde{w}(q) = w(q) = \sum_{(i,j) \in q} w_{ij},$$

whereas if $(\hat{s}_1,\ldots,\hat{s}_k)$ is the sequence of relay nodes of q other than x and y, then

$$\tilde{w}(q) = \begin{cases} w(\sigma_q(\hat{s}_k, \quad y)) & \text{if } y=t \text{ or } y \neq t \text{ is not a relay node,} \\ 0 & \text{if } y \neq t \text{ is a relay node.} \end{cases}$$

Let $R(q)$ be the the set of relay nodes of q. Again for paths $q = \langle 1,2,4 \rangle$ and $q' = \langle 1,\hat{2},4 \rangle$ in $(\mathcal{N}_1,\mathcal{A}_1)$ we have $R(q)=\emptyset$, $R(q')=\{\hat{2}\}$, and $f(q)=7$, $\tilde{w}(q)=w(q)=6$, $f(q')=17$, $\tilde{w}(q')=w(\langle \hat{2},4 \rangle)=3$.

The function f aggregates c and r. For instance, still in $(\mathcal{N}_1,\mathcal{A}_1)$ and considering $W=5$, $q'' = \langle 1,2,\hat{3},2,4 \rangle$ is the optimal solution of the MCPPR. However, q' is also feasible, and better than q'' in terms of cost $(c(q'')=13 > c(q')=7)$ but worse in terms of relay cost $(r(q'')=0 < r(q')=10)$. Motivated by situations like this, we will consider a variant of the MCPPR which consists of minimizing c and r, while looking for feasible paths in terms of weight. This will be called the *bicriteria minimum cost path problem with relays* (BMCPPR).

The MCPPR arises in the context of a telecommunications network design problem. It has been introduced and studied in [4,5], and applied to several contexts in [2,6–8]. In [5] three methods are proposed to find a minimum cost path with relays. The most efficient has a complexity of $\mathcal{O}(Wnm\log W)$. In this paper we develop a more efficient algorithm. We first address the original version of the MCPPR and then study the variant that considers arc and relay costs separately.

The literature on the MCPPR is rather limited. A problem of mail delivery is studied in [2]. It consists of minimizing the number of relay box locations at which postmen can replenish their mail bags along their route, given that they can only carry a limited amount of mail at any time. The problem is formulated as a set covering problem and solved heuristically. The works [4,5] describe three exact pseudo-polynomial algorithms for the MCPPR. The first solves a shortest path problem on an expansion of the network, the second works on the original network and uses a label-correcting procedure to maintain Pareto optimal solutions while considering the relay fixed costs, while the third uses a label-correcting algorithm combined with a merge-sort structure to improve the computational complexity of the second algorithm. As mentioned, the most efficient of these methods has a time complexity of $\mathcal{O}(Wnm\log W)$. Ref. [6] focuses on the network design problem with relays. This problem is defined on a graph in which several commodities must be moved between origin-destination pairs. The problem consists of selecting a subset of edges and of locating relays at a subset of vertices in order to minimize the sum of edge costs and relay costs, in such a way that there exists a path linking the origin and the destination of each commodity in which the length between any two consecutive relays does not exceed a preset upper bound. A lower bound procedure and four heuristics are developed. Ref. [7] proposes a solution methodology for the design of a wide area telecommunication network. This comprises the network design problem as described in [6], and the loading problem which consists of determining which signal transport technology should be installed on the selected edges of the network. Mathematical models are described for these subproblems, and a tabu search algorithm is developed. Finally, [8] deals with the black and white traveling salesman problem, whose aim is to design a shortest Hamiltonian tour on a graph with black or white vertices, subject to upper bound constraints on the number of white vertices and on the length of the path between two consecutive black vertices. The paper proposes an integer linear formulation and classes of valid inequalities, and develops an exact branch-and-cut algorithm. The algorithm can also be applied directly to the unit demand vehicle routing problem.

The remainder of the paper is organized as follows. Section 2 is devoted to the MCPPR. After the problem is defined, two labeling algorithms to find the optimal path are described and the theoretical results that support them are presented. In Section 3 the MCPPR objective function is decomposed into two, the BMCPPR is defined and methods for dealing with this problem are developed. Section 4 presents results of computational experiments on random instances for the problems studied in Sections 2 and 3. Conclusions follow in Section 5.

2. The minimum cost path problem with relays

The purpose of the MCPPR is to determine a feasible path between s and t, i.e., a path that satisfies the weight constraint and has a minimum objective function value. In other words, denoting by

$$\overline{\mathcal{P}} = \{p \in \mathcal{P} : w(\sigma_p(\hat{s}_{i-1},\hat{s}_i)) \leq W (i=2,\ldots,k \text{ and } \hat{s}_1,\ldots,\hat{s}_k \text{ the sequence of relay nodes of } p)\}$$

the set of feasible paths in $\mathcal{P}$, the MCPPR aims to find a path from s to t in $(\mathcal{N},\mathcal{A})$ satisfying

$$\min_{p \in \overline{\mathcal{P}}} \{f(p)\}. \tag{1}$$

This problem can be seen as a more general case of the weight constrained shortest path problem. However, in the MCPPR the relay location affects both the weight constraint and the relay costs involved in the objective function itself. One of the differences with respect to the shortest path problem is that an optimal path can have loops if it contains a relay node. For instance, when $W=5$ the minimum cost path with relays in the network in Fig. 1a is $q'' = \langle 1,2,\hat{3},2,4 \rangle$. Another difference is the fact that optimal paths may contain subpaths that are not optimal, which means that labeling algorithms supported by Bellman's principle of optimality [1] cannot be applied directly to the MCPPR. The only optimal solution for the MCPPR in network $(\mathcal{N}_2,\mathcal{A}_2)$ of Fig. 1b, still considering $W=5$, is path $\langle 1,3,2,4 \rangle$. However $\langle 1,3,2 \rangle$, which is feasible and has objective value 6, is not optimal from 1 to 2 because $\langle 1,2 \rangle$ is feasible and has objective value 5.

A variant of (1) is obtained if the weight is considered as an objective function

$$\min_{p \in \overline{\mathcal{P}}} \{f(p)\}, \quad \min_{p \in \overline{\mathcal{P}}} \{\tilde{w}(p)\}. \tag{2}$$

If f and $\tilde{w}$ are not correlated, then (2) may have no optimal solution. Instead, the set of feasible efficient paths can be defined as the set of feasible solutions for which there is no other solution that improves one of the objectives without worsening the other. Given two feasible paths p_1, p_2 between the same pair of nodes, i and j, p_1 dominates p_2 ($p_1{}_D p_2$) if and only if $f(p_1) \le f(p_2)$, $\tilde{w}(p_1) \le \tilde{w}(p_2)$, and at least one of the inequalities is strict. Then it is also said that $(f(p_1),\tilde{w}(p_1))$ dominates $(f(p_2),\tilde{w}(p_2))$, which is denoted by $(f(p_1),\tilde{w}(p_1))_D(f(p_2),\tilde{w}(p_2))$. A feasible path p is efficient if and only if it is not dominated by any other. Our algorithms produce sets of paths that contain one path for each non-dominated pair of objective values but there is no guarantee that all efficient paths will be generated.

Observe that if the MCPPR is feasible then there is an optimal solution that is also an efficient solution of (2). Also, finding a set of efficient paths for this bicriteria problem can be achieved by labeling procedures. In fact, even though the function $\tilde{w}$ is not additive, a variant of the principle of optimality can be proved if the procedure is restricted to feasible paths, and thus a labeling algorithm can be used. Finally, if a path is obtained for every non-dominated pair of objective values, then one of them is an optimal solution to (1).

Proposition 1. *If* (1) *is feasible it has an optimal solution that is an efficient solution of* (2).

Proof. Let p^* be a minimum cost path with relays from s to t, so that p^* satisfies the weight constraint and $f(p^*)$ is the minimum value for paths in $\overline{\mathcal{P}}$. Let $\overline{p}$ be another path from s to t such that $f(\overline{p}) = f(p^*)$ and $\tilde{w}(\overline{p})$ is minimum in $\overline{\mathcal{P}}$, and, by contradiction, assume $\overline{p}$ is dominated by another path $p \in \overline{\mathcal{P}}$, i.e.,

1. $f(\overline{p}) \ge f(p)$ and $\tilde{w}(\overline{p}) > \tilde{w}(p)$, or
2. $f(\overline{p}) > f(p)$ and $\tilde{w}(\overline{p}) \ge \tilde{w}(p)$.

Since p is feasible and $\overline{p}$ is a minimum cost feasible path in terms of f, 2. yields a contradiction. On the other hand 1. cannot hold because it implies that $\tilde{w}(\overline{p}) > \tilde{w}(p)$, which contradicts the assumption that $\tilde{w}(\overline{p})$ is minimum. Therefore we conclude that $\overline{p}$ is an efficient solution of (2), as well as an optimal solution of (1). □

For example, there are three efficient solutions of problem (2) in the network of Fig. 2 with $W=5$, $p_1 = \langle 1,2,\hat{4},5 \rangle$, $p_2 = \langle 1,3,2,\hat{4},5 \rangle$ and $p_3 = \langle 1,\hat{3},2,\hat{4},5 \rangle$, all with objective values $(f(p_1),\tilde{w}(p_1)) = (f(p_2),\tilde{w}(p_2)) = (f(p_3),\tilde{w}(p_3)) = (2,3)$. However, $\langle 1,\hat{3},2 \rangle_D \langle 1,3,2 \rangle_D \langle 1,2 \rangle$. This means that finding all efficient solutions to (2) would require storing dominated labels. Nevertheless, one of those

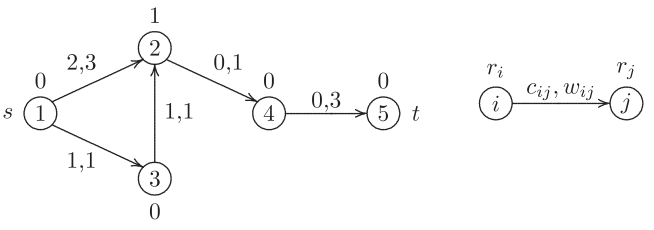

Fig. 2. Network $(\mathcal{N},\mathcal{A})$.

efficient paths, p_3, is in fact formed only by efficient subpaths. Theorem 1 proves that there always exists a path satisfying this condition. As a consequence, the MCPPR can be solved by finding only the non-dominated pairs of objective values, which can be achieved with labeling procedures, despite the constraint and the fact that f is not a standard function.

Theorem 1. *For any non-dominated pair of objective values of* (2)*there is an efficient path formed by efficient subpaths only.*

Proof. Let p^* be an efficient path in $\overline{\mathcal{P}}$ and let i be one of its intermediate nodes. Assume that $\sigma_{p^*}(s,i)$ is dominated by another feasible path from s to i, and let q be an efficient path under those conditions. Thus

1. $f(\sigma_{p^*}(s,i)) \ge f(q)$ and $\tilde{w}(\sigma_{p^*}(s,i)) > \tilde{w}(q)$, or
2. $f(\sigma_{p^*}(s,i)) > f(q)$ and $\tilde{w}(\sigma_{p^*}(s,i)) \ge \tilde{w}(q)$.

Let $q^* = q \diamond \sigma_{p^*}(i,t)$ be the path that results from replacing $\sigma_{p^*}(s,i)$ with q in p^*.

If i is a relay node in p^* then $\tilde{w}(\sigma_{p^*}(s,i)) = 0$, therefore $\tilde{w}(q) = 0$ and $f(\sigma_{p^*}(s,i)) > f(q)$. By assumption q and every subpath of $\sigma_{p^*}(i,t)$ are feasible. Given j and k, the last relay node in q before i and the first relay node in $\sigma_{p^*}(i,t)$ after i, respectively,

$$\tilde{w}(\sigma_{q^*}(j,k)) = w(\sigma_q(j,i)) + w(\sigma_{p^*}(i,k)) = w(\sigma_{p^*}(i,k)),$$

and this does not exceed W because p^* was assumed to be feasible. Therefore q^* dominates p^*, which contradicts the assumption that p^* is efficient.

Otherwise, let k be the first relay node in p^* after i. Then

$$W \ge \tilde{w}(\sigma_{p^*}(s,k)) = \tilde{w}(\sigma_{p^*}(s,i)) + \tilde{w}(\sigma_{p^*}(i,k))$$
$$\ge \tilde{w}(q) + \tilde{w}(\sigma_{p^*}(i,k)) = \tilde{w}(\sigma_{q^*}(s,k)).$$

Thus $\tilde{w}(\sigma_{q^*}(s,k)) \le W$ and q^* is feasible. Besides, $\tilde{w}(q^*) \le \tilde{w}(p^*)$ and, in case 1.

$$f(q^*) = f(q) + f(\sigma_{p^*}(i,t)) \le f(p^*)$$

holds, while in case 2. the inequality is strict.

If condition 2 is satisfied, p^* is dominated by q^*. As for condition 1, either the same conclusion is valid, or else p^* and $q \diamond \sigma_{p^*}(i,t)$ are equivalent, which means there is an efficient path in $\overline{\mathcal{P}}$ equivalent to p^* and formed only by efficient subpaths. □

By this result a set of efficient paths of (2) can be found by means of a labeling algorithm that associates a network node with several different labels, corresponding to paths starting in s. Each label (or path) is identified by a number, stored in a working variable X until it is scanned. Let x be an element of X, then the correspondent label has the form $l_x = [\pi_x^f, \pi_x^w, \xi_x, \beta_x]$, where

- π_x^f denotes the path f value (that depends on the arc and node relay costs),
- π_x^w denotes its weight,
- ξ_x is the index in X that precedes x in the path, and

- β_x is the network node that corresponds to the last node of the path identified by x.

Given $\beta_x = i \in \mathcal{N}$ and $(i,j) \in \mathcal{A}$, if $j \neq t$ then two new elements, y, z, are labeled with

$$l_y = [\pi_x^f + c_{ij} + r_j, 0, x, j] \quad \text{and} \quad l_z = [\pi_x^f + c_{ij}, \pi_x^w + w_{ij}, x, j], \tag{3}$$

where the first case corresponds to extending x by locating a relay at node j. When $j = t$ only the second case has to be considered. The insertion of a new label should be accompanied by a dominance test that compares it with labels previously set for the same node. If the new label is dominated by one of the others, it is discarded; otherwise it is stored as a potential non-dominated label, and any label that is dominated by this one can be deleted. Similar labeling algorithms for bicriteria shortest path problems (thus different objective functions) can be found in [3,9,11,12], among others.

When solving problem (1) some modifications can be introduced to a general labeling algorithm because the goal is now to determine a single optimal solution, which has to be feasible and must have minimum objective value f. First a label l_y, or l_z, is only created if it corresponds to a feasible path, that is, if $\pi_x^w + w_{ij} \leq W$. Besides, since W is a problem input, then assuming W is an integer, without loss of generality, there is at most one label of weight $0, \ldots, W$ associated with each node, where 0 occurs, for instance, when the node is used as a relay. Otherwise W can be replaced with W', the number of possible distinct weight values. This means that checking the dominance of a new label requires $\mathcal{O}(W)$ operations in the worst case. Taking also into account for the fact that in practice W is usually a small value, a different strategy can be used. It aims to simplify the dominance test and replace it with an $\mathcal{O}(1)$ operation for each new label, although it might require dominated labels to be stored. Let M be a $(W+1) \times n$ matrix, where M_{di} stores the index of the label with the best objective value for each feasible weight $d = 0, \ldots, W$ and node $i \in \mathcal{N}$. Given a label l_x that produces two new labels as in (3), l_y is inserted in the set of labels if and only if $\pi_x^f + c_{ij} + r_j < \pi_{M_{0j}}^f$, and l_z is inserted if and only if $\pi_x^f + c_{ij} < \pi_{M_{dj}}^f$, with $d = \pi_x^w + w_{ij}$. Each label replaces the previous labels at M_{0j} and M_{dj}, if they exist. This test can be easily implemented and, since non-dominated labels can be stored in different matrix positions they are no longer deleted. This improves the theoretical complexity order of the algorithm by comparison to a bicriteria shortest path labeling algorithm. For instance, the paths $\langle 1,2,3 \rangle$ and $\langle 1,2,\hat{3} \rangle$ in network $(\mathcal{N}_1, \mathcal{A}_1)$ shown in Fig. 1a correspond to labels $l_a = [8,4,2,3]$ and $l_b = [8,0,2,3]$, respectively. Although $(\pi_b^f, \pi_b^w) D (\pi_a^f, \pi_a^w)$ the weights are different, thus both l_a and l_b are stored by the algorithm. Finally note that in general bicriteria shortest path problems the cost functions range can be unknown in advance or can be too wide to efficiently use such a matrix indexed by the nodes and one of the objective function values.

The pseudo-code of Algorithm 1 summarizes the procedure described for the MCPPR.

Algorithm 1. *Determination of the minimum cost path with relays*

```
1  $M_{dj} \leftarrow 0$, for any $d \in \{0, \ldots, W\}$, $j \in \mathcal{N}$
2  $nX \leftarrow 1$; $l_{nX} \leftarrow [0,0,-,s]$; $X \leftarrow \{1\}$
3  While $X \neq \emptyset$ Do
4    $x \leftarrow$ element in $X$; $Xxlarr; X - \{x\}$; $i \leftarrow \beta_x$
5    Forall $j \in \mathcal{N}$ such that $(i,j) \in \mathcal{A}$ Do
6      $d \leftarrow \pi_x^w + w_{ij}$
7      If $d \leq W$ Then
8        If$(i \neq t)$ Then
9          If$(M_{0\,j} = 0)$ or$(M_{0j} \neq 0$ and $\pi_x^f + c_{ij} + r_j < \pi_{M_{0j}}^f)$
   Then
10           $nX \leftarrow nX + 1$; $l_{nX} \leftarrow [\pi_x^f + c_{ij} + r_j, 0, x, j]$; $M_{0j} \leftarrow nX$
11           $X \leftarrow X \cup \{nX\}$
12         EndIf
13       EndIf
14       If $(M_{dj} = 0)$ or $(M_{dj} \neq 0$ and $\pi_x^f + c_{ij} < \pi_{M_{dj}}^f)$ Then
15         $nX \leftarrow nX + 1$; $l_{nX} \leftarrow [\pi_x^f + c_{ij}, d, x, j]$; $M_{dj} \leftarrow nX$
16         $X \leftarrow X \cup \{nX\}$
17       EndIf
18     EndIf
19   EndFor
20 EndWhile
21 $p^* \leftarrow$ path corresponding to the label in $M_{\cdot t}$ with the best
   value of $\pi^f$
```

The labels stored by Algorithm 1 in column $M_{\cdot t}$ correspond to optimal paths from s to t for every feasible weight, thus one optimal solution of the MCPPR can be found by examining them. The path nodes can be retrieved by following the ξ component of each label backwards, starting at the node that corresponds to t and ending at s.

No rule is imposed for the selection of the next element in X to be scanned. Theorem 2 proves that if the selection is made by non-decreasing order of f, then the elements chosen in X by Algorithm 1 are permanent for the MCPPR, that is, each corresponds to a feasible path, between s and a certain node, with optimal value of f. As a consequence the algorithm can halt as soon as a label associated with node t is chosen in X.

Theorem 2. *If Algorithm 1 selects labels by non-decreasing order of f, then those that have been selected are permanent for the MCPPR.*

Proof. Suppose index x is chosen in X at a step of Algorithm 1 and assume that besides label l_x there is another label for the same network node, l_y, which is feasible and such that $\pi_y^f < \pi_x^f$ will be chosen later on. There are two possibilities:

1. If $y \in X$ when x is selected, then at that moment $\pi_y^f < \pi_x^f$.
2. If y is inserted in X after x has been selected, then its label is obtained from a sequence of feasible labels $l_{y_1}, \ldots, l_{y_k}$ starting at some element y_1, which belongs to X at the time x is selected. Furthermore, since $c_{ij} \geq 0$, $(i,j) \in \mathcal{A}$, and $r_i \geq 0$, $i \in \mathcal{N}$, we have $\pi_{y_1}^f \leq \cdots \leq \pi_{y_k}^f \leq \pi_y^f$, therefore $\pi_{y_1}^f < \pi_x^f$.

Both cases lead to contradiction since x should not have been chosen because it was not the element associated with the best value of f in X. $\square$

Initializing M takes $\mathcal{O}((W+1)n)$ operations (line 1). If the selection in X is made according to Theorem 2, then there are at most $(W+1)\,n$ labels to consider. Each node can have $W+1$ labels to analyze, which implies that each arc in the network is scanned at most $W+1$ times. Since X can contain up to $(W+1)\,n$ elements, selecting the minimum element can be achieved in $\mathcal{O}(\log((W+1)n))$ operations if it is manipulated as a Fibonacci heap (line 4), and this is repeated $\mathcal{O}((W+1)n)$ times. Besides, each attempt to create new labels (lines 8–13 and 14–17) implies $\mathcal{O}(1)$ comparisons and $\mathcal{O}(1)$ insertions. These steps are repeated and $\mathcal{O}((W+1)m)$ operations are performed. Therefore the worst case time complexity bound of Algorithm 1 is $\mathcal{O}((W+1)n + (W+1)m + (W+1)n\log((W+1)n))$, or simply $\mathcal{O}(Wm + Wn\log(\max\{W,n\}))$. In terms of memory space Algorithm 1 requires storing the network, $\mathcal{O}(m)$, as well as the labels produced, $\mathcal{O}((W+1)n)$. Thus the memory requirement is $\mathcal{O}(m+Wn)$.

A variant of Algorithm 1 that eliminates all the dominated labels can also be implemented. In this case each new candidate label should be compared to the former ones associated with the same node, by replacing line 14 in Algorithm 1 with

If $(M_{dj}=0)$ or $(M_{dj}\neq 0$ and $\pi_x^f+c_{ij}<\pi_{M_{kj}}^f$ for some $k\in\{0,\ldots,d\})$ Then

Since there are at most W other labels the worst case complexity of this variant is then $\mathcal{O}(W^2m+Wn\log(\max\{W,n\}))$.

3. The bicriteria minimum cost path problem with relays

The objective function of the MCPPR aggregates the path cost and the relay cost. Considering these objectives separately can provide effective alternative solutions. The BMCPPR is thus a constrained bicriteria problem defined as

$$\min_{p\in\overline{\mathcal{P}}}\{c(p)\},\quad \min_{p\in\overline{\mathcal{P}}}\{r(p)\}. \tag{4}$$

Its aim is to find efficient paths in $\overline{\mathcal{P}}$, that is, feasible paths $p\in\overline{\mathcal{P}}$ such that there is no $q\in\overline{\mathcal{P}}$ with $c(p)\geq c(q)$, $r(p)\geq r(q)$ and at least one inequality is strict. Similarly to problem (1), the solutions of (4) can be found amongst the efficient solutions of the tricriteria problem

$$\min_{p\in\overline{\mathcal{P}}}\{c(p)\},\quad \min_{p\in\overline{\mathcal{P}}}\{r(p)\},\quad \min_{p\in\overline{\mathcal{P}}}\{\tilde{w}(p)\}. \tag{5}$$

Still, function $\tilde{w}$ is not additive and the principle of optimality is again invalid. As a result, a labeling method would have to store intermediate dominated labels. Alternatively, Algorithm 1 can be extended by including in the node labels a new parameter that refers to the path cost. Given an index x associated with a path starting in s, the new labels have the form $l_x=[\pi_x^c,\pi_x^r,\pi_x^w,\xi_x,\beta_x]$, where

- π_x^c, denotes the path cost,
- π_x^r, denotes the cost of the relays in the path.

Taking x as the starting point, new labels for paths associated with the elements y and z are updated as

$$l_y=[\pi_x^c+c_{ij},\pi_x^r+r_j,0,x,j]\quad\text{and}\quad l_z=[\pi_x^c+c_{ij},\pi_x^r,\pi_x^w+w_{ij},x,j], \tag{6}$$

where the first case corresponds to using j as a relay node if $j\neq t$. Similarly to the matrix M in Section 2, a $(W+1)\times n$ matrix L is defined. For every feasible weight $d=0,\ldots,W$ and a node i, L_{di} contains the paths from s to i with weight d, which are not dominated by any other of the same weight. Each position in L is usually associated with more than one path, unlike matrix M.

Given an element x that corresponds to a path from s to $i\in\mathcal{N}$ and $(i,j)\in\mathcal{A}$, the labels in (6) are produced. Then, assuming the associated paths to be feasible, l_y is inserted in set L_{0j} if and only if it contains no other label that dominates it (i.e., another label l_a such that $\pi_a^c\leq\pi_y^c$, $\pi_a^r\leq\pi_y^r$ and at least one of the inequalities is strict), while l_z is inserted in L_{dj}, where $d=\pi_x^w+w_{ij}$, under analogous conditions. The labels in $L_{0\ j}$ and in L_{dj} that are dominated by l_y and by l_z, respectively, can be removed from these sets; otherwise $L_{\cdot j}$ will contain other labels besides the necessary ones. Finally, it is worth stressing that this approach in effect computes the set of efficient paths for every feasible weight $d=0,\ldots,W$. At the end of the process all the efficient BMCPPR solutions, and possibly some dominated ones which should be filtered, are in $L_{\cdot t}$.

The solutions obtained with this labeling method applied to the BMCPPR with $W=5$ in $(\mathcal{N}_1,\mathcal{A}_1)$ are represented as a tree in Fig. 3. After all labels have been scanned the set of candidate solutions is given by $L_{\cdot 4}=\{\langle 1,2,\hat{3},2,4\rangle,\langle 1,\hat{2},4\rangle,\langle 1,\hat{2},3,2,4\rangle\}$. In this set $\langle 1,2,\hat{3},2,4\rangle$ and $\langle 1,\hat{2},4\rangle$ are solutions of BMCPPR, while $\langle 1,\hat{2},3,2,4\rangle$ is dominated by the first path and thus should be discarded in the end.

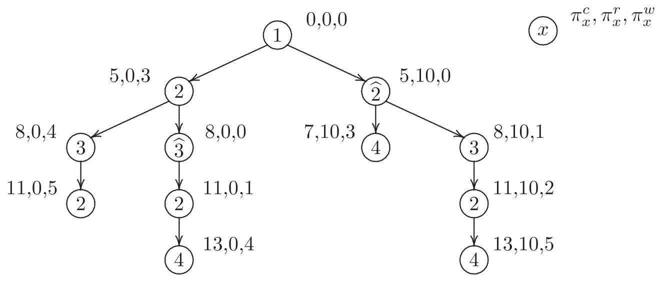

Fig. 3. Tree of the labels produced to find the candidate solutions for the BMCPPR with $W=5$ in $(\mathcal{N}_1,\mathcal{A}_1)$.

The pseudo-code presented in Algorithm 2 outlines the method to solve the BMCPPR.

Algorithm 2. *Determination of the efficient minimum cost paths with relays*

```
1   $L_{dj}\leftarrow\emptyset$, for any $d\in\{0,\ldots,W\}$, $j\in\mathcal{N}$
2   $nX\leftarrow 1$; $l_{nX}\leftarrow[0,0,0,-,s]$; $X\leftarrow\{1\}$
3   While $X\neq\emptyset$ Do
4     $x\leftarrow$ element in $X$; $Xxlarr;X-\{x\}$; $i\leftarrow\beta_x$
5     Forall $j\in\mathcal{N}$ such that $(i,j)\in\mathcal{A}$ Do
6       $d\leftarrow\pi_x^w+w_{ij}$
7       If $d\leq W$ Then
8         If$(i\neq t)$ Then
9           If$((\pi_x^c+c_{ij},\pi_x^r+r_j)$ is not dominated in $L_{0\ j})$ Then
10            $nX\leftarrow nX+1$; $l_{nX}\leftarrow[\pi_x^c+c_{ij},\pi_x^r+r_j,0,x,j]$;
    $L_{0j}\leftarrow L_{0j}\cup\{nX\}$
11            $X\leftarrow X\cup\{nX\}$
12          EndIf
13        EndIf
14        If $((\pi_x^c+c_{ij},\pi_x^r)$ is not dominated in $L_{dj})$ Then
15          $nX\leftarrow nX+1$; $l_{nX}\leftarrow[\pi_x^c+c_{ij},\pi_x^r,d,x,j]$; $L_{dj}\leftarrow L_{dj}\cup\{nX\}$
16          $X\leftarrow X\cup\{nX\}$
17        EndIf
18      EndIf
19    EndFor
20  EndWhile
21  Select the non-dominated labels associated with $t$ in $L_{dt}$,
    $d=0,\ldots,W$
```

Algorithm 2 can be implemented either as a label correcting method, if the selection of nodes in X is arbitrary, or as a label setting method, if the node with the least lexicographic label in terms of c and r is chosen (given the fact that $r_i,c_{ij}\geq 0$, for $i\in\mathcal{N}$, $(i,j)\in\mathcal{A}$). In the second case the labels chosen in X are permanent, that is they are non-dominated in the correspondent set L_{di}, for some $d\in\{0,\ldots,W\}$ and $i\in\mathcal{N}$, and this includes the BMCPPR solutions. The proof is analogous to that of Theorem 2 and is therefore omitted.

As mentioned earlier this approach solves a problem similar to the bicriteria shortest path problem for every feasible weight, which Hansen proved to have a number of efficient solutions that may grow exponentially [10], and thus the BMCPPR is NP-hard.

A different method can be derived from Algorithm 2, modifying it to prune more dominated labels by comparing each new candidate label to the former ones associated with the same node, which can be done by replacing line 14 with

If $((\pi_x^c+c_{ij},\pi_x^r)$ is not dominated in L_{kj}, $k=0,\ldots,d)$ Then

and adding

If $((\pi_x^c+c_{ij},\pi_x^r)$ dominates l_y, with $y\in L_{0j})$ Then $L_{0j}\leftarrow L_{0j}-\{y\}$

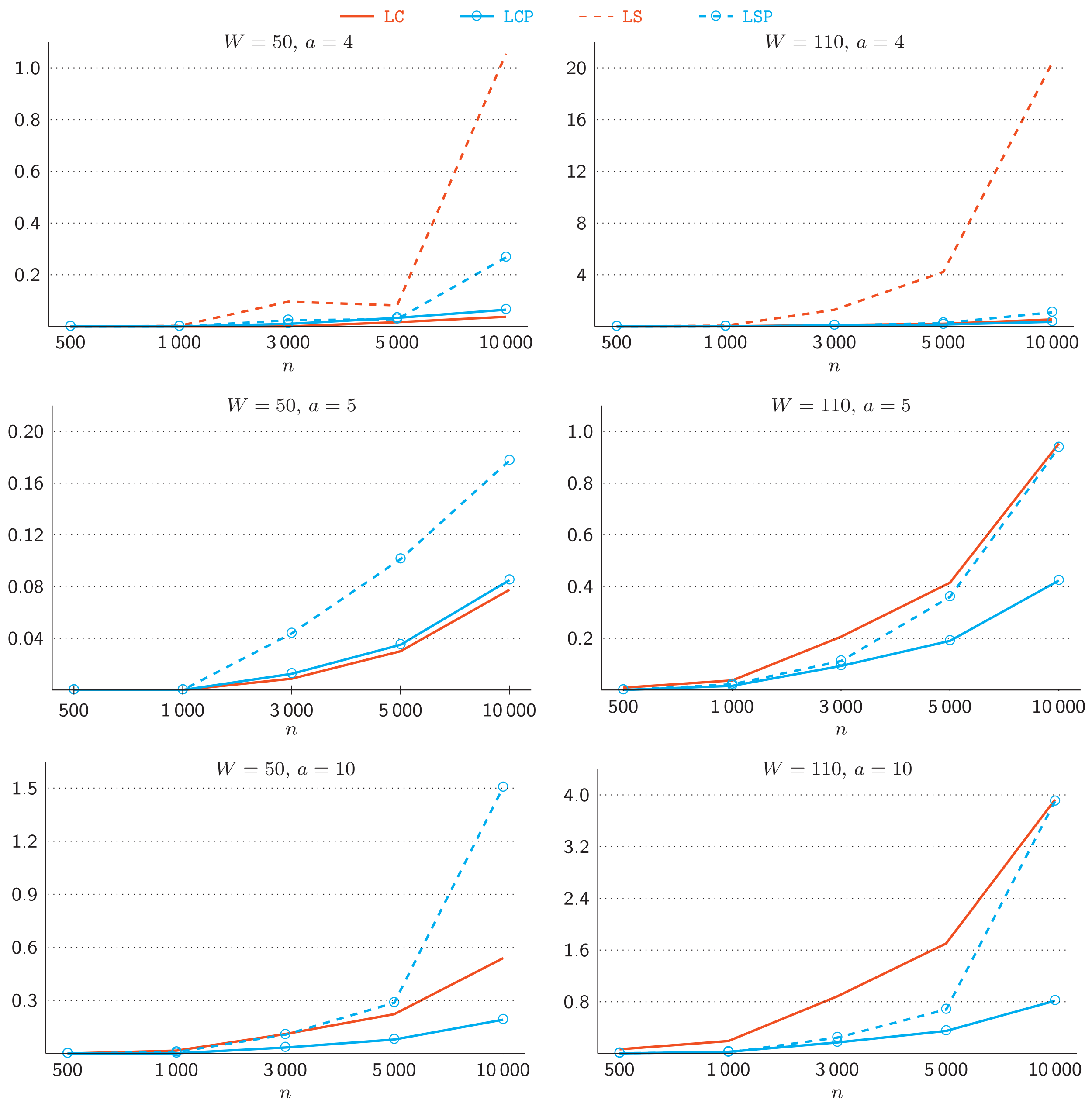

Fig. 4. Average running times (in seconds) of Algorithm 1 versus n.

and

`If` $((\pi^c_x + c_{ij}, \pi^r_x)$ dominates l_y, with $y \in L_{kj}$ and $k = d, \ldots, W)$ `Then` $L_{kj} \leftarrow L_{kj} - \{y\}$

whenever a new label is to be inserted (after lines 9 and 14, respectively). By doing so, only paths with non-dominated images are calculated. Finally, it is worth noting that instead of forming the set of solutions only after the `While` loop, this set can be updated every time a label corresponding to t is selected in X.

4. Computational results

Computational experiments were carried out to evaluate and to compare the empirical performance of the algorithms just described. The tests were run on a Pentium 4 at 3 GHz, with 512 Kb of cache memory and 1 Gb of RAM, over SUSE Linux 10.3. The results presented in the following were obtained on ten different instances generated for each dimension of the data sets.

4.1. The minimum cost path problem with relays

Four versions of Algorithm 1 were coded in C to determine a minimum cost path with relays: two versions of a label correcting algorithm where set X is managed as a FIFO list, and two similar variants of a label setting algorithm which are interrupted when a label associated with t is selected. The two codes written for each type of algorithm correspond to an implementation with no comparison between labels of a node with different weights, as in Algorithm 1, (designated by `LC` and `LS` for the label correcting and the label setting versions, respectively), and another where labels dominated by another one associated with the same node are pruned (`LCP` and `LSP` for label correcting and label setting, respectively).

Table 1
Average number of efficient paths and of labels generated by Algorithm 2.

n	a	# sol.s		# Labels							
				LC		LCP		LS		LSP	
		50[a]	110[b]	50[a]	110[b]	50[a]	110[b]	50[a]	110[b]	50[a]	110[b]
500	4	2	5	41 166	467 586	4 201	14 891	10 841	118 155	3 362	10 274
500	5	5	6	111 581	849 381	7 582	22 820	22 027	212 783	5 525	14 669
500	10	8	7	438 790	1 875 547	20 128	34 917	107 591	387 901	14 669	13 028
1000	4	3	6	1 242 745	1 311 769	9 821	35 422	22 877	274 078	7 503	23 441
1000	5	4	8	194 753	1 940 744	14 710	49 637	42 440	445 977	10 557	31 769
1000	10	6	5	935 237	4 083 538	44 704	71 138	228 306	772 336	28 692	38 807
3000	4	8	8	646 666	5 652 335	38 724	155 585	89 665	1 063 535	29 288	97 999
3000	5	7	9	1 064 827	7 837 033	58 380	192 815	164 612	1 631 856	40 568	118 755
3000	10	11	9	4 297 521	17 382 626	188 200	333 706	897 409	3 332 050	115 401	176 173
5000	4	5	10	1 175 927	10 965 277	72 362	295 381	160 877	2 050 544	52 537	185 976
5000	5	9	10	1 815 686	15 879 506	119 385	353 145	334 428	2 869 595	82 064	206 278
5000	10	10	9	6 757 671	29 162 630	310 720	555 084	1 467 596	5 457 407	189 442	280 901

[a] $W=50$.
[b] $W=110$.

The instances used were random networks with $n=500$, 1000, 3000, 5000, 10 000, and $m=an$ arcs, for $a=4,5,10$. Uniformly integer cost, relay and weight values generated in [1,100] and $W=50,110$ were considered. The plots in Fig. 4 show the average running times of the codes for ten different instances of each data set dimension.

For the considered set of instances the versions with pruning are never worse than the correspondent simpler versions. Moreover, despite the number of labels created by label setting codes being smaller than by the label correcting class, managing, and maintaining them sorted is more complex, and thus the results for that first class of algorithms are worse than those presented by the latter. For these reasons the running times of LS are depicted only for the cases of $a=4$. For a fixed value of W the CPU times grow with instance size, namely with n and with a. The impact of a seems larger than that of n, as times increase faster with the first parameter than with the number of nodes for $W=50$ and $W=110$. In general the relative behavior of the algorithms was similar for both values of W. However, greater values imply a larger matrix and a higher number of labels will be necessary. Therefore the times are clearly greater when $W=110$. The label correcting methods outperformed the correspondent label setting versions for all problem dimensions. It should also be noted that their sensitivity to the increase of the instance parameters was much smaller than for the label setting version. When $W=50$ label correcting was always better than label setting. In contrast, for greater weight values, that produce more labels, it tends to be slower if no pruning is applied. LCP is always the best of the four codes. This program solved the larger instances, $n=10\,000$, $a=10$ and $W=110$, in an average time of 0.816 s.

4.2. *The bicriteria minimum cost path problem with relays*

Four other implementations of Algorithm 2 were coded in C for the bicriteria minimum cost path with relays, and tested on a subset of the previous random networks with $n=500$, 1000, 3000, 5000. The four variants correspond to a label correcting version where the set X is managed as a FIFO list, and a label setting version. For each version two variants were implemented. One performing no pruning of the labels within each set L_{di}, $d=0,\dots,W$, $i\in\mathcal{N}$, and another that compares a new label of node i against all the others in L_{di}, $d=0,\dots,W$. The results in the following are averages for ten instances of each dimension of the data set.

As expected in this case the number of paths produced, and thus the total number of labels generated, was greater than for the MCPPR, which was also reflected by larger running times for the four variants of Algorithm 2. The average numbers of solutions for each problem and the average number of labels generated by the programs are presented in Table 1. For the pruning and no pruning versions, label correcting was less economic than label setting in terms of the memory, as the number of generated labels was always greater with the first two methods.

The same remark can be made concerning the times growth with the number of nodes, which is related to the number of paths that have to be computed and with the total number of generated labels. Fig. 5 depicts the running times obtained by the variants on the test bed. As before pruning dominated labels leads to faster programs and the impact of this option is higher for larger instances. Even though in general label correcting algorithms outperform label setting algorithms, this is not always the case. For this variant of the problem, pruning has greater impact for label correcting than for label setting, namely LS becomes faster than LC. Moreover, when pruning is used there are fewer labels and the process of inserting a new label by lexicographic order in set X becomes more demanding, and thus the running times of LCP grow faster as a function of instance size than those of LSP. In general LCP was the fastest code: it solved instances of the BMCPPR with $n=5000$, $a=10$ and $W=110$, in a maximum average time of 16.497 s.

5. Conclusions

This paper addressed the minimum cost path problem with relays. It introduced an algorithm with a time complexity of $\mathcal{O}(Wm+Wn\log(\max\{W,n\}))$ for the constrained problem with a single objective function. That method is based on a labeling algorithm aided by an auxiliary matrix that stores labels with different feasible weights. A second version of the problem, dealing with the same weight constraint but considering each path and relay costs separately, was also studied. The algorithm proposed for the MCPPR was extended and adapted for this problem. Computational results of label setting and label correcting forms for the two developed algorithms were presented. These results show that the MCPPR can be solved in networks with up to 10 000 nodes and 100 000 arcs in less than 0.816 s. As for the BMCPPR instances with 5000 nodes and 50 000 the sets of efficient paths were computed in about 16.497 s.

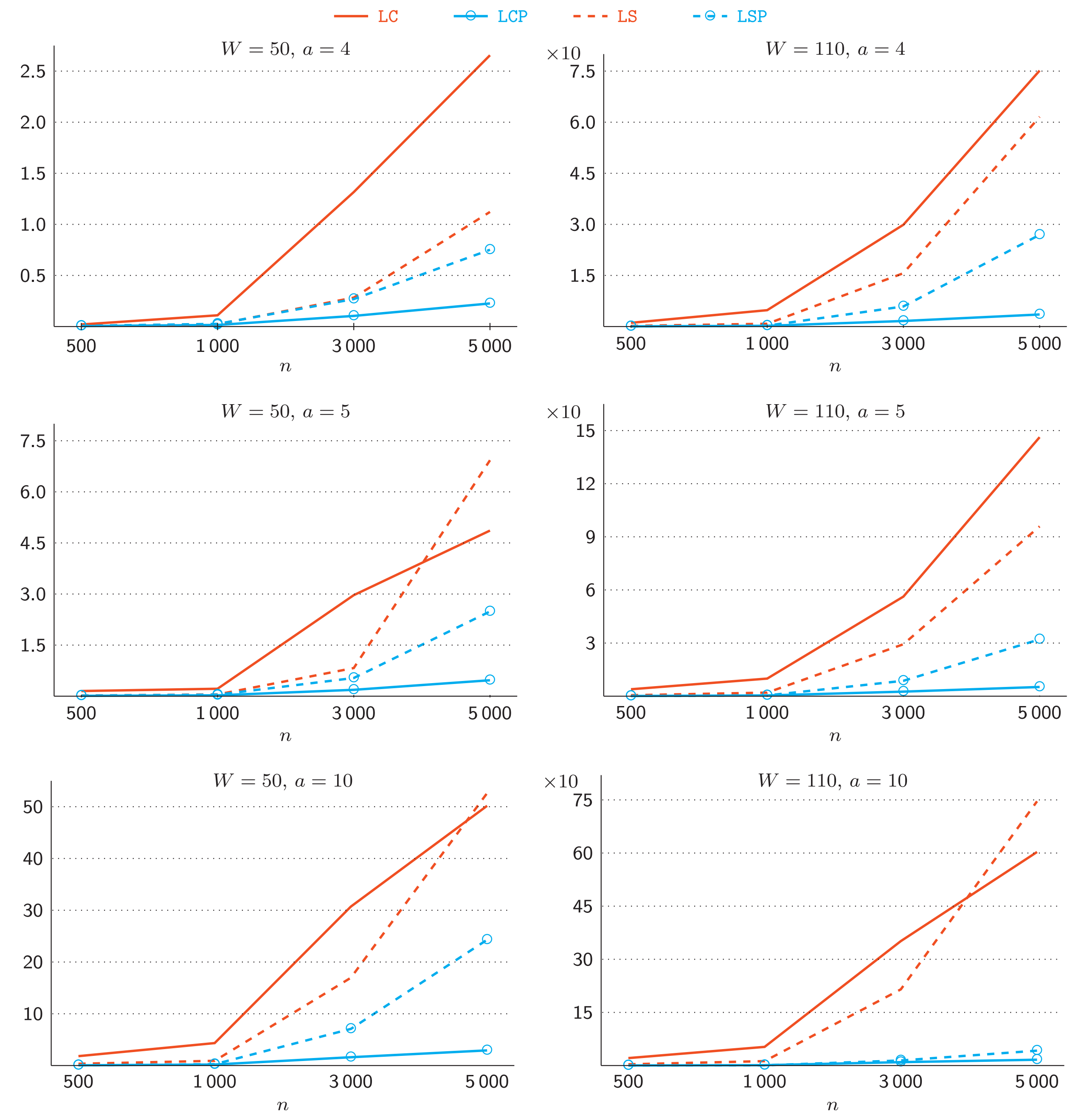

Fig. 5. Average running times (in seconds) of Algorithm 2 versus *n*.

Acknowledgments

This research was partially funded by the Canadian Natural Sciences and Engineering Research Council under grant 39682-05. Part of this work was developed during a visit of Marta Pascoal to the CIRRELT, Montréal, Canada. Her 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 due to the referees for their valuable comments.

References

[1] Bellman R. On the theory of dynamic programming. In: Proceedings of the national academy of sciences of the United States of America. National Academy of Sciences, vol. 38, 1952, p. 716–9.
[2] Bouliane J, Laporte G. Locating postal relay boxes using a set covering algorithm. American Journal of Mathematical and Management Sciences 1992;12:65–74.
[3] Brumbaugh-Smith J, Shier DR. An empirical investigation of some bicriterion shortest path algorithms. European Journal of Operational Research 1989;43:216–24.
[4] Cabral EA. Wide area telecommunication network design: problems and solution algorithms with application to the Alberta SuperNet. PhD thesis, School of Business, University of Alberta, 2005.
[5] Cabral EA, Erkut E, Laporte G, Tjandra SA. The shortest path problem with relays, 2005. [unpublished manuscript].
[6] Cabral EA, Erkut E, Laporte G, Patterson RA. The network design problem with relays. European Journal of Operational Research 2007;180:834–44.
[7] Cabral EA, Erkut E, Laporte G, Patterson RA. Wide area telecommunication network design: application to the Alberta SuperNet. Journal of the Operational Research Society 2008;59:1460–70.
[8] Ghiani G, Laporte G, Semet F. The black and white travelling salesman problem. Operations Research 2006;54:366–78.
[9] Guerriero F, Musmanno R. Label correcting methods to solve multicriteria shortest path problems. Journal of Optimization Theory and Applications 2001;111:589–613.

[10] Hansen P. Bicriterion path problems. In: Fandel G, Gal T, editors. Multiple criteria decision making: theory and applications, Lectures notes in economics and mathematical systems, vol. 177. Berlin: Springer-Verlag; 1980. p. 109–27.

[11] Martins E. On a multicriteria shortest path problem. European Journal of Operational Research 1984;16:236–45.

[12] Skriver A, Andersen K. A label correcting approach for solving bicriterion shortest-path problems. Computers & Operations Research 2000;27:507–24.