Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 07–42

LABELLING METHODS FOR THE GENERAL CASE OF THE MULTI-OBJECTIVE SHORTEST PATH PROBLEM - A COMPUTATIONAL STUDY

JOSÉ MANUEL PAIXÃO AND JOSÉ LUIS SANTOS

ABSTRACT: This paper is devoted to the study of labelling techniques for solving the multi-objective shortest path problem (MSPP) which is an extension of the shortest path problem (SPP) resulting from considering simultaneously more than one cost function (criteria) for the arcs.

The generalization of the well known SPP labelling algorithm for the multi-objective situation is studied in detail and several different versions are considered combining two labelling techniques (setting and correcting), with different data structures and ordering operators.

The computational experience was carried out making use of a large and representative set of test problems, consisting of around 9000 instances, involving three types of network (random, complete and grid) and a reasonable range for the number of criteria.

The computational results show that the labelling algorithm is able to solve large size instances of the MSPP, in a reasonable computing time. The computational experience reported in this paper is complemented by the results presented in a twin paper [22] showing that the label correcting technique proves to be the fastest procedure when the computation of the full set of non-dominated paths is required.

KEYWORDS: Multiple objective programming, combinatorial optimization, shortest path problem, labelling algorithm, non-dominated path.
AMS SUBJECT CLASSIFICATION (2000): 90B10, 90C27, 90C29, 90C35.

1. Introduction

The shortest path problem (SPP) is a well known combinatorial optimization problem intensively studied since the pioneer works published in the middles of the last century [4, 15, 17, 24]. Given a network $\mathcal{G} = (\mathcal{N}, \mathcal{A}, c)$, where $\mathcal{N}$ is the set of nodes (or vertices), $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$ is the set of oriented arcs and c is a cost function defined over $\mathcal{A}$, the SPP consists of finding a path linking, at least cost, two specific nodes of $\mathcal{N}$, say s and t.

Received December 27, 2007.
This work was supported in part by FCT through POCTI - Research Units Pluriannual Funding to CMUC (*Centro de Matemática da Universidade de Coimbra*) and CIO (*Operations Research Center of the University of Lisbon*), and grant POCTI/MAT/139/2001 cofunded by the EU program FEDER.

When c is a k-dimensional vectorial function ($k \geq 2$), one has the multi-objective shortest path problem (MSPP) firstly addressed in the literature in a paper published in 1974, [34]. Until now, most of the work done relatively to the MSPP is focused on the bi-objective case ($k = 2$), where the solutions verify some properties yielding to the development of specific techniques for solving the problem.

The general case has progressively deserved more attention from the researchers, mainly due to the increasing number of real life applications requiring a multi-criteria approach. In fact, the shortest path problem describes real life situations related to transportation and distribution systems where several criteria must be taken into account. In several cases, the MSPP appears as a sub-problem of other models (knapsack problem, set covering problem, project management, etc.) [1, 3, 6, 7, 26].

Most of the algorithms presented in the literature for the MSPP correspond to an extension of the labelling technique known as very efficient for the single objective case.

Let us recall that the labelling method used for the mono-criteria case consists of assigning a label to each node i of the network with the cost of the current best path from s to i. In each iteration of the algorithm, the labels are updated accordingly to one of two distinct techniques known as, respectively, label correcting and label setting. The latter corresponds to scanning the labels in such way that at least one label is made definitive meaning that the least cost path from s to a certain node was found. Using the label correcting technique, any label may be updated until a stopping condition is verified. In [1], the interested reader can find a good description of efficient implementations using both techniques for the SPP.

The generalization of this technique to deal with several criteria must have into account that, in general, there is not a path exhibiting the best value for all the criteria. Hence, for the MSPP, we are looking for non-dominated (ND) paths in the network (also called of Pareto optimal or efficient solutions), that is, paths for which it is not possible to find a better value for one criterion without getting worse for some of the other criteria. As mentioned in [23], the algorithms for solving the MSPP can be grouped into two classes depending on when a specific single solution is required or there is a need of computing all of the non-dominated paths. The first class includes global optimization procedures [7] (where a utility function is defined) and interactive methods [8] (where the user guides the searching within the criteria space). In the second group, one considers the well-known labelling algorithm, [5, 20, 33, 34], the ranking paths procedures, [9, 10], and an algorithm presented by Mote [25] that computes the full set of ND paths, after solving the linear programming relaxation for the MSPP.

Concerning to the labelling approach for the MSPP, part of the published papers describe algorithms based on the label correcting technique [5, 18, 19, 33, 34], while the others present procedures making use of the label setting technique [18, 20, 21]. A particular case occurs in a twin paper of the present one [22], where we describe an algorithm combining the labelling method with deviation path techniques.

As we said before, the several labelling procedures presented in the literature for the MSPP correspond to extensions of the single objective algorithm. Likewise for the SPP, at each iteration of the procedure, both for the label correcting and the label setting technique, a label is selected for scanning in order to create other labels from it. Using the label correcting technique, some authors tested a FIFO (first in, first out) [1], a LIFO (last in, first out) [1], or a DEQue (Double End Queue) [27, 28] policy for selecting the label that will be scanned. When adopting the label setting method, there must be the guarantee that the selected label can be made definitive. As we show later in this paper, that can be achieved by using a utility function in order to assure that the optimality principle holds for the multi-objective case. One example of such a function is associated with the lexicographic order operator used in [18, 20, 21]. In this work, we present and test other operators that can be used for determining the label that will be made definitive.

Again, like for the SPP, the structure used for keeping the set of candidates for scanning, plays an important role in the computational performance of the labelling approach for the MSPP. In this work, we test three different well-known structures (list [1], binary heap [1] and Dial [13, 14] data structures) for keeping the set of candidates ordered and, consequently, selecting the "best" candidate in the label setting versions. For the label correcting versions, we use the FIFO and DEQue data structures.

Now, let us recall that, unlike for the single-objective case, in the MSPP one node may have several labels all of them corresponding to non-dominated paths, even in a temporary basis. Hence, one alternative that may be considered for the MSPP is to adopt a node selection policy. However, confirming the conclusions expressed in previous works [5, 18], our previous computational experience [30], showed that adopting a node selection policy requires a greater implementation effort and, in general, is less efficient than using a label selection policy. Hence, we discarded the node selection policy from the set of implementations tested in this paper for the MSPP.

Although the reasonable amount of work reported in the literature, it is not easy to make a judgment, in terms of computational performance, about the different versions presented for the labelling algorithm applied to the MSPP. In fact, many of the papers above referred just describe the procedures and do not provide, at all, computational results. Others only consider the bi-objective case and report computational experience with randomly generated networks with up to 1000 nodes and 10000 arcs. In the literature, as far as we know, there are only two papers reporting computational results for instances with a number of criteria larger than two [18, 19].

In [19], the author reports a limited computational experience with randomly generated networks (7 instances with a maximum of 100 nodes for $k = 5$ and 500 nodes for $k = 2$). At the time, only three versions for the label correcting technique were tested.

Almost 20 years later, the authors in [18] consider instances with up to 4 criteria for randomly generated networks (up to 40000 nodes and 100000 arcs) and grid networks (up to 625 nodes). That paper compares the computational performance of 7 different versions for the labelling algorithm corresponding to combinations of the label techniques (correcting or setting), the label/node selection policy and the data structure for the set of labels. The conclusions in the referred paper state that the label selection policy is more efficient than the node selection one, and, only for the largest size randomly generated test problems, the label correcting technique outperforms the label setting method.

The remaining works strictly referring to a labelling approach for the MSPP, show computational results only for the bi-objective case. In [5], several versions for the label correcting method with a node selection policy (FIFO, LIFO and DEQue structures), are tested for randomly generated networks with 100 and 250 nodes, and up to 3000 arcs. More recently, the authors in [33] report experience relative to three versions of the label correcting algorithm on much larger size randomly generated networks (1000 nodes/3000 arcs and 500 nodes/7500 arcs).

Additionally, let us refer that computational experiments with a ranking path approach for the bi-objective shortest path problem are reported in

[2, 9, 10]. The largest size instances considered for those tests correspond to randomly generated networks having up to 5000 nodes and 40000 arcs.

Yet for the bi-objective case, a computational study is reported in [25], comparing the label setting algorithm proposed in [20], the algorithm based on a ranking paths procedure described in [9, 10] and the Linear Programming based algorithm presented in the paper. The computational tests involved a small number of instances for two type of networks: randomly generated (up to 1000 nodes and, at most, 10000 arcs), and grid networks (400 nodes).

Finally, in a twin paper of this one [22], we present an algorithm combining the labelling approach with a deviation path technique for the MSPP. There, we compare the computational performance of the new algorithm with the best versions identified in the present work for the labelling algorithm. We tested the algorithms with very large size instances using three type of networks (randomly generated, complete and grid networks). For example, we tried out the algorithms for test problems with $k = 6$ on networks with 15000 nodes and 90000 (randomly generated), 120 nodes (complete networks) and 144 nodes (grid). The largest number of criteria ($k = 10$) was considered for networks with up to 5000 nodes and 30000 arcs (randomly generated) and 100 nodes (complete and grid). Altogether, more than 3500 instances were solved.

In the present paper, we report the extensive computational study carried out in order to identify the best out of 27 different versions of the labelling algorithm for the MSPP. Those versions correspond to combinations of the two label techniques (correcting/setting), with five distinct data structures for the set of candidate labels and four different operators used for ordering the labels. For the computational experiments reported in this paper, we considerer the set of test problems used in [22] plus some larger size instances. Full information about the instances used in this work is available in the following internet access [29]. The computational results showed that all of the largest size instances can be solved in relatively short computing time.

All the details about this computational experience are given in the section 5 of this paper. Before that, the generalization of the labelling algorithm is discussed in section 3 while in section 4, some theoretical results are presented and the correctness of the algorithm is proved. The section 2 contains the mathematical background needed through out the paper and, the final section of the paper summarizes the conclusions.

2. Definitions and basic results

A network is denoted by $\mathcal{G} = (\mathcal{N}, \mathcal{A}, c)$, where $\mathcal{N} = \{1, \ldots, n\}$ is the set of nodes (or vertices) and $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$ is the set of oriented arcs. Let k be the number of criteria, then the vectorial function c attributes a k dimensional vector cost to each arc:

$$\begin{array}{rccl} c: & \mathcal{A} & \longrightarrow & \mathbb{R}^k \\ & (i,j) & \longmapsto & c(i,j) = \mathbf{c_{i,j}} = (c^1_{i,j}, \ldots, c^k_{i,j}). \end{array}$$

A path p, from the vertex i to j, is an alternating sequence of nodes and arcs of the form $p = \langle v_0, a_1, v_1, \ldots, a_r, v_r \rangle$, where:

- $v_\ell \in \mathcal{N}$, $\forall \ell \in \{0, \ldots, r\}$;
- $v_0 = i$ and $v_r = j$;
- $a_\ell = (v_{\ell-1}, v_\ell) \in \mathcal{A}$, $\forall \ell \in \{1, \ldots, r\}$.

By convention, $\langle v_0 \rangle$ is considered as a null path ($r = 0$).

The set of all paths from i to j is denoted by $\mathcal{P}_{i,j}$ and $\mathcal{P}_\mathcal{G}$ represents the set of all paths in the network, that is, $\mathcal{P}_\mathcal{G} = \bigcup_{i,j \in \mathcal{N}} \mathcal{P}_{i,j}$. A cycle is a path with non repeated vertices except the initial and terminal ones which are coincident; that is, $v_0 = v_r$.

With no loss of generality, we consider that $\mathcal{N}$ has an initial node s and a terminal node t such that $\mathcal{P}_{s,t} \neq \emptyset$, $\mathcal{P}_{s,i} \neq \emptyset$ and $\mathcal{P}_{i,t} \neq \emptyset$, for any $i \in \mathcal{N} - \{s, t\}$. In order to simplify the notation, $\mathcal{P}$ will be used instead of $\mathcal{P}_{s,t}$.

Multiple arcs (arcs between the same pair of nodes) are not allowed. As a consequence, p can be denoted only by the sequence of its nodes, $\langle v_0, v_1, \ldots, v_r \rangle$.

A vectorial objective function f is defined over the set all the paths on the network, as follows:

$$\begin{array}{rccl} f: & \mathcal{P}_\mathcal{G} & \longrightarrow & \mathbb{R}^k \\ & p & \longmapsto & f(p) = (f_1(p), \ldots, f_k(p)), \end{array}$$

where $f_\ell(p) = \sum_{(i,j) \in p} c^\ell_{i,j}, \forall \ell \in \{1, \ldots, k\}$.

The concatenation operator, $\diamondsuit$, joins two paths $p = \langle v_0, \ldots, v_{r_p} \rangle$ and $q = \langle u_0, \ldots, u_{r_q} \rangle$ such that $v_{r_p} = u_0$. Then, $p \diamondsuit q = \langle v_0, \ldots, v_{r_p} = u_0, \ldots, u_{r_q} \rangle$.

Now, let us recall that, in order to solve the MSPP, one looks for the set of non-dominated (ND) paths from s to t, mathematically described as follows:

Definition 2.1. *Let p and q be two paths on the network with the same initial and terminal node. One says that $\underline{p \text{ dominates } q}$ or $\underline{q \text{ is dominated by } p}$*

($p <_D q$) if and only if

$$f(p) \neq f(q) \text{ and } f(p) \leq_{\mathbb{R}^k} f(q) \quad (f_\ell(p) \leq f_\ell(q), \ell = 1, 2, \ldots, k).$$

Definition 2.2. *Let p be a path in $\mathcal{P}_{i,j}$, $i, j \in \mathcal{N}$. If there is no path $q \in \mathcal{P}_{i,j}$ such that $q <_D p$, then p is called non-dominated, efficient or Pareto optimal path. The set of ND paths from i to j is denoted by $\bar{\mathcal{D}}_{i,j}$ and $\bar{\mathcal{D}}$ will be used for $\bar{\mathcal{D}}_{s,t}$.*

Now, let us remind that for the single case ($k = 1$), the shortest path problem (SPP) can be easily solved when the *optimality principle* holds, meaning that every shortest path is formed by shortest sub-paths. A necessary and sufficient condition for the SPP to verify the optimality principle is that the network has no cycles with negative cost, [1].

For the multi-objective case, the optimality principle corresponds to saying that every ND path is formed by ND sub-paths. However, for the optimality principle to be verified in this case, only the following sufficient condition is known: for each criterion, the network has no cycle with negative cost (see [21] for a proof and [31] for a counterexample for the reciprocal condition).

3. The labelling algorithm

3.1. Mono-objective shortest path. The labelling algorithm for the mono-objective shortest path problem has been intensively studied and its simplicity and efficiency are considered as strong advantages relatively to other algorithms (see [12]). A general description of the algorithm is given next.

Algorithm 3.1 (Mono-objective Labelling Algorithm).

```
{p_i^*:  best path from s to i found at this moment}
{π_i:    label of i, that is, f(p_i^*)}
{X:      set of "unscanned" nodes}

X ← {s}; π_s ← 0; π_i ← ∞, ∀i ∈ N − {s};
while X ≠ ∅ do
   π_i ← the label of some node i ∈ X
   X ← X\{i}
   for all (i, j) ∈ A do
      if π_i + c_{i,j} < π_j
      then π_j ← π_i + c_{i,j}
           p_j^* ← p_i^* ◇ ⟨i, j⟩
           X ← X ∪ {j}
   end_for all
end_while
```

In few words, the algorithm scans the vertices of a subset $X \subseteq \mathcal{N}$ in order to update the labels assigned to their successors. For $i \in X$, with label π_i, the procedure computes $\pi_i + c_{i,j}$ for each j such that $(i, j) \in \mathcal{A}$. If $\pi_i + c_{i,j} < \pi_j$, then π_j is updated and the vertex j joins the set X.

The label π_i corresponds to the value of the shortest path found, up to the present, from the source s to the vertex i. Hence, one strategy for selecting the vertex of X to be scanned is to pick the one with the less value for the label. This is named in the literature as label setting technique since if the optimality principle holds then each vertex is scanned at most one time. The proof for this, lays on the monotonic property exhibited by the objective function, that is, $f(p\Diamond\langle i, j\rangle) \geq f(p)$; for a complete proof, see [1]. Let us remind that using this technique one may find the shortest path from s to t without having to compute the shortest tree rooted at s.

Alternatively, one may select the vertex to scan just following, for instance, a FIFO policy (first in, first out). In this case, the same vertex may be selected more than once and, therefore, the procedure stops only when the tree constituted by the shortest paths from s to $i \in \mathcal{N}$, is obtained. Using this technique, known in the literature as the label correcting algorithm, one needs to compute the shortest tree rooted at s.

3.2. Data structures for the set of candidates. Now, let us note that the structure adopted for the set X plays a crucial role in the implementation of both versions (setting or correcting) for the labelling algorithm.

In fact, at each iteration, the label setting approach requires the ordering of X in order to pick the node with the smallest label. Several types of data structure have been investigated in literature, for the mono-objective and multi-objective shortest path problem. In this work, we consider the following ones:

- *List* [1]: each element points to the next one in a sequential non-decreasing order for the labels. The node with minimum label is directly accessed but the insertion of a new element forces the searching for its correct position in the list. For removing an element from X, it is very convenient to use a double list structure where in the second list the elements of X are linked by the reverse order.
- *Dial* [13, 14]: the nodes in X are grouped accordingly to the value of the labels; each group is represented by a list or double list. An auxiliary vector $\mathbf{v}$ is used to index the groups, that is, $\mathbf{v}(\ell)$ points to the list of elements with a label equal to ℓ. Considering $M = 1 + \max\{c_{i,j} : (i,j) \in \mathcal{A}\}$, at most M components of $\mathbf{v}$ are not null at each iteration. So, $\mathbf{v}$ is defined as a M-dimensional vector where $\mathbf{v}(\ell)$ refers to the elements with a label equal to ℓ in module M. An auxiliary integer variable *"first"* is also needed to indicate the index of the list with the minimum label for the nodes in X.
- *Heap* [1]: each element (the father) points to two other elements (its sons) in X, with labels greater than or equal to the father. Thus, the set X is partially ordered and the minimum label corresponds to the root of the heap. The insertion of new elements is accomplished in $\mathcal{O}(\log_2(\#X))$ operations, but the deletion needs $\mathcal{O}(\#X)$ operations because one has to find first the element to remove and X is not totally ordered. The removing task can be simplified if an auxiliary n-dimensional vector $\mathbf{v}$ is used, where $\mathbf{v}(i)$ points to the position in the heap corresponding to the node i.

In Figure 1, we give an example for each one of the mentioned data structures while, in Table 1, the respective worst case complexity is presented for the basic operations involving the elements of X.

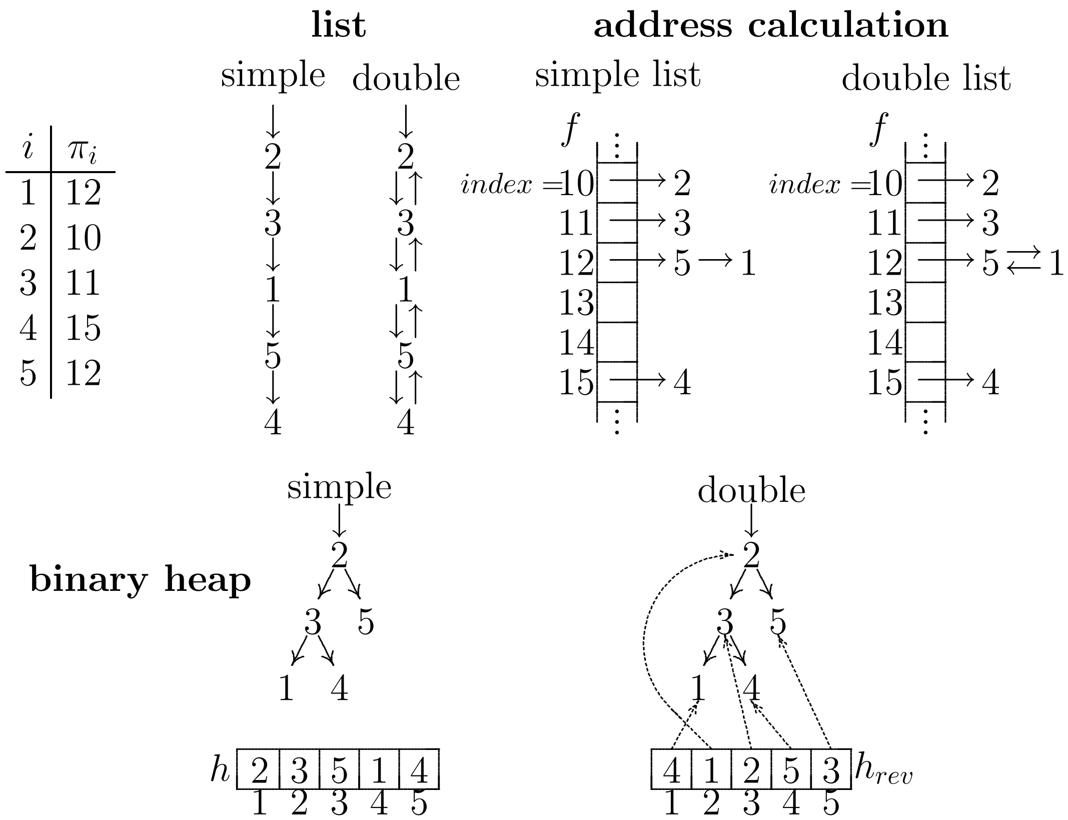

FIGURE 1. **Example of the data structures for** $X = \{1,2,3,4,5\}$ **where** $\pi_1 = 12, \pi_2 = 10, \pi_3 = 11, \pi_4 = 15, \pi_5 = 12$**.**

For the label correcting version of the algorithm, we use a FIFO (First In, First Out) policy for selecting the nodes from X which is represented by a queue where new nodes are added to the tail and the selected node is removed from the head. We also adopted a slightly modification of this structure, proposed by Pape [27, 28], where a new node is inserted at the head, rather than at the tail, if the corresponding label is less than the label for the node currently positioned at the head of the queue.

3.3. Generalization for the multi-objective case. When one wants to generalize the labelling algorithm for the MSPP, some key aspects must be taken into account. Firstly, let us remark that, now, the concept of best path from s to t is not appropriate since we may have several *"best"* paths from s to any vertex. So, for each $i \in \mathcal{N}$, we define a set Π_i containing all the paths from s to i for which, up to the moment, there is no other path dominating

X	find minimum	remove minimum	insert new element	remove element	memory space
not ordered	$\mathcal{O}(\#X)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\#X)$	$\#X$
List (simple)	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\#X)$	$\mathcal{O}(\#X)$	$\#X$
List (double)	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\#X)$	$\mathcal{O}(1)$	$2(\#X)$
Heap (simple)	$\mathcal{O}(1)$	$\mathcal{O}(\log_2(\#X))$	$\mathcal{O}(\log_2(\#X))$	$\mathcal{O}(\#X)$	$\#X$
Heap (double)	$\mathcal{O}(1)$	$\mathcal{O}(\log_2(\#X))$	$\mathcal{O}(\log_2(\#X))$	$\mathcal{O}(\log_2(\#X))$	$2(\#X)$
Dial (simple)	$\mathcal{O}(M)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\#X/M)$	$\#X+M$
Dial (double)	$\mathcal{O}(M)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$2(\#X+M)$

TABLE 1. **Worst case complexity order for basic set operations.**

it. We call Π_i the set of temporarily non-dominated paths from s to i which has to be updated whenever a new s-i path joins it.

A second key aspect is related to the possibility of having, contrarily to the single-objective case, several labels associated to the same vertex. Therefore, in the labelling algorithm for the multi-objective shortest path problem, the set X will consist of unscanned labels rather than nodes, as it was the case for the single-objective case. If we consider a straightforward generalization of the labelling algorithm, one follows a *label selection* policy, that is, at each iteration, a single unscanned label is picked from X and the corresponding path from s to $i \in \mathcal{N}$ is expanded by adding up the arcs $(i,j) \in \mathcal{A}$. An alternative studied in the literature consists of following a *node selection* policy by selecting a node and simultaneously considering all non-dominated labels associated to the node. In this paper, we only consider the label selection policy since the node selection policy requires a greater implementation effort and, in general, is less efficient [5, 18, 30].

When proceeding to the generalization of the Algorithm 3.1 for the multi-objective case, the stop condition is another aspect that requires particular attention. In fact, since the number of ND paths from s to any destination $i \in \mathcal{N}$ is not known in advance, the algorithm stops only when $X = \emptyset$, regardless the label technique (setting/correcting) and the selection policy (label/node) being used. This means that the resolution of the multi-objective shortest path problem, from a source s to a destination t, requires the computation of all ND paths from s to each vertex $i \in \mathcal{N}$.

The Algorithm 3.2 shows a sketch of the multi-objective labelling algorithm, where the logical function $DT(q, \Pi_j)$, performs a *dominance test* on

$q \in \mathcal{P}_{s,j}$ relatively to the temporary ND paths in Π_j. Hence, $DT(q, \Pi_j) =$ TRUE if and only if there is a path $w \in \Pi_j$ for which $w <_D q$.

Finally, let us remind that for the label setting version of the algorithm, the computation of the minimum label in X is required but, this can not be accomplished by using the operator $\leq_{\mathbb{R}^k}$ which is not a total order relation in $\mathbb{R}^k$, [16]. This question will be addressed in the next section.

Algorithm 3.2 (Multi-objective Labelling Algorithm)**.**

{Π_i: `set of temporarily ND paths from` s `to` i}
{X: `set of paths corresponding to "`*unscanned*`" labels`}

$X \leftarrow \{\langle s\rangle\}$; $\Pi_s \leftarrow \{\langle s\rangle\}$; $\Pi_i \leftarrow \emptyset, \forall i \in \mathcal{N} - \{s\}$;
while $X \neq \emptyset$ **do**
    $p \leftarrow$ *a path of* X
    $i \leftarrow$ *the terminal node of* p
    $X \leftarrow X \backslash \{p\}$
    for all $(i,j) \in \mathcal{A}$ **do**
        if $DT(p \diamondsuit \langle i,j\rangle, \Pi_j) =$ FALSE
        then $q \leftarrow p \diamondsuit \langle i,j\rangle$
            $\Pi_j \leftarrow (\Pi_j \cup \{q\}) - \{u \in \Pi_j : q <_D u\}$
            $X \leftarrow X \cup \{q\}$
    end_for all
end_while

4. Determining the minimum label in X

Let us recall that the label setting version of the algorithm is based upon the condition that the label selected, at each iteration, for the scanning procedure will become definitively non-dominated (ND). Next, we will show that this can be guaranteed if one considers an operator R defined over the set $\mathcal{P}_{\mathcal{G}}$ and holding two properties.

So, given an operator R establishing a relation between paths linking the same pair of nodes, one may define an auxiliary function $h_R : \mathcal{P}_{\mathcal{G}} \longrightarrow \mathbb{R}$ and the following binary relations:

- $\leq_R \subset \mathcal{P}_{i,j} \times \mathcal{P}_{i,j}$, such that $p \leq_R q \Leftrightarrow h_R(p) \leq h_R(q)$;
- $<_R \subset \mathcal{P}_{i,j} \times \mathcal{P}_{i,j}$, such that $p <_R q \Leftrightarrow h_R(p) < h_R(q)$;

The Theorem 4.1, stated next, proves the correctness of the label setting algorithm when $<_R$ and $\leq_R$ hold the following properties:

Property 1:[dominance] $p <_D q \Rightarrow p <_R q, \forall p, q \in \mathcal{P}_{i,j}$.
Property 2:[monotonic] $p \leq_R p\diamondsuit\langle j, \ell\rangle, \forall p \in \mathcal{P}_{i,j}, \forall (j, \ell) \in \mathcal{A}(j)$.

Theorem 4.1. *Let $<_R$ and $\leq_R$ be the relations defined over $\mathcal{P}_{i,j}(i, j \in \mathcal{N})$ with an operator R, holding the dominance and monotonic properties. If $p \in X$ is a path selected for scanning by the label setting such that $p \leq_R q$, $\forall q \in X$, then p is a (definitive) ND path.*

Proof: Suppose that $p \in \mathcal{P}_{s,i}$ is a dominated path, then there is a path $w \in \mathcal{P}_{s,i}$ such that $w <_D p$. On the other hand, since $p \in X$, the path w only can be generated after the selection of p, for scanning. This means that w is obtained by an extension of one of the paths in X. That is, $w = w_1\diamondsuit w_2$, with $w_1 \in X$. Therefore, $w_1 \leq_R w <_R p$ contradicting the hypothesis that $p \leq_R q, \forall q \in X$. ■

There are several operators that can be defined over $\mathcal{P}$ for which the properties 1 and 2 are verified. That is the case, for instance, of the lexicographic order operator used in [18, 20, 21] which can be easily accomplished as follows:

$p \leq_{lex} q \Leftrightarrow f(p) = f(q)$ or $\exists x \in \{1, \dots, k\} : f_x(p) < f_x(q)$ and $f_y(p) = f_y(q), \forall y < x$
and $p <_{lex} q \Leftrightarrow f(p) \neq f(q)$ and $p \leq_{lex} q$.

The auxiliary function associated with this operator is given in the next table where the order operators tested in this work are also presented. For h_{lex}, $B = \max\{f_\ell(q) : 1 \leq \ell \leq k, q \in \mathcal{P}_\mathcal{G},$ and q has no cycles$\}$.

R	lex	sum	$norm$	max
h_R	$\sum_{\ell=1}^k f_\ell(p) \times B^{k-\ell}$	$\sum_{\ell=1}^k f_\ell(p)$	$\sum_{\ell=1}^k (f_\ell(p))^2$	$\max_{1\leq\ell\leq k}\{f_\ell(p)\}$

Now, note that the value for the parameter M used in the implementation of a Dial structure for the set of labels depends on the operator that is being used. In fact, for an operator R that parameter should be defined as $M_R = \max\{h_R(p\diamondsuit\langle i, j\rangle) - h_R(p) : i \in \mathcal{N}, p \in \mathcal{P}_{s,i}, (i, j) \in \mathcal{A}\}$. Hence, the next table summarizes the upper-bound values associated to each operator, with $M_c = \max\{c_{i,j}^\ell : (i, j) \in \mathcal{A}$ and $1 \leq \ell \leq k\}$:

R	lex	sum	$norm$	max
M_R	$(B^k - 1) \times M_c + 1$	$k \times M_c + 1$	$2 \times M_c \times (\sum_{\ell=1}^k f_\ell(p)) + kM_c{}^2$	$M_c + 1$

The upper bounds obtained for "$R = lex$" and "$R = norm$" are too large to be computationally practicable. Hence, we opted for taking $M_{lex} = M_c + 1$

while we did not consider the operator "$R = norm$" when using a Dial structure for the label setting algorithm.

Having in mind the expressions for the auxiliary functions considered above, it is clear that if $c^{\ell}_{i,j} \geq 0, \forall (i,j) \in \mathcal{A}$ and $\ell \in \{1,\ldots,k\}$, then $h_R(p \Diamond \langle j,\ell \rangle) \geq h_R(p), \forall p \in \mathcal{P}_{i,j}$ and $\forall (j,\ell) \in \mathcal{A}(j)$. Therefore, as stated in Lemma 4.3, in those conditions the monotonicity property is valid for $R \in \{lex, sum, norm, max\}$. However, that is not sufficient for guaranteeing the dominance property for "$R = max$" as expressed by the following result whose proof is skipped.

Lemma 4.2. *If* $c^{\ell}_{i,j} \geq 0, \forall (i,j) \in \mathcal{A}$ *and* $\ell \in \{1,\ldots,k\}$*, then* $<_{sum}$*,* $<_{norm}$ *and* $<_{lex}$ *hold property 1.*

Lemma 4.3. *If* $c^{\ell}_{i,j} \geq 0, \forall (i,j) \in \mathcal{A}$ *and* $\ell \in \{1,\ldots,k\}$*, then property 2 is verified for* $\leq_{sum}$*,* $\leq_{max}$*,* $\leq_{norm}$ *and* $\leq_{lex}$*.*

Recalling the relation $<_{max}$, note that if p and q are two paths, between the same pair of nodes, with costs $f(p) = (3,5)$ and $f(q) = (4,5)$, $p <_D q$ but $h_{max}(p) = 5 = h_{max}(q)$ regardless the costs for arcs. This example shows that the non-negativity for the costs is not a sufficient condition to guarantee the dominance property for the "$R = max$" operator. But, as stated in the Lemma 4.4, that will be a sufficient condition for the validity of the following properties:

Property 3:[weak dominance] $p <_D q \Rightarrow p \leq_R q, \forall p,q \in \mathcal{P}_{i,j}$.
Property 4:[strictly monotonic] $p <_R p \Diamond \langle j,\ell \rangle, \forall p \in \mathcal{P}_{i,j}, \forall (j,\ell) \in \mathcal{A}(j)$.

Lemma 4.4. *If* $c^{\ell}_{i,j} > 0, \forall (i,j) \in \mathcal{A}$ *and* $\ell \in \{1,\ldots,k\}$*, then* $\leq_R$ *and* $<_R$ *(*$R \in \{lex, max, sum, norm\}$*) satisfy properties 3 and 4, respectively.*

Finally, the Theorem 4.5 proves the correctness of the label setting algorithm when the properties 3 and 4 hold for the sets of paths $\mathcal{P}_{i,j}, i,j \in \mathcal{N}$.

Theorem 4.5. *Let* $\leq_R$ *and* $<_R$ *be the relations defined over* $\mathcal{P}_{i,j}$ *with an operator R, holding the weak dominance and strictly monotonic properties. If* $p \in X$ *is a path selected for scanning by the label setting such that* $p \leq_R q, \forall q \in X$*, then p is a (definitive) ND path.*

Proof: Suppose that $p \in \mathcal{P}_{s,i}$ is a dominated path, then there is a path $w \in \mathcal{P}_{s,i}$ such that $w <_D p$. On the other hand, since $p \in X$, the path w only can be generated after the selection of p, for scanning. This means that w is obtained by an extension of one of the paths in X. That is, $w = w_1 \Diamond w_2$,

Versions	Data structure	R operator			
		lex	max	norm	sum
SETTING	List (simple/double)	X	X	X	X
	Dial (simple/double)	X	X	—	X
	Heap (simple/double)	X	X	X	X
CORRECTING	FIFO	—	—	—	—
	DEQUE	X	X	X	X

TABLE 2. **The 27 tested versions for the labelling.**

with $w_1 \in X$. Therefore, $p \leq_R w_1 <_R w$ contradicting the hypothesis that $w <_D p$ because in this case $w \leq_R p$. ■

A last word in this section to remind that, when using a DEQUE structure for the label correcting version of the algorithm, one may use the operators $R \in \{lex, max, sum, norm\}$ for deciding if the new label will be added at the head or at the tail of the DEQUE.

5. Computational results

In this section, we report the computational experiments carried out in order to evaluate the performance of the labelling algorithm by testing the different combinations summarized in the Table 2, making up a total number of 27 code implementations (22 for the setting version and 5 for the correcting version).

The test instances, shortly described in Table 3, were generated for three different types of network (Random, Complete, and Grid) with the costs for the arcs randomly generated in [1, 1000], using a uniform distribution. For each type of network, we considered variations on the number of nodes and on the number of criteria, producing 12 classes of instances each one of them involving several classes accordingly to the values considered for the parameters (nodes and criteria). Altogether, we considered a total number of 180 groups of instances, 112 classified as "small size" instances and 68 corresponding to large size test problems.

The computational experience was conducted in two stages. In the first stage, we used the small size instances for eliminating the low performance codes. In the second stage, we aimed at identifying the most efficient codes for solving the large size instances.

Now, let us refer that an empirical statistical analysis done by the authors [32] led to the conclusion that a minimum number of 50 instances should be taken as a representative set of computational test cases for assessing the

Class	Network type	Network parameters range	Groups
RandN-small	Random	$n \in \{i * 250 : i \in \mathbb{N}, 1 \leq i \leq 20\}$; $d = 3$; $k = 3$	20
RandK-small	Random	$n = 1000$; $d = 3$; $k \in \{i : i \in \mathbb{N}, 2 \leq i \leq 20\}$	19
CompN-small	Complete	$n \in \{i * 5 : i \in \mathbb{N}, 1 \leq i \leq 20\}$; $d = n - 1$; $k = 3$	20
CompK-small	Complete	$n = 25$; $d = n - 1$; $k \in \{i : i \in \mathbb{N}, 2 \leq i \leq 20\}$	19
GridN-small	Square Grid	$n \in \{i^2 : i \in \mathbb{N}, 5 \leq i \leq 20\}$; $d \approx 4$; $k = 3$	16
GridK-small	Square Grid	$n = 49$; $d \approx 4$; $k =\in \{i : i \in \mathbb{N}, 2 \leq i \leq 20\}$	19
RandN-large	Random	$n \in \{i * 1000 : i \in \mathbb{N}, 1 \leq i \leq 20\}$; $d = 6$; $k = 6$	20
RandK-large	Random	$n = 5000$; $d = 6$; $k \in \{i : i \in \mathbb{N}, 2 \leq i \leq 9\}$	8
CompN-large	Complete	$n \in \{i * 10 : i \in \mathbb{N}, 1 \leq i \leq 14\}$; $d = n - 1$; $k = 6$	14
CompK-large	Complete	$n = 100$; $d = n - 1$; $k \in \{i : i \in \mathbb{N}, 2 \leq i \leq 9\}$	8
GridN-large	Square Grid	$n \in \{i^2 : i \in \mathbb{N}, 5 \leq i \leq 13\}$; $d \approx 4$; $k = 6$	9
GridK-large	Square Grid	$n = 100$; $d \approx 4$; $k =\in \{i : i \in \mathbb{N}, 2 \leq i \leq 10\}$	9

n = number of nodes; d = number of arcs/n; k = number of criteria

TABLE 3. **Set of test instances with the arc costs randomly generated in $[1, 1000]$, using an uniform distribution.**

performance of the MSPP algorithms on a specific type of network with the same size and number of criteria. Hence, we generated 50 different cases for each sub-group of instances, making up a total number of 9050 test problems which are available in the following public database:

http://www.mat.uc.pt/~zeluis/INVESTIG/MSPP/mspp.htm

The Table 4 reports the computational performance of the different versions for the labelling algorithm. The last column of the Table 4 shows the average computational time required by the label correcting version with a FIFO policy for solving the instances included in each class of test problems. Remember that, for each class of test problems, we considered a range of different size instances but, for the sake of simplicity, only total average time is shown in the table. The variation of the CPU time, in seconds, with n and k, is given in the Table 5. For the other versions, the entries correspond to multipliers on the time spent by the FIFO version. That is, for RandN-small instances, the Label Setting version "List" with simple link data structure and the "lexicographic" operator requires, in average, almost 48 times as much the FIFO version does. On the other hand, the Label Setting version "Dial" with double link data structure and the "max" operator is, in average, faster than the FIFO version for CompN-Small instances.

From Table 4, it is clear that FIFO proved to be the most efficient version since the majority of the values shown in the table are greater than

Class	h	List (s)	List (d)	Heap (s)	Heap (d)	Dial (s)	Dial (d)	DEQUE	average CPU FIFO
RandN-small	h_{lex}	47.998	44.257	2.195	2.409	1.284	1.368	1.143	0.016
	h_{norm}	32.687	34.548	1.678	1.934	—	—	1.122	
	h_{sum}	34.776	36.917	1.661	1.937	1.179	1.260	1.085	
	h_{max}	23.589	24.506	1.636	1.890	1.125	1.237	1.075	
RandK-small	h_{lex}	24.854	24.054	1.645	1.746	1.255	1.280	1.094	0.286
	h_{norm}	14.012	14.515	1.373	1.496	—	—	1.089	
	h_{sum}	14.395	14.943	1.359	1.492	1.154	1.180	1.066	
	h_{max}	11.121	11.862	1.369	1.499	1.134	1.172	1.089	
CompN-small	h_{lex}	1.820	1.559	1.370	1.349	1.194	1.173	1.246	0.046
	h_{norm}	1.239	1.236	1.061	1.158	—	—	0.903	
	h_{sum}	1.308	1.313	0.950	1.173	1.002	0.919	1.165	
	h_{max}	1.184	1.157	1.115	1.123	0.953	0.886	0.947	
CompK-small	h_{lex}	2.036	2.003	1.230	1.251	1.172	1.167	1.073	0.652
	h_{norm}	1.697	1.700	1.131	1.206	—	—	1.019	
	h_{sum}	1.754	1.735	1.128	1.209	1.069	1.070	1.012	
	h_{max}	1.587	1.647	1.130	1.182	1.035	1.043	1.015	
GridN-small	h_{lex}	4.183	3.879	1.840	2.172	2.001	1.754	1.603	0.190
	h_{norm}	2.471	2.371	1.084	1.332	—	—	1.059	
	h_{sum}	2.443	2.568	1.203	1.307	1.196	1.269	1.095	
	h_{max}	2.167	2.162	1.143	1.208	1.113	1.187	1.062	
GridK-small	h_{lex}	1.901	1.887	1.362	1.449	1.213	1.274	1.271	0.086
	h_{norm}	1.603	1.637	1.212	1.194	—	—	1.050	
	h_{sum}	1.623	1.574	1.187	1.222	1.186	1.180	1.124	
	h_{max}	1.526	1.549	1.229	1.301	1.124	1.202	1.171	

(s) - simple link data structure; (d) - double link data structure

TABLE 4. **Small size instances - average CPU time ratios relatively to the FIFO version.**

1. In terms of computational efficiency, the other Label Correcting version (DEQUE) comes next, being even faster than FIFO for the CompN-small instances.

n					k				
1000	2000	3000	4000	5000	4	8	12	16	20
0.003	0.007	0.016	0.026	0.043	0.057	0.047	0.194	0.506	1.062
RandN-small					RandK-small				

n					k				
20	40	60	80	100	4	8	12	16	20
0.001	0.008	0.032	0.087	0.182	0.005	0.062	0.301	1.039	3.102
CompN-small					CompK-small				

n					k				
81	169	256	324	400	4	8	12	16	20
0.002	0.034	0.204	0.559	1.328	0.001	0.022	0.068	0.144	0.309
GridN-small					GridK-small				

TABLE 5. **average CPU time consumed by the FIFO version for the small size instances.**

The label setting algorithm is in general less efficient in terms of CPU time consuming than the label correcting one. The worst performance for each class of instances is by far achieved when using a List data structure to represent the set X, regardless the operator. Therefore, this version is not considered in the second stage of the computational experience.

Concerning to the R operators, one can see that the lexicographic operator produces, for each class of instances, the worst results either for the label correcting (DEQUE) version or any of the label setting versions. Consequently, this operator is not used in the second phase computational experiments.

Now, relatively to the Heap and the Dial versions, one can not find a significant difference between the corresponding performances and it seems reasonable to test them with the larger size instances. Finally, let us mention that, for those versions, after excluding the lexicographic operator, the single link data structure versions proves slightly better than the double link structure, for the vast majority of the cases. Hence, we decided to consider only the single link data structure to carry out the further computational testing reducing the number of versions from 27 to 9.

Table 6 presents the results obtained with the reduced set of versions for the large size test problems. The table is organized likewise the Table 4, that is, the last column shows the computational time required by the FIFO version for solving the instances (Table 7 reports the variation of the CPU time, in seconds, with n and k); the other entries in the table are multipliers of that

Class	h	ratio Heap	Dial	DEQUE	av. CPU FIFO
RandN--large	h_{norm}	1.184	—	1.010	33.473
	h_{sum}	1.190	1.087	1.010	
	h_{max}	1.126	1.035	1.002	
RandK--large	h_{norm}	1.197	—	0.987	14.438
	h_{sum}	1.218	1.061	0.989	
	h_{max}	1.145	1.006	0.984	
CompN--large	h_{norm}	1.113	—	0.988	44.812
	h_{sum}	1.154	1.133	0.986	
	h_{max}	1.109	1.141	1.066	
CompK--large	h_{norm}	1.064	—	0.980	126.516
	h_{sum}	1.093	1.071	0.980	
	h_{max}	1.045	1.027	0.974	
GridN--large	h_{norm}	1.493	—	1.109	18.915
	h_{sum}	1.578	1.454	1.218	
	h_{max}	1.716	1.640	1.488	
GrigK--large	h_{norm}	1.561	—	1.110	12.532
	h_{sum}	1.603	1.502	1.104	
	h_{max}	1.797	1.726	1.504	

TABLE 6. **Large size instances - average CPU time ratios relatively to the FIFO version.**

computational time. From those values, one can conclude that all the codes have a similar performance but, the label correcting versions remain as the most efficient ones. In particular, the FIFO algorithm is consistently quite competitive and the label correcting with a DEQUE structure and $R = max$ produced very good results for the random and the complete networks.

Concerning to the operators, let us refer that "max" performs better that "sum" and "norm", for the random and complete networks; the contrary occurs for the grid networks. Relatively to the data structures for the label setting algorithm, it is clear that the Dial structure proves slightly better than the Heap structure.

A final comment on the relationship between the computational times required for solving the test problem and the total number of non-dominated

n				k			
5000	10000	15000	20000	3	5	7	9
9.428	29.151	51.800	75.919	0.451	4.516	18.697	49.224
RandN-large				RandK-large			

n				k			
50	80	110	140	3	5	7	9
0.613	7.752	51.355	300.740	0.212	5.916	95.638	594.135
CompN-large				CompK-large			

n				k			
49	81	121	169	3	5	7	9
0.007	0.218	5.048	131.848	0.004	0.229	3.543	27.850
GridN-large				GridK-large			

TABLE 7. **average CPU time consumed by the FIFO version for the large size instances.**

paths from s to t. In the Table 8, we show relevant information relative to a subset of the instances, concerning to the average number of ND s-t paths and the average number of ND labels, for each group of 50 instances. Note that, despite being interested only in the computation of the ND s-t paths (ND row), one has to compute all of the ND paths from s to every node in the network ($rotND$). From Table 8, one can see that ND is much larger for the grid networks than for the other type of networks but as it has be seen (Table 5 and Table 7), there is not a clear relationship between the computing time and the number of non-dominated paths from s to t. The same applies for $rotND$, where one may have a much larger number of paths with a significantly shorter CPU time (RandN-Large versus CompK-large). As far as our experience shows, one may expect that the computational time increases with ND and $rotND$ but we can not say much about the corresponding rates.

6. Conclusion

In this paper, we described and tested 27 variants of the labelling algorithm for the multi-objective shortest path problem (MSPP). Those different implementations result from considering the two label techniques (setting and correcting), with several combinations for data structure used for the set of unscanned labels and for the ordering operators on the set of paths.

n	1000	2000	3000	4000	5000	5000	10000	15000	20000
ND	5.70	6.12	6.42	7.28	7.34	71.42	82.50	84.80	100.08
$rotND$	5997	12756	21287	28483	36559	351698	820018	1308723	1812431
class	RandN-small					RandN-large			

n	20	40	60	80	100	50	80	110	140
ND	14.46	27.74	37.14	48.60	58.12	265.24	509.86	811.92	1170.34
$rotND$	276	1126	2349	4047	5898	13119	41422	87206	152551
class	CompN-small					CompN-large			

n	81	169	256	324	400	49	81	121	169
ND	85.28	289.04	547.26	869.64	1095.40	329.20	1545.72	6183.24	17622.75
$rotND$	1661	10457	28385	52576	84244	2500	14043	68714	245812
class	GridN-small					GridN-large			

k	4	8	12	16	20	3	5	7	9
ND	8.42	23.52	41.02	59.52	75.30	16.60	50.72	95.74	154.84
$rotND$	9009	22527	37534	53381	69549	81937	242029	479822	789156
class	RandK-small					RandK-large			

k	4	8	12	16	20	3	5	7	9
ND	37.64	170.80	392.08	690.26	1067.68	58.12	369.12	1173.00	2665.04
$rotND$	888	4150	9574	17115	26450	5898	36859	116461	271254
class	CompK-small					CompK-large			

k	4	8	12	16	20	3	5	7	9
ND	99.68	608.48	1024.44	1372.12	1906.40	130.02	1363.92	6139.22	15037.04
$rotND$	1033	3999	6525	9013	12384	2888	16905	53248	108550
class	GridK-small					GridK-large			

TABLE 8. **Average number of non-dominated s-t paths (ND) and average number of non-dominated labels ($rotND$) for the instances.**

The computational experience was carried out making use of a large and representative set of test problems, consisting of around 9000 instances for MSPP, available at a public site. As far as we know, this is the first comprehensive computational study on algorithms for the MSPP, since it involves three types of network (random, complete and grid) and considers a reasonable range for the number of criteria on large size networks.

The computational results show that the labelling algorithm is able to solve large size instances of the MSPP, in a reasonable computing time. The label correcting version, with a FIFO policy for selecting the labels, proved to be consistently efficient. For some test instances, the using of a DEQUE structure for the label correcting produced the fastest performance.

Those results are further confirmed in a twin paper [22] where the best versions for the labelling algorithm are compared with a new algorithm combining the labelling procedure with a deviation path technique. There, the superior comparative performance of the label correcting algorithm is proved when the computation of the full set of ND paths from s to t is required.

References

[1] R.K. Ahuja, T. L. Magnanti, and J.B. Orlin. *Network Flows – theory, algorithms, and applications*. Prentice-Hall, Inc., New Jersey, 1993.

[2] J.A. Azevedo and E.Q. Martins. An algorithm for the multiobjective shortest path problem on acyclic networks. *Investigação Operacional*, 11 (1):52–69, 1991.

[3] R. Batta and S.S. Chiu. Optimal obnoxious paths on a network: Transportation of hazardous materials. *Operations Research*, 36:84–92, 1988.

[4] R.E. Bellman. On a routing problem. *Quarterly Applied Mathematics*, 16:87–90, 1958.

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

[6] M.E. Captivo, J.N. Clímaco, J.R. Figueira, E.Q. Martins, and J.L. Santos. Solving bicriteria 0-1 knapsack problems using a labeling algorithm. *Computers & Operations Research*, 30:1865–1886, 2003.

[7] R.L. Carraway, T.L. Morin, and H. Moskowitz. Generalized dynamic programming for multicriteria optimization. *European Journal of Operational Research*, 44:95–104, 1990.

[8] J.N. Clímaco, C.H. Antunes, and M.J. Alves. Interactive decision support for multiobjective transportation problems. *European Journal of Operational Research*, 65/1:58–67, 1993.

[9] J.N. Clímaco and E.Q. Martins. On the determination of the nondominated paths in a multiobjective network problem. Proceedings of V Sympösium über Operations Research, Köln, (1980), in *Methods in Operations Research*, 40, (Anton Hain, Königstein, 1981), 255–258.

[10] J.N. Clímaco and E.Q. Martins. A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11:399–404, 1982.

[11] C.B. Cunha and J. Swait. New dominance criteria for the generalized permanent labelling algorithm for the shortest path problem with time windows on dense graphs. *International Transactions in Operational Research*, 7:139–157, 2000.

[12] N. Deo and C. Pang. Shortest path algorithms: taxonomy and annotation. *Networks*, 14 (2):275–323, 1984.

[13] R. Dial. Algorithm 360. shortest path forest with topological ordering. *Communications of ACM*, 12:632–633, 1969.

[14] R. Dial, G. Glover, D. Karney, and D. Klingman. A computational analysis of alternative algorithms and labelling techniques for finding shortest path trees. *Networks*, 9:215–348, 1979.

[15] E. Dijkstra. A note on two problems in connection with graphs. *Numerical Mathematics*, 1:395–412, 1959.

[16] M. Ehrgott. *Multiple Criteria Optimization – Classification and Methodology*. Shaker Verlag, Aachen, 1997.

[17] J.L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[18] F. Guerriero and R. Musmanno. Label correcting methods to solve multicriteria shortest path problems. *Journal of Optimization Theory and Applications*, 111:589–613, 2001.
[19] W. Habenicht. Efficient routes in vector-valued graphs. In *Proceedings of the 7th conference on graphtheoretic concepts in computer science (WG 81)*, pages 349–355. Mühlbacher, 1982.
[20] P. Hansen. Bicriterion path problems. in *Multiple Criteria Decision Making: Theory and Application*, editors: G. Fandel and T. Gal, Lectures Notes in Economics and Mathematical Systems, 177, 109-127, Springer Heidelberg, 1980.
[21] E.Q. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16:236–245, 1984.
[22] E.Q. Martins, J.P. Paixão, M.S. Rosa, and J.L. Santos. Ranking multiobjective shortest paths. Working Paper 07-11, CMUC and submitted for publication., 2007.
[23] E.Q. Martins and J.L. Santos. The labelling algorithm for the multiobjective shortest path problem. Internal Technical Report, TR 1999/005, CISUC.
[24] E.F. Moore. The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching*, pages 285–292. Harvard University Press, 1959.
[25] J. Mote, I. Murthy, and D.L. Olson. A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53:81–92, 1991.
[26] I. Murthy and S.S. Her. Solving min-max shortest-path problems on a network. *Naval Research Logistics*, 39:669–683, 1992.
[27] U. Pape. Implementation and efficiency of moore-algorithms for the shortest route problem. *Mathematical Programming*, 7:212–222, 1974.
[28] U. Pape. Algorithm 562: Shortest paths lengths. *ACM Transactions on Mathematical Software*, 6:450–455, 1980.
[29] J.L. Santos. Multiobjective shortest path problem. (http://www.mat.uc.pt/∼zeluis/INVESTIG/MSPP/mspp.htm).
[30] J.L. Santos. O problema do trajecto óptimo multiobjectivo, 1997. (Master degree dissertation; Mathematics Department; University of Coimbra).
[31] J.L. Santos. *Optimização vectorial em redes.* PhD thesis, Departamento de Matemática, Universidade de Coimbra, 2003.
[32] J.L. Santos, J.P. Paixão, and M.S. Rosa. A statistical analysis on the number of non-dominated paths in the multiobjective shortest path problem. International Network Optimization Conference, Lisbon – Portugal, March 21-23, 2005.
[33] A.J. Skriver and K.A. Andersen. A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27:507–524, 2000.
[34] P. Vincke. Problèmes multicritères. *Cahiers du Centre d'Études de Recherche Opérationelle*, 16:425–439, 1974.

José Manuel Paixão
Operations Research Center, Department of Statistics and Operations Research, University of Lisbon
E-mail address: jpaixao@fc.ul.pt

José Luis Santos
Centre for Mathematics of the University of Coimbra, Department of Mathematics, University of Coimbra
E-mail address: zeluis@mat.uc.pt
URL: http://www.mat.uc.pt/∼zeluis