Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 08–27

A NEW RANKING PATH ALGORITHM FOR THE MULTI-OBJECTIVE SHORTEST PATH PROBLEM

JOSÉ MANUEL PAIXÃO AND JOSÉ LUIS SANTOS

Abstract: In this paper, we present a new algorithm for solving the multi-objective shortest path problem (MSPP) which consists of finding all the non-dominated paths between two nodes s and t (ND s-t paths), on a network where a multiple criteria function is defined over the set of arcs. The main feature of the algorithm is that, contrarily to the previous most efficient approaches for the MSPP, not all of the ND sub-paths on the network need to be found. Additionally, the algorithm fully exploits the fact that ND s-t paths are generated at a very early stage of the ranking procedure. The computational experience reported in the paper shows that, for large size general type networks, the new algorithm clearly outperforms the labelling approach.

Keywords: Multiple objective programming, combinatorial optimization, shortest path problem, ranking algorithm, labelling algorithm, non-dominated path.
AMS Subject Classification (2000): 90B10, 90C27, 90C29, 90C35.

1. Introduction

The multi-objective shortest path problem (MSPP) is a generalization of the shortest path problem where a k-dimensional vectorial cost ($k \geq 2$) is assigned to each arc of a network denoted by $G = (N; A; c)$, where $N = \{1, \dots, n\}$ is the set of nodes (or vertices) and $A \subset N \times N$ is the set of arcs. Hence, in the case of the MSPP, one deals with k criteria and aims to find a path linking one node s (source) to another node t (sink), with the minimum value for all of the criteria. However, in general, such an ideal solution does not exist and solving the MSPP turns out to finding non-dominated paths from s to t, i.e. paths for which it is not possible to improve one criterion without worsening another one.

The MSPP was first proposed by Vincke [23] and, later, Hansen [10] proved that, in the worst case, the number of non-dominated paths may increase exponentially with the number of nodes. Some researchers have tried to

Received June 11, 2008.
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.

overcome this obstacle by defining a mono-criterion function that involves all of the criteria as is the case of the weighted sum method (Mote *et al.*, [14]) or the min-max utility function (Murthy and Her, [15]). A different approach is proposed by Clímaco *et al.*, [4], where an interactive searching procedure is presented in order to capture the decision maker preferences.

A natural method for determining the complete set of non-dominated paths consists of generalizing the labelling procedure that has proved to be quite efficient for the mono-criterion shortest path problem. In fact, several algorithms have been presented in the literature following this approach (Hansen [10], Vincke [23], Martins [11], Paixão and Santos [12]).

Let us recall that, for the mono-criterion case, the labelling method consists of assigning a label to each node i of the network with the cost of the incumbent best path from the source s to i (Ahuja *et al.*, [1]). Each iteration, the labels are updated accordingly to one of two techniques known as, respectively, label correcting and label setting. The latter corresponds to scanning the labels in such way that at least one label becomes permanent and the algorithm stops when the node t gains a permanent label. This means that the shortest path from s to t may be obtained without requiring the computation of the shortest path from s to all the other vertices of the network. However, that is no longer true when this technique is adopted for the multi-criteria case. Actually, finding the full set of non-dominated paths from s to t requires the determining of all non-dominated sets from to s to each node of the network.

The labelling algorithm was generalised for the MSPP by Hansen [10] (bi-objective label correcting version), Vincke [23] (bi-objective label setting version) and Martins [11] (label setting version for more than 2 criteria). Briefly, the procedure assigns the zero label to the initial node and, iteratively, expands the search tree from all the outgoing arcs associated with a specific label. This label is selected by following a pre-defined label/node policy and accordingly to the adopted label correcting/setting version. As the number of non-dominated paths is unknown beforehand, the algorithm stops when the search tree cannot be expanded any further. In this case, one has computed the set of non-dominated path from the initial node s to all the other nodes of the network.

Recently, Paixão and Santos [12] proposed a new labelling algorithm based on a deviation path procedure for selecting the label for "expansion" at each iteration. That leads to the speeding-up of the generation of non-dominated paths but the stopping condition for the algorithm could be not improved.

A different kind of algorithm was proposed by Martins [5, 6] for the bi-objective shortest-path problem. Taking into account this specific case, paths from s to t are enumerated by value for the first objective function until the shortest path in the second criteria is obtained using the ranking procedure described in Azevedo *et al.* [2]. Although for this case there is not the requirement for computing the shortest paths from s to every vertex in the network, the algorithm proved to be not competitive since the non-dominance test was performed only at the terminal node t. A generalisation of this technique for $k > 2$ was also done for acyclic networks by Azevedo and Martins [3]. Later, Santos [20, 21] proposed a new upper bound for the stop ranking condition, reducing the number of path to be determined. Deviation path procedures (Martins *et al.* [13]) were also studied to rank paths for the multi-objective shortest path problem, allowing performing the dominance test on the intermediate nodes of the path ([22]).

In Section 3 of this paper, we describe a new algorithm for the general multi-objective shortest path problem based on the label and deviation path algorithm proposed in [12] and consisting of an enhancement on the algorithm proposed in [22]. The definitions and notation required for the comprehension of the contents are given in Section 2 and the computational experience results with the algorithm are shown in Section 4. Final remarks and conclusions are summarized in the last section of the paper.

2. Definitions and notation

A network is denoted by $G = (N, A, c)$, where $N = \{1, \ldots, n\}$ is the set of nodes (or vertices) and $A \subset N \times N$ is the set of arcs. Each arc $a \in A$, $a = (i, j)$, has a tail ($tail(a) = i$) and a head ($head(a) = j$) node. Let k be the number of criteria with the corresponding k dimensional vector cost assigned to each arc:

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

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:

- $v_\ell \in N$, $\forall \ell \in \{0, \ldots, r\}$;
- $v_0 = i$ and $v_r = j$;

- $a_\ell = (v_{\ell-1}, v_\ell) \in A$, $\forall \ell \in \{1, \ldots, r\}$.

The set of all paths from i to j is denoted by $P_{i,j}$, and P_G represents the set of all paths in the network, that is $P_G = \bigcup_{i,j \in N} P_{i,j}$. 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 N has an initial node s and a terminal node t such that:

- for any arc $a \in A$, $tail(a) \neq t$ and $head(a) \neq s$;
- for any $i \in N - \{s, t\}$, $P_{s,i} \neq \emptyset$ and $P_{i,t} \neq \emptyset$.

In order to simplify the notation, P will be used instead of $P_{s,t}$. Multiple arcs (arcs with the same pair of head and tail nodes) are not allowed. As a consequence, a path p can be simply denoted by the sequence of its nodes, $\langle v_0, v_1, \ldots, v_r \rangle$. The vectorial objective function f is defined by

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

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

Now, let us recall that, for the MSPP, one looks for the set of non-dominated paths from s to t defined as follows:

Definition 1. : *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be two elements of* $\mathbb{R}^k$*. Then,* $\mathbf{a} \leq_{\mathbb{R}^k} \mathbf{b}$ *(*$\mathbf{a}$ *is less than or equal to* $\mathbf{b}$*) if and only if*

$$a_\ell \leq b_\ell, \forall \ell \in \{1, \ldots, k\}.$$

Definition 2. : *Let p and q be two paths of $P_{i,j}$. We say that <u>p dominates q</u> or <u>q is dominated by p</u> ($p <_D q$) if and only if*

$$f(p) \neq f(q) \text{ and } f(p) \leq_{\mathbb{R}^k} f(q).$$

Definition 3. : *Let p be a path in $P_{i,j}$, $i, j \in N$. If there is no path $q \in P_{i,j}$ such that $q <_D p$, then p is called a <u>non-dominated</u> (<u>efficient</u> or <u>Pareto optimal</u>) path. The set of non-dominated paths from i to j is denoted by $\bar{D}_{i,j}$ and $\bar{D}$ will be used for $\bar{D}_{s,t}$. We will use $\bar{D}_s$ to denote $\bigcup_{i \in N \setminus \{s\}} \bar{D}_{s,i}$.*

Note that $\leq_{\mathbb{R}^k}$ is not a total order relation in $\mathbb{R}^k$ and, therefore, does not allow the full ranking of the paths in the network. Nevertheless, this may be achieved by considering a total order relation [9] as the following one:

Definition 4. : *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be two elements of* $\mathbb{R}^k$. *Then,* $\mathbf{a}$ *is lexicographically less than or equal to* $\mathbf{b}$ *(*$\mathbf{a} \leq_{lex} \mathbf{b}$*) if and only if*

$$\mathbf{a} = \mathbf{b} \text{ or } (\exists x \in \{1, \ldots, k\} : a_x < b_x \text{ and } a_y = b_y, \forall y < x).$$

Let us remark that $\leq_{lex}$ yields the setting up of preference levels amongst the criteria. For instance, considering the lexicographic order means that top preference is given the first objective; only in the case of a tie, the second criterion is used for the ranking, and so forth. Also, note that a total of $k!$ lexicographic order relations can be defined by permuting the priority levels with the corresponding of the paths defined as follows:

Definition 5. : *Let* $\mathbf{a}, \mathbf{b}$ *be two elements of* $\mathbb{R}^k$ *and* $\Delta = (\delta_1, \ldots, \delta_k)$ *a permutation of the elements in* $\{1, \ldots, k\}$*, where* δ_i *indicates the* i*-th criterion to be analyzed in the lexicographic order. Then,* $\mathbf{a}$ *is* Δ*-lexicographically less than or equal to* $\mathbf{b}$ *(*$\mathbf{a} \leq_{\Delta-lex} \mathbf{b}$*) if and only if*

$$\mathbf{a} = \mathbf{b} \text{ or } (\exists x \in \{1, \ldots, k\} : a_{\delta_x} < b_{\delta_x} \text{ and } a_{\delta_y} = b_{\delta_y}, \forall y < x).$$

Definition 6. : *Let* p *and* q *be two paths of* $P_{i,j}$ *and* $\Delta = (\delta_1, \ldots, \delta_k)$ *a permutation of the elements in* $\{1, \ldots, k\}$*. Then,*

$$p \leq_{\Delta-lex} q \Leftrightarrow f(p) \leq_{\Delta-lex} f(q).$$

The following Lemma will be crucial for the correctness of the algorithm.

Lemma 1. : *Let* $\Delta = (\delta_1, \ldots, \delta_k)$ *be a permutation of the elements in* $\{1, \ldots, k\}$*. If* $p, q \in P_{i,j}$ *and* $p <_D q$*, then* $p \leq_{\Delta-lex} q$*.*

Proof: By definition of $p <_D q$, $f(p) \neq f(q)$ and $f_\ell(p) \leq f_\ell(q), \forall \ell \in \{1, \ldots, k\}$. On the other hand, considering a permutation Δ and defining $\ell' \in \{1, \ldots, k\}$ as the first index for which $f_{\delta_{\ell'}}(p) \neq f_{\delta_{\ell'}}(q)$ we obtain $f_{\delta_{\ell'}}(p) < f_{\delta_{\ell'}}(q)$. Consequently, $p \leq_{\Delta-lex} q$. □

Note that, the resulting sequence of paths obtained with a ranking procedure using an order $\leq_{\Delta-lex}$ is similar for permutations Δ and Δ' where $\delta_1 = \delta'_1$. Thus, we define the lexicographic order in the r-th objective ($\leq_{lex,r}$) as a Δ-lexicographic order where $\delta_1 = r$.

3. The new algorithm

As mentioned earlier in this paper, the weak point of the labelling algorithm for the MSPP is related to the need of computing the full set $\bar{D}_s$ in order to know the set $\bar{D}$ of the non-dominated paths from s to t. In a previous work, we show that a significant portion of $\bar{D}$ may be found by ranking only a few number of s-t paths. In this paper, we combine the ranking path and the labelling strategies using the L&DP algorithm [12] for each criterion with a stop ranking condition studied in [22].

The rationale for the new algorithm is the following: if p is a path from s to t, then for every path $q \in \bar{D}$ there exists at least one criterion $i \in \{1, 2, \dots, k\}$ such that $f_i(q) \leq f_i(p)$. Now, suppose that one ranks the s-t paths by the corresponding value for criterion i. It is clear that the ranking procedure, for that criterion, should stop whenever such value becomes greater than $f_i(p)$.

Hence, the stop ranking condition can be defined by a vector $v \in \mathbb{R}^k$ where the v_i component establishes an upper bound for the value of f_i in the ranking procedure for the i criterion. This vector may be computed before carrying out the ranking procedure (sequential version) or can be dynamically computed during the ranking (parallel version). Next, we describe both versions where p^i_j means the j-th s-t path found by the L&DP procedure when $\leq_{lex,i}$ is used and W^i denotes the set of ND paths from s to t computed by the L&DP procedure ($i \in \{1, \dots, k\}$). In this work, we have used the permutation $\Delta = (r, 1, \dots, r-1, r+1, \dots, k)$ for the lexicographic order in the r-th objective (i.e., $\leq_{lex,i}$).

3.1. Sequential version. As said above, in this version, v is computed before starting the ranking procedure as the objective value for a particular ND s-t path. In this work, we use the path, denoted by p^*_{sum}, that minimizes $\sum_{i=1}^{k} f_i(q)$ over P which can be easily computed by a mono-criteria shortest path algorithm. Consequently, any s-t path p for which $f(p) \in S = \{w \in \mathbb{R}^k : v \leq_{\mathbb{R}^k} w\} \backslash \{v\}$ is dominated by p^*_{sum}. Therefore, we only need to determine paths p where $f(p) \notin S$; i.e. for which $f_i(p) < v_i$, for some $i \in \{1; \dots; k\}$. Additionally, we also need to find path which $f(p) = v$, if one aims to find all of the paths with a value less than or equal to $f(p * sum)$.

In Algorithm 1, we report the pseudocode for the sequential version of the new algorithm for finding all the paths verifying $f_i(p) \leq v_i$, for some $i \in \{1; \dots; k\}$. Basically, the procedure iterates over the k criteria determining for each one of them the set of paths W^i. Note, now, that some ND dominated paths can be found for more than one criterion. In order to avoid repetition of those paths, the step 3.5 restricts the inclusion in $\bar{D}$ to the paths p found in the i-th ranking for which $f_\ell(p) > v_\ell, \forall \ell < i$.

step 1:	set $v \in \mathbb{R}^k$ for the stop ranking condition
step 2:	$\bar{D} \longleftarrow \emptyset$
step 3:	for $i = 1$ to k do
step 3.1:	$j \longleftarrow 0$
step 3.2:	repeat
step 3.3:	$j \longleftarrow j + 1$
step 3.4:	Compute p^i_j using $\leq_{lex,i}$
	until $(f_i(p^i_j) > v_i)$
step 3.5:	$\bar{D} \longleftarrow \bar{D} \cup \{p \in W^i : f_i(p) \leq v_i \text{ and } f_\ell(p) > v_\ell, \forall \ell < i\}$
	endfor

Algorithm 1: sequential version of the new algorithm.

The next results validate the Algorithm 1 and show that $v = f(p)$ for any $p \in \bar{D}$ is a feasible choice for the stop ranking condition set at step 1.

Lemma 2. : *Let p be a path from s to t in G. Then, every path $q \in \bar{D}$ verifies $f_r(q) \leq f_r(p)$ for some $r \in \{1, \ldots, k\}$.*

Proof: If the statement is not true, there will exist a path $q \in \bar{D}$ such that $f_r(q) > f_r(p)$ at all criteria. Consequently, $p <_D q$ which contradicts the hypothesis. □

Corollary 1. : *At the end of the algorithm 1, $\bar{D} = \bigcup_{i=1}^{k} W^i$.*

Proof: Let p be the path from s to t chosen by Algorithm 1 and let $q \in \bar{D}$. Lemma 2 asserts that there is, at least, one $r \in \{1, \ldots, k\}$ for which $f_r(q) \leq f_r(p)$ and then $q \in W^r$. On the other hand, if there is a path $p \in W^r$, for some $r \in \{1, \ldots, k\}$, such that $p \notin \bar{D}$ then there will be a path q dominating p. Therefore, $f_r(q) \leq f_r(p)$ and q should be found in the ranking procedure using $\leq_{lex,i}$ contradicting the fact of $p \in W^i$. □

3.2. Parallel version. In the parallel version, each component v_i of v is updated during the ranking procedure with the largest value of $f_i(p)$ over the set of s-t paths that have been ranked up to that moment by the i-th criterion. Consequently, v usually does not correspond to the f value for a s-t path but, on the other hand, one has $v \leq_{\mathbb{R}^k} f(p)$ for all unranked path. That means that v is a lower bound in the space of the criteria for the value of f over all s-t paths that have not been determined up to that instant. Therefore, if at some step of the algorithm we find a ND s-t path q satisfying

$f(q) \neq v$ and $f(q) \leq_{\mathbb{R}^k} v$, then all the unranked path will be dominated and the algorithm stops. This version of the new algorithm is summarised in Algorithm 2. Here, steps 4 and 5 avoid the repetition of ND paths computed on different rankings.

step 1:	$v \in 0_{\mathbb{R}^k}$
step 2:	$j \longleftarrow 0$
step 2:	allNDfound $\longleftarrow$ FALSE
step 3:	repeat
step 3.1:	$j \longleftarrow j+1$
step 3.2:	for $i = 1$ to k do
step 3.3:	Compute p_j^i using $\leq_{lex,i}$
step 3.4:	$v_i \longleftarrow f_i(p_j^i)$
step 3.5:	if $\exists q \in W^i : f(q) \neq v$ and $f(q) \leq_{\mathbb{R}^k} v$
step 3.6:	allNDfound $\longleftarrow$ TRUE
	endfor
	until allNDfound = TRUE
step 4:	$\bar{D} \longleftarrow \emptyset$
step 5:	for $i = 1$ to k do
step 5.1:	$\bar{D} \longleftarrow \bar{D} \cup \{p \in W^i : f_i(p) \leq v_i$ and $f_\ell(p) > v_\ell, \forall \ell < i\}$
	endfor

Algorithm 2: parallel version of the new algorithm.

The correctness of the Algorithm 2 is assured by the next results.

Lemma 3. : *Let v be a vector of $\mathbb{R}^k$ for which the set $S = \{w \in \mathbb{R}^k : v \leq_{\mathbb{R}^k} w\}$ contain the cost of all unranked s-t paths. If there is a s-t path p for which $f(p)$ dominates v then all ND s-t paths have be obtained by the algorithm.*

Proof: Let p be a path from s to t for which $f(p)$ dominates v. Then, $f(p) \neq v$ and $f(p) \leq_{\mathbb{R}^k} v \leq_{\mathbb{R}^k} f(q)$ for all paths q verifying $f(q) \in S$. Consequently, p dominates all unranked s-t paths. □

Corollary 2. : *At the end of the algorithm 2, $\bar{D} = \bigcup_{i=1}^k W^i$.*

Proof: Let q be a ND s-t path. By lemma 3, at the end of the algorithm 2, q was determined. Consequently, $f(q) \notin S$, i.e., there is, at least, one $r \in \{1, \ldots, k\}$ for which $f_r(q) \leq f_r(p)$ and then $q \in W^r$. On the other hand, similarly as was proved in Corollary 1, $\bigcup_{i=1}^k W^i \subseteq \bar{D}$. □

Finally, let us mention that the L&DP algorithm was carried out with other orders different from the lexicographic one obtaining a better performance [12]. However, it can be proved that only the lexicographic order is able to define an upper bound for the ranking procedure [22].

4. Computational experience

In this section, we report the computational experiments carried out in order to assess the performance of the new algorithm proposed in this work. As benchmark code, we used the public version of the label correcting algorithm (with fifo data structure) available at [19]. The most efficient versions for the labelling algorithm [16] were also considered for the assessment of the new method. The computational results were obtained by running the codes on a Intel(R) Pentium(R) 4 CPU 3.00GHz personal computer with 512 MB RAM and 1 MB cache size at the Laboratory for Computational Mathematics of Centre for Mathematics of the University of Coimbra. The codes are written in C language and compiled using the "cc" compiler of Linux system (Suse 9.3 version) without any optimization option.

The two versions of the new algorithm (*newP* - parallel version, *newS* - sequential version) were implemented using a heap [1] and a Dial [7, 8] data structure. For the label setting algorithm, we ran the codes reported in [12] as having a better performance: the labelling and deviation path algorithm (*L&DP*) and the label setting algorithm (*LS*), both using a heap and Dial data structure. We also considered the classical versions of the label correcting (*LC*) algorithm - with *deque* [17, 18] and *fifo* [1] data structures - with a better performance based on the results presented in [12, 16].

As summarised in the Table 1, a total number of 11 codes were tested for a set of large size instances for three different types of network (random, complete and grid) with the costs for the arcs randomly generated in the interval [1,1000] using a uniform distribution. Table 2 reports the main features for each class of problem defined by the type of network and the corresponding dimensions - number of vertices (n), density (d=ratio between the number of arcs and the number of nodes) and number of criteria (k). Note that we tried out 50 instances for each class of problems and columns *ND* (number of non dominated paths) and *rotND* (number of non dominated labels) show average values for those indicators. Also, the average CPU time (in seconds of a) consumed by the public code for solving each instance within each class of problem, is shown in the last column of Table 2. The

	newP	newS	L&DP	LS	LC	public
heap	X	X	X	X	—	—
Dial	X	X	X	X	—	—
deque	—	—	—	—	X	—
fifo	—	—	—	—	X	X

TABLE 1. Description of codes used in the computational experience.

name	instances	n	d	k	ND	$rotND$	CPU (sec)
RandN	50	15000	6	6	84.80	1308608.7	59.23
RandD	50	5000	10	6	153.04	814576.7	87.45
RandK	50	5000	6	10	191.52	969339.6	82.15
CompN	50	120	119	6	951.82	107122.7	116.31
CompK	50	100	99	8	1790.24	183120.9	312.61
GridN	50	144	4	6	11702.90	141228.3	34.59
GridK	50	100	4	10	20797.18	141412.7	77.69

TABLE 2. Description of the set of instances solved. The last column corresponds to the CPU running time for the public code.

instances tested in this work are the largest ones considered in [12] and available on-line at [19].

Now, let us remark that we were able to solve all of the instances with all of codes but the parallel version of the new algorithm. Actually, due to exceeding a time limit of 600 secs or to producing more than 2×10^6 labels, *newP* could not produce a solution for almost half of the test instances for classes CompK and GridK, and for a much more reduced percentage (10%) of the class GridN instances.

Table 3 reports, for each code, the ratio relatively to the public code, in terms of computing time required for solving the instances. This ratio indicates how many times a code is faster (ratio < 1) or slower (ratio > 1) than the public code. From that table one can conclude that the parallel version of the new algorithm is not competitive with the sequential one. In addition, the sequential version of the new algorithm is much faster than the labelling ones - 20 to 50 folds faster than the public code for the random type networks; 4 to 5 folds faster for the complete networks. On the other hand,

	newP		newS		L&DP		LS		LC	
name	heap	Dial	heap	Dial	heap	Dial	heap	Dial	deque	fifo
RandN	0.05	0.05	0.02	0.02	0.93	1.14	0.95	0.87	0.86	0.86
RandD	0.20	0.19	0.04	0.04	1.03	1.23	0.95	0.92	1.00	0.99
RandK	0.07	0.07	0.01	0.01	0.92	0.89	0.94	0.90	0.88	0.88
CompN	3.73	3.07	0.20	0.21	0.97	1.18	0.90	0.87	0.87	0.87
CompK	2.31	2.29	0.17	0.18	0.88	1.03	0.99	0.97	0.86	0.84
GridN	10.46	10.38	3.22	3.31	3.46	2.55	2.46	2.36	1.21	0.86
GridK	8.56	8.37	2.01	2.04	2.46	1.94	1.88	1.83	0.98	0.87

TABLE 3. Average CPU time ratio for each algorithm relatively to the public code.

	newP		newS		L&DP		LS		LC	
name	heap	Dial	heap	Dial	heap	Dial	heap	Dial	deque	fifo
RandN	361363.9	361363.9	92682.7	92682.7	1373296.3	2350161.6	1317932.7	1317942.8	1326010.4	1334770.2
RandD	823155.0	823155.0	246796.8	246796.8	867450.2	1296013.3	822790.9	822801.4	828686.8	835634.7
RandK	510287.4	510287.4	99344.2	99344.2	987788.8	1420508.5	970385.9	970387.8	970263.4	970795.8
CompN	526561.7	526561.7	234800.3	234800.3	120366.5	112856.8	109665.8	109670.2	111887.6	113597.8
CompK	861422.1	860323.2	349303.4	349303.4	192938.3	189921.3	184245.2	184247.2	185417.3	186155.3
GridN	569576.3	569576.3	392115.9	392115.9	162467.1	156671.2	143655.8	143656.1	149337.2	155554.9
GridK	815283.5	806314.8	503787.9	503787.9	149309.0	146340.7	142065.2	142065.3	144417.7	147013.9

TABLE 4. Average number of labels computed.

the new algorithm is slower than the labelling one (2 to 3 times slower than the public code) for the grid network instances.

Finally, Table 4 shows the average number of labels determined by each code to solve the instances proposed. From there, one may conclude that the stop ranking condition in the new algorithm is more effective for the sequential version than for the parallel one (reflecting the higher performance of the sequential version). Moreover, we can also see that the stop ranking condition allows to avoid the computation of a large number of labels that have to be determined by the labelling algorithm, when dealing with random or complete networks. On the other hand, the opposite scenario occurs on grid networks.

It should be noted that all versions of the labelling algorithm are very effective in the computation of $\bar{D}_s$. In fact, more than 90% of the total number of labels generated by the labelling algorithm are ND ones (confront Table 4 with the $rotND$ column in Table 2). However, as the number of ND s-t paths is unknown beforehand the labelling algorithm has to determine the whole set $\bar{D}_s$ to make sure that $\bar{D}$ is obtained.

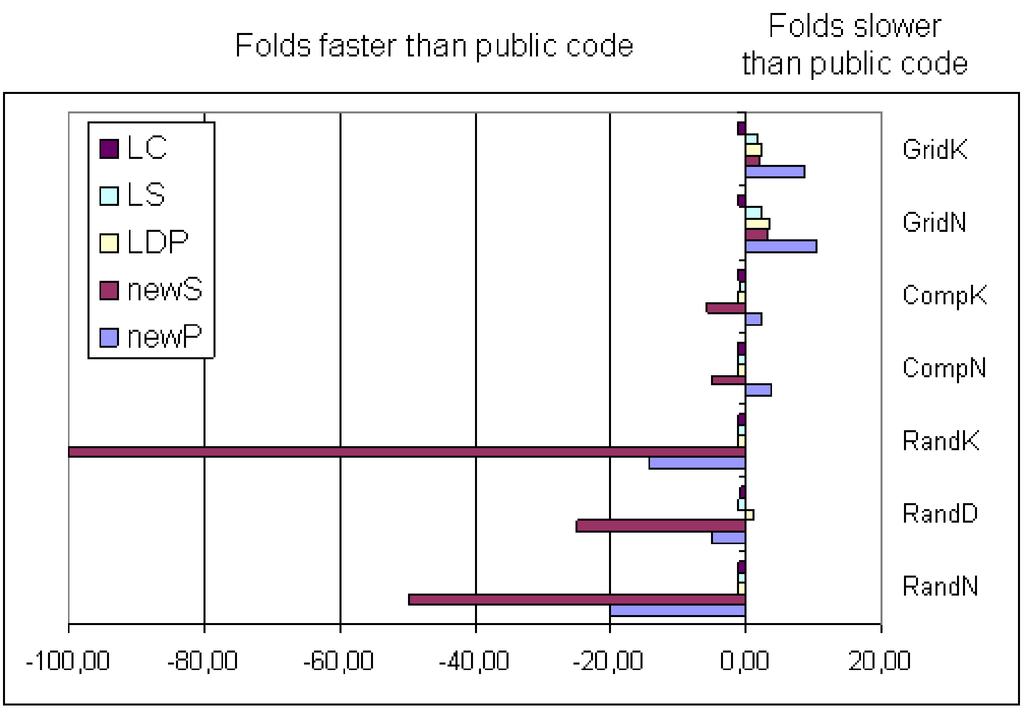

FIGURE 1. Conversion factor on the CPU time for each algorithm relatively to the public code.

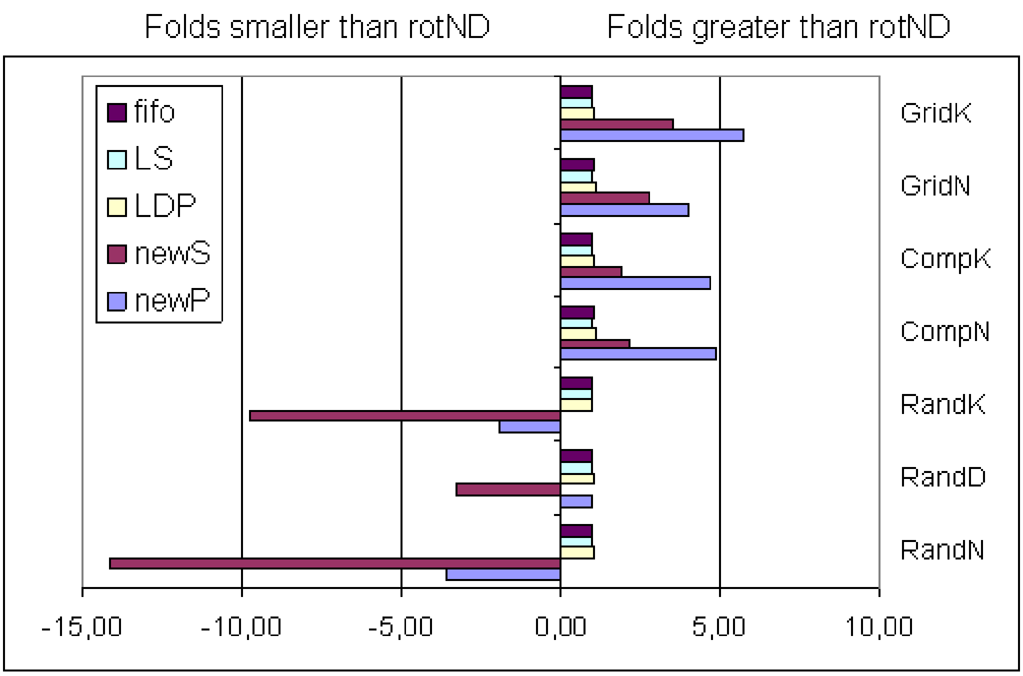

FIGURE 2. Conversion factor from rotND to the number of labels computed by each algorithm.

5. Conclusion

In this paper, we propose a new algorithm for finding the set of ND s-t paths based on ranking paths procedure. The algorithm makes use of a stop ranking condition which allows to determine the entire set of ND s-t paths ($\bar{D}$) without requiring that all the ND paths starting at the initial node s ($\bar{D}_s$) have to be computed. Two versions of the new algorithm were analyzed: sequential and parallel. However, the last one was not competitive when confronted with the sequential version.

The sequential version of the new algorithm showed to be much faster than the several versions for the labelling algorithm (L&DP, LS and LC), for solving random and complete network instances. In fact, as exhibited in Figure 1, the average reduction factor on the CPU is quite impressive for the class of randomly generated problems with a larger number of criteria (RandK). Even for the complete network instances the average reduction factor on the CPU time is circa 5.

The improvements obtained for the randomly generated networks are clearly explained by the effectiveness of the stop ranking condition producing, as shown in Figure 2, a considerable reduction on the number of labels determined by the new algorithm. Concerning to the complete network instances, the new algorithm benefits from the well identified drawback for the labelling procedures which comes from the need of computing all of the non-dominated paths from s to every vertex in the network. For the grid network instances, the labelling algorithm proved to be the best procedure with particular advantage for the label correcting version.

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, J.J. Madeira, E.Q. Martins, and F.M. Pires. A computational improvement for a shortest paths ranking algorithm. *European Journal of Operational Research*, 73:188–191, 1994.

[3] 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.

[4] 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.

[5] 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.

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

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

[8] 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.

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

[10] 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.

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

[12] 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.

[13] E.Q. Martins, M.M. Pascoal, and J.L. Santos. Deviation algorithms for ranking shortest paths. *International Journal of Foundations of Computer Science*, 10 (3):247–261, 1999.

[14] 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.

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

[16] J.P. Paixão, M.S. Rosa, and J.L. Santos. Labelling methods for the general case of the multi-objective shortest path problem - a computational study. Working Paper 07-xx, CMUC and submitted for publication., 2007.

[17] U. Pape. Implementation and efficiency of moore-algorithms for the shortest route problem. *Mathematical Programming*, 7:212–222, 1974.

[18] U. Pape. Algorithm 562: Shortest paths lengths. *ACM Transactions on Mathematical Software*, 6:450–455, 1980.

[19] J.L. Santos. Multiobjective shortest path problem. (http://www.mat.uc.pt/∼zeluis/INVESTIG/MSPP/mspp.htm).

[20] J.L. Santos. O problema do trajecto óptimo multiobjectivo, 1997. (Master degree dissertation; Mathematics Department; University of Coimbra).

[21] J.L. Santos. Uma abordagem ao problema do trajecto óptimo multiobjectivo. *Investigação Operacional*, 19:211–226, 1999.

[22] J.L. Santos. *Optimização vectorial em redes.* PhD thesis, Departamento de Matemática, Universidade de Coimbra, 2003.

[23] 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`