Efficient paths by local search

L. Paquete * J.L. Santos † D.J. Vaz *

* CISUC, Department of Informatics Engineering, University of Coimbra
Pólo II, 3030-290 Coimbra
paquete@dei.uc.pt, dvaz@student.dei.uc.pt

† CMUC, Department of Mathematics, University of Coimbra
3001-454 Coimbra
zeluis@mat.uc.pt

ABSTRACT

In this article, we describe an experimental analysis on a given property of connectedness of optimal paths for the multicriteria shortest path problem. Moreover, we propose a local search that explores this property and compare its performance with an exact algorithm in terms of running time and number of optimal paths found.

1. INTRODUCTION

Multicriteria shortest path problems arise in many applications. For instance, GPS systems allow choosing different criteria such as time or cost. However, there is no shortest path that optimizes all criteria since the fastest path may not be the cheapest. For instance, highways are fast but expensive since they are tolled, whereas national roads are free of charge but slow. Hence, one has to develop algorithms that output a set of optimal paths representing the optimal trade-off between the several criteria, from which the user chooses the most preferable.

This work describes a large experimental analysis to understand the structure of the efficient paths that can be exploited from an algorithmic point of view. In particular, we aim to know whether those efficient paths are close to each other, according to a proper definition of "closeness". To know whether this holds for most of the instances is highly relevant, since we could use this information to develop even more effective algorithms [5]. Our experimental results reported indicate that a large number of instances present such property. Therefore, we propose a local search that explores this property and compare it against an exact approach described in the literature.

2. NOTATION AND DEFINITIONS

Let G be a network, $G = (V,A)$ and w a mapping that defines each arc's weight, $w : A \mapsto \mathbb{Z}^Q$. For the simplicity of notation, we will say that a path is a sequence of arcs or nodes, depending of the context. Let us also denote the set of feasible paths as P. The goal of this problem is to find the efficient set of paths as follows

$$\min_{p \in P} f(p) := \left(\sum_{a \in p} w^1(a), \ldots, \sum_{a \in p} w^Q(a) \right) \tag{1}$$

The meaning of operator min is as follows: We say that a feasible path p *dominates* another feasible path p' if and only if $f^j(p) \leq f^j(p')$ for $j = 1, \ldots, Q$, with at least one strict inequality. If there is no feasible path that dominates p, then we say that p is an *efficient path*. The set of all efficient paths is denoted by $\mathscr{N}^E$. The image of the feasible set P forms a set of distinct points in the criterion space. We say that a vector z is *non-dominated* if it is the image of some efficient path $p \in \mathscr{N}^E$. The set of all non-dominated vectors is called the *non-dominated set*. In Eq. (1), operator min finds the nondominated set.

A label correcting algorithm to solve this problem (or to find the efficient set) is given by Paixão and Santos [4], which consists of an adaptation of the algorithm given by Vincke [7]. Although this algorithm finds the efficient set, it is too slow for large networks. In this work, we propose a new local search algorithm that explores a given property of the efficient paths that may improve the running time. We say that two paths, p_1 and p_2, are *adjacent* if and only if, after removing the arcs in common, we obtain a single cycle in the resulting undirected graph [3]. Also, we define the *adjacency graph* G', such that G' has a vertex for each efficient path $p \in \mathscr{N}^E$ and an edge between two vertices if and only if the corresponding paths are adjacent. The algorithm that is reported here explores the *connectedness of efficient paths*, which is defined as the connectedness of G'. Although it is not necessarily true that the efficient set for a given network is connected [3], a large fraction of networks may satisfy this condition. To the knowledge of the authors, connectedness of the efficient set only holds for particular cases of knapsack problems [1, 6].

3. CONNECTEDNESS ANALYSIS

In the experimental investigation mentioned above, we used benchmark instances described in the literature [4]. Those instances are grouped in three categories according to their size: *small*, *medium* and *large*. In each of the categories, there are 7 classes: *RandomN*: Random network (randomly generated arc), with the number of nodes varying, and having constant density and number of criteria; *RandomD*: Random network with constant number of nodes

D.J. Vaz acknowledges its grant BII-2009 from Fundação de Ciência e Tecnologia.

Algorithm 1 Local Search Algorithm

Input: Network $G = (V,A)$, $s,t \in V$.
Output: Set S.

$T, S := \emptyset$
Let $v : P \mapsto V$
for each crit. $q = 1,\ldots,Q$ **do**
  a) Find T_q, the reversed shortest path tree with root t on crit. q.
  b) Let $p \in T_q$ be path from s to t.
  c) Flag p as *not visited*.
  d) $v(p) := s$
  e) $S := S \cup p$
for each path p in S that is *not visited* **do**
  a) Flag p as *visited*.
  for each node $i \in p$ from $v(p)$ to t **do**
    for each arc $(i,j) \in A$ **do**
      for each crit. $q' = 1,\ldots,Q$ **do**
        a) Let $p_{(s,i)}$ be a path from node s to i and $p_{(s,i)} \subseteq p$
        b) Let $r \in T_{q'}$ be the path from node j to t in crit. q'
        c) $p' := p_{(s,i)} \cup \{(i,j)\} \cup r$
        d) $v(p') := j$
        e) Flag p' as *not visited*
        f) $S := \text{Filter}(S \cup \{p'\})$

and number of criteria, but varying density; *RandomK*: Random network with constant number of nodes and density, but varying the number of criteria; *CompleteN*: Complete network with constant number of criteria, but varying number of nodes; *CompleteK*: Complete network with constant number of nodes, but varying number of criteria; *GridN*: Grid (square mesh) with constant number of criteria, but varying number of nodes; *GridK*: Grid (square mesh) with constant number of nodes, but varying number of criteria. Each group corresponds to 50 distinct instances. For each class, there are 15-20 groups of 50 instances each. There are 19950 instances, from which 6600 are *small*, 6550 are *medium* and 6800 are *large*. For each of those instances, the weight of each arc for each criterion is generated randomly according to an uniform distribution in the range of $[1,1000]$.

We developed an algorithm for detecting connectedness for a given set of efficient paths. This algorithm outputs the number of connected components of the adjacency graph. For detecting whether a given instance is connected according to the notion of connectedness described in Section 2 we ran the algorithm for finding the set of efficient paths as described by Paixão and Santos [4], and then used this set as input to the algorithm described above to determine if the set of efficient paths was connected. All the *small* and *medium* instances, along with some *large* instances that have been tested, were found to have the set of efficient paths connected.

4. LOCAL SEARCH ALGORITHM

The local search algorithm presented in this section generates candidate efficient paths that are neighbors with respect to the definition of adjacency given in Section 2. Note that the number of efficient and neighbor paths can be exponentially large [2]. Therefore, we focus on a subset of neighbors whose size only depends linearly on the number of criteria, number of nodes and/or arcs.

The local search works as follows. First, all the shortest paths from every node to the target and for each criterion are generated

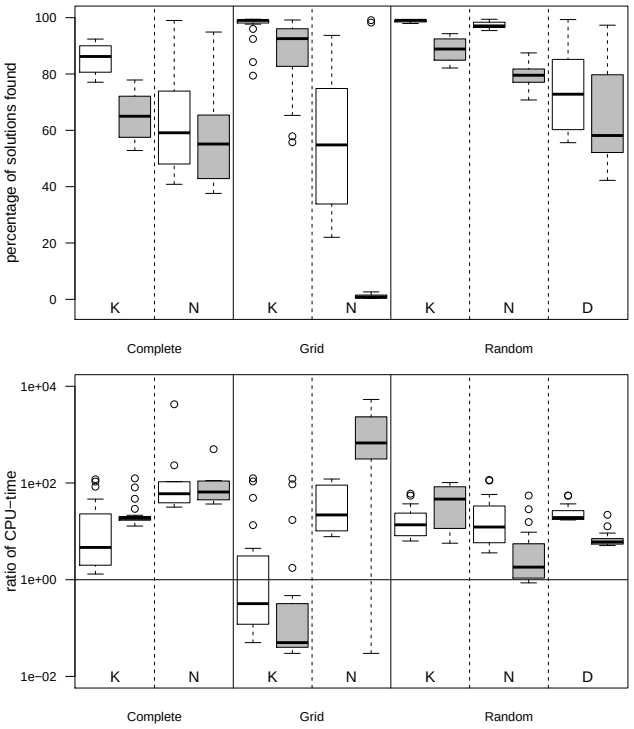

Figure 1: *Percentage of efficient solutions found (top) and ratio of CPU-time between label correcting and local search (bottom). The ratio is shown in logarithmic scale. The white and grey boxplots represent small and medium instances, respectively.*

by using Dijkstra's algorithm. Then, for each one of these shortest paths, new paths are generated from a path p as follows: for each node i of p from s to the target t, deviate from p at node i through an arc (i,j) and then, for each criterion, follow the shortest path that was previously computed from node j to the target. With this procedure, further new candidates for efficient paths are generated. The algorithm iterates over the procedure above for all paths that are generated. To avoid generating repeated paths, at each new path p' generated from path p, the algorithm starts from the first node in p' where the detour occured. This node will be denoted by $v(p)$ and for the shortest path initially determined, we define $v(p) = s$. We also denote a path that follows path p from node s to node i by $p_{(s,i)}$. The resulting algorithm is shown in Algorithm 1. The procedure Filter(S) in the final step removes the dominated paths from set S At each iteration of the second loop, the algorithm uses a LIFO strategy to choose the next path from S. In order to define a stopping criterion, we use the following technique [5]: The algorithm flags each new path found as *not visited*; the path becomes *visited* when it is chosen to generate new paths. This algorithm stops when all paths in S are flagged as *visited*.

Figure 1 presents the experimental results obtained by using the local search algorithm as compared to the label correcting approach. The plot in the top gives a boxplot for the percentage of efficient paths found by the local search algorithm for each instance type and size. The plot in the bottom shows a boxplot for the ratio of CPU-time between the label correcting approach and the local search algorithm. The experimental results indicate that the local search algorithm behaved well in Random and Complete in-

stances, where it finds over 80% of the efficient paths in RandomK and RandomN instances and between 50% and 90% in the remaining. For these classes of problems, the local search algorithm takes less than one tenth of the run-time of the exact approach. Additionally, in GridK instances, the local search finds more than 80% of the efficient paths, but it is slower than the exact approach. Finally, only a few portion of efficient paths was found by the local search algorithm in GridN instances.

5. CONCLUDING REMARKS

In this article, we performed an experimental analysis of connectedness for the multicriteria shortest path problem. The positive results obtained in this study suggest that local search algorithms may be an effective approach. We propose a local search algorithm that explores a stricter version of the neighborhood considered for connectedness. From our point of view, the results obtained by our approach were quite positive for most of the instance types, both in terms of number of efficient paths found and running time.

This approach can even be improved in terms of solution quality, mainly for instances of type Grid with increasing size, by considering an extension of the neighborhood that is explored by our approach. However, it is an open question whether it would still be efficient in terms of running time as compared to exact algorithms for this problem. As for instances of type Grid with increasing number of objectives, a more efficient dominance check may improve our approach. Finally, we remark that this local search exploration can be also applied for other problems defined over networks, such as the multicriteria minimum spanning tree problem.

6. REFERENCES

[1] J. Gorski, L. Paquete, F. Pedrosa, Greedy algorithms for a class of knapsack problems with binary weights, Computers & Operations Research, 2011, in press.

[2] P. Hansen, Bicriterion path problems, In G. Fandel and T. Gal (Eds.), Multiple Criteria Decision Making Theory and Application, LNEMS 177, Springer, pp. 109–127, 1979.

[3] M. Ehrgott, K. Klamroth, Connectedness of efficient solutions in multiple criteria combinatorial optimization. European Journal of Operational Research, 97: 159–166, 1997.

[4] J.P. Paixão and J.L. Santos, Labelling methods for the general case of the multiobjective shortest path problem - a computational study. Working paper CMUC 07-42, University of Coimbra, 2007.

[5] L. Paquete, T. Stützle, On local optima in multiobjective combinatorial optimization problems. Annals of Operations Research, 156(1): 83–97, 2007.

[6] F. Seipp, S. Ruzika, L. Paquete, On a cardinality constrained multicriteria knapsack problem, Report in Wirtschaftsmathematik Nr. 133/2011, University of Kaiserslautern. 2011.

[7] P. Vincke, Problémes multicritères, Cahiers du Centre d'Etudes de Recherche Opérationelle 16, 425–436, 1974