A Comparison of Two New Exact Algorithms for the Robust Shortest Path Problem

Roberto Montemanni Luca Maria Gambardella Alberto Donati

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Galleria 2
CH-6928 Manno-Lugano, Switzerland
{roberto, luca, alberto}@idsia.ch

1 Introduction

When transportation problems are modelled in mathematical terms, a road network is usually represented as a weighted digraph, where each arc is associated with a road and costs represent travel times. In this context, a shortest path problem has to be solved every time the quickest way to go from one place to another has to be calculated.

Unfortunately in the reality it is not easy to estimate travel times exactly, since they depend on many factors which are difficult to predict, such as traffic conditions, accidents or weather conditions. For this reason the fixed cost model previously introduced may be inadequate. To overcome this problem more complex frameworks have been studied. In particular a model where an interval of values, representing a range of possible real costs, is associated with each arc has been proposed. It is the *interval data model*, which is considered in this paper and will be described in details in Section 2.

Having adopted this model to represent reality, a criterion to drive optimization has to be chosen. We use the *robust deviation criterion* (sometimes referred to as *relative robustness criterion*). This criterion was discussed in Kouvelis and Yu [3], a book entirely devoted to robust discrete optimization.

A *robust deviation shortest path* from s to t is a path from s to t which minimizes the maximum deviation from the optimal shortest path from s to t over all realizations of arc costs.

Yu and Yang [7] conjectured that the robust deviation shortest path problem with interval data is $\mathcal{NP}$-hard. This conjecture was proven to be true in Zieliński [8].

Karaşan et al. [2] proposed a mixed integer programming formulation for the problem based on an important theoretical result (see Theorem 1).

In this paper we present an overview of the two main exact methods developed so far to solve the robust deviation shortest path problem with interval data.

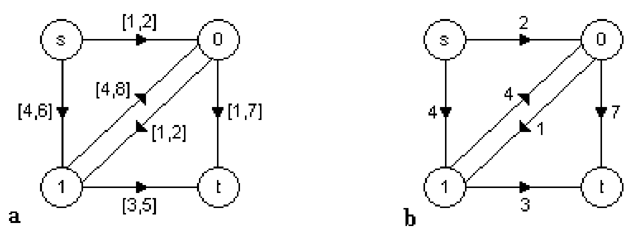

Figure 1: Interval graph (a) and scenario induced on it by $p = \{s, 0, t\}$ (b).

Computational results are also presented. They aim to compare the performance of the algorithms on a set of road networks with different characteristics.

2 Problem description

A directed graph $G = (V, A)$, where V is a set of vertices and A is a set of arcs is given together with a starting vertex $s \in V$, and a destination vertex $t \in V$. An interval $[l_{ij}, u_{ij}]$, with $0 < l_{ij} \leq u_{ij}$, is associated with each arc $(i, j) \in A$. Intervals represent ranges of possible costs (travel times). An example of interval graph is given in Figure 1(a).

We can formally describe the robust deviation shortest path problem with interval data through the following definitions:

Definition 1 *A* scenario *r is a realization of arc costs, i.e. a cost $c^r_{ij} \in [l_{ij}, u_{ij}]$ is fixed $\forall (i, j) \in A$.*

Definition 2 *The* robust deviation *for a path p from s to t in a scenario r is the difference between the cost of p in r and the cost of the shortest path from s to t in scenario r.*

Definition 3 *A path p from s to t is said to be a* robust deviation shortest path *if it has the smallest (among all paths from s to t) maximum (among all possible scenarios) robust deviation.*

A scenario can be seen as a snapshot of the network situation, and a robust deviation shortest path is a path which guarantees reasonably good performance under any possible configuration of travel times over the network.

Given a directed graph and an origin/destination pair (s, t), the robust deviation shortest path problem is the problem of retrieving a robust deviation shortest path.

The following important result is at the basis of the methods described in Section 3.

Theorem 1 (Karaşan et al. [2]) *The robust deviation for path p is maximized at the scenario in which the lengths of all arcs on p are at upper bounds and the lengths of all other arcs are at lower bounds.*

Theorem 1 implies that we need to consider only a finite number of scenarios, namely as many as the number of paths in the graph.

In the remainder of this paper we will refer to the scenario r, derived from path p as described in Theorem 1, as the scenario *induced* by path p. We will also refer to the cost of p (i.e. $\sum_{(i,j)\in p} u_{ij}$) minus the cost of a shortest path of the scenario induced by p, as the *robustness cost* of p. Figure 1(b) depicts the scenario induced by path $p = \{s, 0, t\}$ on the graph of Figure 1(a). The robustness cost of p is in this case $(2 + 7) - (2 + 1 + 3) = 3$.

3 Exact algorithms

3.1 Algorithm 1

This method was presented in Montemanni and Gambardella [5]. It is based on the conjecture that a path ranking on scenario u, where $c^u_{ij} = u_{ij}\ \forall (i,j) \in A$, is also a good ranking in terms of robust deviation.

Exploiting this approach, i.e. examining paths following the ranking on scenario u, a lower bound for the robustness cost of the feasible paths not yet examined can be provided at each iteration.

We indicate with p_i the i-th path in the ranking on scenario u, with $\text{Cost}_u(p_i)$ the cost in scenario u of path p_i, with $\text{Cost}_R(p_i)$ the robustness cost of p_i and with ub the robustness cost of the path with minimum robustness cost retrieved so far. The following inequality holds:

$$\text{Cost}_R(p_j) \geq \text{Cost}_u(p_i) - \text{Cost}_u(p_1) \quad \forall j \geq i \tag{1}$$

Inequality (1) suggests an exit criterion for the algorithm. The optimal solution has been found if, at iteration i, the following inequality holds:

$$\text{Cost}_u(p_i) - \text{Cost}_u(p_1) \geq ub \tag{2}$$

In practice, if paths are examined in non-decreasing order of $\text{Cost}_u(p)$, a lower bound for the robustness cost of the paths not yet examined is constantly available. This bound can be compared with the robustness cost of the best path retrieved, in order to decide whether to continue examining paths or to stop the computation.

According to Martins and dos Santos [4], and notwithstanding a very high theoretical computational complexity for their algorithm, ranking the shortest paths of a fixed scenario is, in practice, an easy task.

Another important consideration, which follows from Theorem 1, is that the robustness cost of a given path can be evaluated by solving a classic shortest path problem in the scenario induced by it. This operation can be carried out in polynomial time (see Dijkstra [1]).

The algorithm which follows from the theoretical results and considerations above, works in the following way: a procedure to rank the paths from s to t in scenario u by non-decreasing values of $\text{Cost}_u(p)$ is run. For each path retrieved, the respective robustness cost is calculated (by solving a shortest path problem in the scenario induced by it) and eventually the value of ub is updated. The algorithm stops when the condition described by inequality (2) is matched.

Some theoretical results, useful to speed up the algorithm, are also presented in [5]. If there is no overlap between a path p from s to t and a shortest path SP_l from s to t in scenario l, where $c^l_{ij} = l_{ij}\ \forall(i,j) \in A$, then the robustness cost of p can be calculated without solving a shortest path problem on the graph induced by p, because SP_l is a shortest path in this scenario.

Another speed-up rule is the following one. If we call SP_{p_j} the shortest path in the scenario induced by path p_j, and we have that $p_i \cap SP_{p_j} \subseteq p_j \cap SP_{p_j}$ with $i > j$, then we do not need to calculate $\text{Cost}_R(p_i)$, because it possible to prove that $\text{Cost}_R(p_i) \geq ub$.

3.2 Algorithm 2

Montemanni et al. [6] propose a branch and bound algorithm. This algorithm constructs and visits a search-tree.

We refer to the search-subtree rooted in the search-tree node d as $T(d)$. Each node d of the search tree is identified by four elements: **1.** the arc list $in(d)$. The arcs contained in $in(d)$ must appear in all of the paths associated with the nodes of $T(d)$; **2.** the arc list $out(d)$. The arcs contained in $out(d)$ are forbidden for all of the paths associated with the nodes of $T(d)$; **3.** the path $p(d)$. It is the path with the minimum cost in scenario u which respects the limitations imposed by arc sets $in(d)$ and $out(d)$, i.e. $p(d)$ must contain the arcs in $in(d)$ and cannot include the arcs in $out(d)$; **4.** the lower bound $lb(d)$. It is a lower bound for the robustness cost of the paths associated with the search-tree nodes of $T(d)$. Because of the meaning of $out(d)$ and Theorem 1, we know that the cost of each arc (i,j) in $out(d)$ will be equal to l_{ij} in each of the scenarios derived from the paths associated with the nodes of $T(d)$. We define $SP_{out(d)}$ as the shortest path of the scenario where the cost of all the arcs in $out(d)$ are at their lower bounds and the cost of the remaining arcs are at their upper bounds, and $\text{Cost}_{out(d)}(p)$ as the cost of path p in this scenario. We can then derive the following definition for $lb(d)$:

$$lb(d) := \text{Cost}_u(p(d)) - \text{Cost}_{out(d)}(SP_{out(d)}) \tag{3}$$

The root r of the search-tree constructed by the algorithm is the node with $in(r) = \emptyset$, $out(r) = \emptyset$ and $lb(r) = 0$. Initially r is the only node of the set of nodes to be examined, and ub is set to the robustness cost of $p(r)$.

At each iteration the not yet examined node d with the smallest value of $lb(d)$ is selected and examined. The first arc a on path $p(d)$ (starting from node s) which is not contained in $in(d)$ is identified (if it exists). If $p(d) = in(d)$, node d is a leaf of the search-tree and consequently a does not exist. Otherwise two new search-tree nodes are created. The first new node, d', has $in(d') = in(d)$ and $out(d') = out(d) \cup \{a\}$, while d'', the second one, has $in(d'') = in(d) \cup \{a\}$ and $out(d'') = out(d)$. Once arc $a = (i,j)$ is inserted into $in(d'')$, all the arcs of type (i,k) and (k,j) are inserted into $out(d'')$, since they cannot be part of a shortest path from s to t anymore. $p(d')$ is calculated and in case its robustness cost is lower than ub, ub is updated

Table 1: Networks characteristics.

Networks	$\|V\|$	$\|A\|$	$\frac{1}{\|A\|}\sum \frac{l_{ij}}{u_{ij}}$
Sottoceneri[1]	387	1038	0.663
Lugano[2]	576	1327	0.899
Stuttgart[3]	2490	16153	0.822
Padova[4]	1522	2579	0.313

and every search tree node $\overline{d}$ with $lb(\overline{d}) \geq ub$ is deleted from the set of nodes to be examined. In case the lower bounds at nodes d' and d'' are less then ub, d' and d'' are inserted into the set of search-tree nodes to be examined.

The procedure stops when the set of nodes to be examined becomes empty.

To evaluate the robustness cost of a path p, it is enough to solve a classic shortest path problem in the scenario induced by p (according to Theorem 1). The algorithm described in Dijkstra [1] was used, while a straightforward modification of it was adopted to calculate $p(d)$, given a search-tree node d.

4 Computational experiments

The graphs on which tests are run represent real road networks, and the interval costs associated with arcs are realistic. Characteristics of the graphs are presented in the first four columns of Table 1, where meanings are as follows: *Networks* contains the name of the graphs, $|V|$ and $|A|$ the number of vertices and arcs respectively. In the forth column the average values for the ratio $\frac{l_{ij}}{u_{ij}}$ are reported. They give an indication of the width of travel time intervals.

It is important to notice that Stuttgart has a large number of arcs, while Padova presents a value smaller than the other networks in the forth column. This depends on a very high variability in travel times between pick and non-pick hours and is a consequence of the network conformation itself.

The results presented in Table 2 have been obtained on a Pentium 4 1.5 GHz / 256 MB computer. For each network considered we report the average value and the standard deviation for computation times (in seconds) over 20 instances (with random origins and destinations). A maximum computation time of 3600 seconds is allowed. Since Algorithm 2 almost never concluded the computation within this time limit on network Padova, the corresponding entries of the table are consequently empty.

The results suggest that Algorithm 2 is faster than Algorithm 1 for the first three problems,

[1]Main roads of the Sottoceneri region, i.e. the southern part of Canton Ticino (CH). Provided by *Pina Petroli SA* (http://www.pina.ch).

[2]Road network of the city of Lugano (CH). Provided by *CRTL (Commissione Regionale dei Trasporti del Luganese).*

[3]Road network (aggregated) of the Stuttgart area (D). Provided by *PTV (Planung Transport Verkehr) AG* (http://www.ptv.de).

[4]Main roads of the Padova area (I). Provided by *Comune di Padova* (http://www.comune.padova.it).

Table 2: Computation times (in seconds).

Networks	Alg. 1 [5]		Alg. 2 [6]	
	Avg	StDev	Avg	StDev
Sottoceneri	0.078	0.159	0.073	0.136
Lugano	0.191	0.316	0.117	0.191
Stuttgart	3.129	5.666	1.752	2.653
Padova	73.785	146.022	-	-

but its computation times explode for the last network considered. For this problem Algorithm 1 is not very fast but remains the only possible choice. This not very good performance of the algorithms on the Padova network depends, in our opinion, on the wideness of the travel time intervals in this network (see Table 1, column four). This factor dramatically impacts on the quality of the lower bounds used by the two algorithms.

In conclusion, Algorithm 2 is in general the fastest, but in case of networks with wide (average) travel time intervals, Algorithm 1 seems to be the best choice.

Another interesting observation is about standard deviations of computation times. They are always very high, and this indicates that both the algorithms (and in particular Algorithm 1) are very sensitive to the values of the origin/destination pair. However, since it is easy to see that different origin/destination pairs can generate problems with a very different complexity on a same graph, this is reasonable.

5 Conclusion

We have compared two exact algorithms for the robust shortest path problem with interval data on some real road networks.

The results suggest that the choice of the best one strongly depends on the characteristics of the road network on which the algorithm has to be run. In particular the wideness of travel times intervals associated with the arcs seems to have a very strong impact on the performance of the methods.

References

[1] E.W. Dijkstra, "A note on two problems in connection with graphs", *Numerische Mathematik* 1, 269-271 (1959).

[2] O.E. Karaşan, M.Ç. Pinar, and H. Yaman, "The robust shortest path problem with interval data", *Computers and Operations Research* (2002).

[3] P. Kouvelis and G. Yu, *Robust Discrete Optimization and its applications*, Kluwer Academic Publishers (1997).

[4] E.Q.V. Martins and J.L.E. dos Santos, "A new shortest paths ranking algorithm", *Investigação Operacional* 20(1), 47-62 (2000).

[5] R. Montemanni and L.M. Gambardella, "An exact algorithm for the robust shortest path problem with interval data", *Computers and Operations Research* 31(10), 1667-1680 (2004).

[6] R. Montemanni, L.M. Gambardella, and A.V. Donati, "A branch and bound algorithm for the robust shortest path problem with interval data", *Operations Research Letters* 32(3), 225-232 (2004).

[7] G. Yu and J. Yang, "On the robust shortest path problem", *Computers and Operations Research* 25(6), 457-468 (1998).

[8] P. Zieliński, "The computational complexity of the relative robust shortest path problem with interval data", *European Journal of Operational Research* (to appear).