An exact algorithm for the robust shortest path problem with interval data

R. Montemanni* and L.M. Gambardella

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Galleria 2, CH-6928 Manno, Switzerland

Abstract

The robust deviation shortest path problem with interval data is studied in this paper.

After the formulation of the problem in mathematical terms, an exact algorithm, based on a very simple concept, is described. Some practical improvements to the basic idea, which speed up the method, are also presented.

Computational results corroborate the correctness of the conjecture on which the algorithm is based and the potential of the approach. In particular, our method is proven to be able to retrieve high-quality solutions very quickly on some families of networks. For this reason it could alternatively be used as a fast heuristic method.

Keywords: Shortest path problem, robust optimization, interval data.

1 Introduction

Many real transportation and telecommunications problems can be represented in mathematical terms as shortest path problems on weighted digraphs, where a fixed cost is associated with each arc. Sometimes the model is not realistic enough, and consequently more complex representations have to be considered.

A model where a set of alternative graphs are considered at the same time (*scenario model* - see Yu and Yang [9] and Dias and Clímaco [2]) and a model where an interval

*Corresponding author. Tel: +41 91 610 8568. Fax: +41 91 610 8661. Email: roberto@idsia.ch.

of values is associated with each arc (*interval data model* - see Dias and Clímaco [2] and Karaşan et al. [5]) have been studied in the literature. In this paper the interval data model, which will be described in detail in Section 2, is considered.

Having adopted this model to represent reality, a criterion to drive optimization has to be chosen. The *robust deviation criterion*, sometimes referred to as *relative robustness criterion* (see Zieliński [10]), is the one we adopt. This criterion is discussed in Kouvelis and Yu [6], a book entirely devoted to robust discrete optimization, and has already been used in Karaşan et al. [5] for the shortest path problem with interval data.

Yu and Yang [9], which proved that the robust deviation shortest path problem with scenarios is $\mathcal{NP}$-hard, conjectured that the problem with interval data is also $\mathcal{NP}$-hard. This conjecture is confirmed in Zieliński [10], when the robust deviation shortest path with interval data is proven to be $\mathcal{NP}$-hard.

In Karaşan et al. [5] an integer programming formulation of the problem (solved using standard integer programming technology) is presented, together with a preprocessing technique (inspired by the one described in Yaman et al. [8] for the robust spanning tree problem). This technique, which identifies and eliminates arcs which will never be in a robust path, significantly improves computational results on the family of benchmarks considered in [5]. Unfortunately it works only for *acyclic*, *layered* graphs with a small *width*. We remind the reader that a graph is *acyclic* when its arcs do not form any cycles, and it is *layered* when its vertices can be partitioned into a chain of disjoint subsets, in such a way that the cardinality of each subset is limited by a given constraint, called *width*, and arcs exist only from each subset to the following one of the chain.

In this paper an exact algorithm, which works by exploiting a simple property we have discovered, is presented. It works for general directed graphs and can be used as a heuristic method by truncating the execution before its natural end.

In Section 2 the problem we consider is formally described, while in Section 3 the algorithm we propose is discussed. In Section 4 computational results are presented. Finally

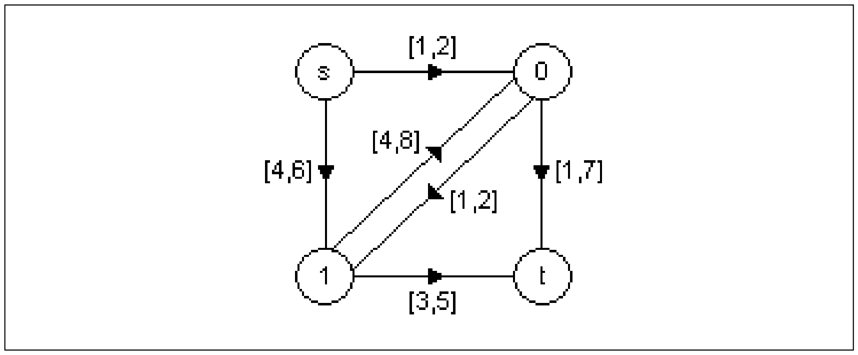

Figure 1: Example of interval graph.

conclusions can be found in Section 5.

2 Problem description

In this paper the robust deviation criterion (Kouvelis and Yu [6]) is applied to the shortest path problem defined on a directed graph $G = (V, A)$, where V is a set of vertices and A is a set of arcs. A starting vertex $s \in V$, and a destination vertex $t \in V$ are given and an interval $[l_{ij}, u_{ij}]$, with $0 \leq l_{ij} \leq u_{ij}$, is associated with each arc $(i, j) \in A$. Intervals represent ranges of possible costs (e.g. travel times, transmission delays) for each arc.

An example of interval graph is given in Figure 1.

According to Karaşan et al. [5], 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.*

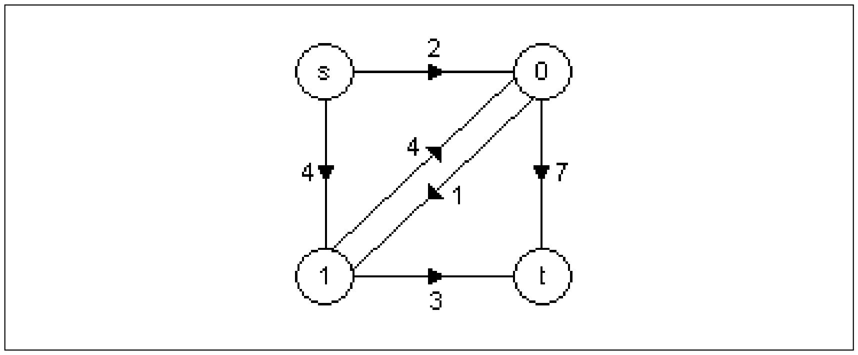

Figure 2: Scenario induced by $p = \{s, 0, t\}$ on the graph of Figure 1.

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 (transmission delays) over the network.

The following result is known from the literature:

Observation 1 (Karaşan et al. [5]). *Given a path p from s to t, a scenario r which makes the robust deviation maximum is the one where each arc (i, j) on p has cost u_{ij} and each arc (k, h) not on p has cost l_{kh}, i.e. $c^r_{ij} = u_{ij}\ \forall (i, j) \in p$ and $c^r_{kh} = l_{kh}\ \forall (k, h) \in A \backslash p$.*

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

Applying Observation 1, Karaşan et al. [5] described the problem through a mixed integer programming formulation, which is presented after the following introduction to its variables:

- y_{ij}: it is 1 when arc (i, j) is on the robust path from s to t defined by the formulation (notice that it may be a non-simple path, see [5] for more details); 0 otherwise;
- x_i: it contains the cost of the shortest path from s to i in the scenario induced by the path defined by y variables.

$$(RDSP) \quad \text{Min} \sum_{(i,j)\in A} u_{ij} y_{ij} - x_t \tag{1}$$

$$\text{s.t. } x_j \leq x_i + l_{ij} + (u_{ij} - l_{ij})\, y_{ij} \qquad \forall (i,j) \in A \tag{2}$$

$$\sum_{(j,k)\in A} y_{jk} - \sum_{(i,j)\in A} y_{ij} = \begin{cases} 1 & \text{if } j = s \\ -1 & \text{if } j = t \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in V \tag{3}$$

$$x_s = 0 \tag{4}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{5}$$

$$x_j \geq 0 \qquad \forall j \in V \tag{6}$$

The key-inequalities of the formulation are those in (2), which set arc-costs according to the scenario induced by y variables, and consequently maintain consistency between x and y variables. The remaining constraints are basically those of the classic formulation for the shortest path problem (see, for example, Karaşan et al. [5]).

3 The algorithm

The only known methodology available to solve the problem is, as far as we are aware, the one proposed in Karaşan et al. [5], based on a preprocessing technique and on the use of a commercial solver to solve *RDSP* (see Section 1). In this section we present an ad-hoc exact algorithm, which can also be used as a heuristic method by stopping the execution before its natural end. We first introduce the notation adopted in the remainder of this paper, then we describe the idea on which the method is based and some rules which speed up the algorithm. Finally the complete algorithm is summarized in Section 3.4.

3.1 Notation

- l: scenario in which $c^l_{ij} = l_{ij}\ \forall (i,j) \in A$;
- u: scenario in which $c^u_{ij} = u_{ij}\ \forall (i,j) \in A$;
- P: set of all the possible paths from s to t in G;
- $C^p(q)$: cost of path q in the scenario induced by path p;
- $SP(p)$: shortest path from s to t in the scenario induced by path p;
- $LC(p)$: cost of path p in scenario l;
- $UC(p)$: cost of path p in scenario u;
- $RC(p)$: robustness cost of path p;
- p_i: i-th shortest path from s to t in scenario u, i.e. path in i-th position in a ranking of paths for non-decreasing values of $UC(p)$;
- LSP: a shortest path from s to t in scenario l.

3.2 Starting idea

The algorithm presented is based on the conjecture that a path ranking on scenario u is also a good ranking in terms of robust deviation.

Exploiting this approach, it is also possible to notice that if we examine the paths from s to t by non-decreasing values of $UC(p)$ (i.e. following the order of a shortest paths ranking in scenario u), we are able to provide at each iteration a lower bound for the robustness cost of the paths from s to t not yet examined. In particular, the following results hold.

Lemma 1. $UC(p_1) \geq C^p(SP(p))\ \ \forall p \in P$

Proof. We call r the scenario induced by p. From the definition of scenario u we have:

$$c^u_{ij} \geq c^r_{ij}\ \ \forall (i,j) \in A \tag{7}$$

Using (7) we can conclude:

$$UC(p_1) = \sum_{(i,j)\in p_1} c^u_{ij} \geq \sum_{(i,j)\in p_1} c^r_{ij} \geq \sum_{(i,j)\in SP(p)} c^r_{ij} = C^p(SP(p))\ \ \forall p \in P \tag{8}$$

□

The result of Lemma 1 is used in the following theorem, which provides, if the paths are examined in non-decreasing order of $UC(p)$, a lower bound for the robustness cost of the paths not yet considered.

Theorem 1. $RC(p_k) \geq UC(p_i) - UC(p_1) \;\; \forall k \geq i$

Proof. Using Lemma 1 we have:

$$RC(p_k) = UC(p_k) - C^{p_k}(SP(p_k)) \geq UC(p_k) - UC(p_1) \;\; \forall p_k \in P \tag{9}$$

By definition we have:

$$UC(p_k) \geq UC(p_i) \;\; \forall k \geq i \tag{10}$$

Using (9) and (10) we have:

$$RC(p_k) \geq UC(p_k) - UC(p_1) \geq UC(p_i) - UC(p_1) \;\; \forall k \geq i \tag{11}$$

□

The result of Theorem 1 is used in Proposition 1, which provides an exit criterion for the exact algorithm we present in this paper.

Proposition 1. *If* $UC(p_i) - UC(p_1) \geq \min_{j \in \{1,\ldots,i\}} \{RC(p_j)\}$ *then* $\arg\min_{p_j \in \{p_1,\ldots,p_i\}} \{RC(p_j)\}$ *is a robust deviation shortest path.*

Proof. From Theorem 1 we have:

$$RC(p_k) \geq UC(p_i) - UC(p_1) \;\; \forall k \geq i \tag{12}$$

From the hypothesis we have:

$$UC(p_i) - UC(p_1) \geq \min_{j \in \{1,\ldots,i\}} \{RC(p_j)\} \;\; \forall k \geq i \tag{13}$$

Using (12) and (13) together we have:

$$RC(p_k) \geq \min_{j \in \{1,\ldots,i\}} \{RC(p_j)\} \;\; \forall k \geq i \tag{14}$$

From (14) we can conclude that $\arg\min_{p_j \in \{p_1,\dots,p_i\}} \{RC(p_j)\}$ is a robust deviation shortest path. $\square$

In practice, if paths are examined in non-decreasing order of $UC(p)$, 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 [7], and notwithstanding a theoretical computational complexity of $O(K|A|)$ for their algorithm, ranking the first K shortest paths of a fixed scenario is, in practice, an easy task.

We have adopted the algorithm described in [7], which is based on the concept of path deletion, because it is easier to code than other, more famous algorithms (e.g. Eppstein [4]), and in the same time is not less efficient in practice (notwithstanding a higher theoretical computational complexity, see [7]).

Another consideration, which follows from Observation 1, is that the evaluation of the robustness cost of a given path can be done by solving a classic shortest path problem in the scenario induced by it. This operation can be carried out with complexity $O(|A|)$ (see Ahuja et al [1]). In our implementation we have adopted the procedure described in Dijkstra [3], which has a computational complexity of $O(|V|^2)$.

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 $UC(p)$ is run. For each path retrieved, the respective robustness cost is calculated (by solving a shortest path problem in the scenario induced by it). The algorithm stops when the condition described in Proposition 1 is matched (exact solution) or when a given maximum number of paths (K) has been examined (heuristic solution).

A more formal description of the algorithm will be given in Section 3.4, where some improvements to the basic idea (which will be presented in Section 3.3) will be also incorporated.

3.3 Improvements

Some theoretical results, which speed up the algorithm briefly sketched near the end of Section 3.2, are presented in this section.

3.3.1 Rule A

The result described in this section will be used, in particular circumstances, to calculate the robustness cost of the path p under investigation, without solving a shortest path problem in the scenario induced by p. This saves computation time.

Theorem 2. *If p is a path from s to t and $p \cap LSP = \emptyset$ then $SP(p) = LSP$.*

Proof. By definition we have:

$$LC(LSP) = \sum_{(i,j)\in LSP} l_{ij} \leq \sum_{(i,j)\in q} l_{ij} \leq$$

$$\sum_{(i,j)\in q\cap p} u_{ij} + \sum_{(i,j)\in q\setminus p} l_{ij} = C^p(q) \quad \forall q \in P \tag{15}$$

Using the hypothesis and (15) we have:

$$C^p(SP(p)) \leq C^p(LSP) = \sum_{(i,j)\in LSP\cap p} u_{ij} + \sum_{(i,j)\in LSP\setminus p} l_{ij} =$$

$$\sum_{(i,j)\in LSP} l_{ij} = LC(LSP) \leq C^p(SP(p)) \tag{16}$$

From (16) we can conclude:

$$SP(p) = LSP \tag{17}$$

□

Theorem 2 states that if there is no overlap between a path p from s to t and LSP, then we can calculate the robustness cost of p without solving a shortest path problem on the graph induced by p, because we already know that $SP(p) = LSP$. Indeed, if there is no overlap between a path p and LSP then, in the scenario induced by p, the costs of all the arcs on LSP are at their lower bounds. This implies that $SP(p) = LSP$.

3.3.2 Rule B

The result presented in this section will be used, in particular circumstances, to skip the calculation of the robustness cost of the path p under investigation. This saves computation time.

Lemma 2. *If $i > j$ and $SP(p_j) \cap p_i \subseteq SP(p_j) \cap p_j$ then $C^{p_i}(SP(p_i)) \leq C^{p_j}(SP(p_j))$.*

Proof. Using the hypothesis we have:

$$C^{p_i}(SP(p_j)) = \sum_{(i,j)\in SP(p_j)\cap p_i} u_{ij} + \sum_{(i,j)\in SP(p_j)\setminus p_i} l_{ij} =$$

$$\sum_{(i,j)\in SP(p_j)\cap p_j} u_{ij} + \sum_{(i,j)\in SP(p_j)\setminus p_j} l_{ij} + \sum_{(i,j)\in\left\{\begin{array}{c}\{SP(p_j)\cap p_j\}\setminus\\ \{SP(p_j)\cap p_i\}\end{array}\right\}} (l_{ij} - u_{ij}) =$$

$$C^{p_j}(SP(p_j)) + \sum_{(i,j)\in\left\{\begin{array}{c}\{SP(p_j)\cap p_j\}\setminus\\ \{SP(p_j)\cap p_i\}\end{array}\right\}} (l_{ij} - u_{ij}) \tag{18}$$

By definition $l_{ij} \leq u_{ij}$, and then:

$$\sum_{(i,j)\in\left\{\begin{array}{c}\{SP(p_j)\cap p_j\}\setminus\\ \{SP(p_j)\cap p_i\}\end{array}\right\}} (l_{ij} - u_{ij}) \leq 0 \tag{19}$$

From (18) and (19) we have:

$$C^{p_i}(SP(p_j)) \leq C^{p_j}(SP(p_j)) \tag{20}$$

By definition we also have:

$$C^{p_i}(SP(p_i)) \leq C^{p_i}(SP(p_j)) \tag{21}$$

From (20) and (21) we can conclude:

$$C^{p_i}(SP(p_i)) \leq C^{p_j}(SP(p_j)) \tag{22}$$

□

Lemma 2 states that, given two paths p_i and p_j with p_i after p_j in the path ranking in scenario u such that p_j overlaps more with $SP(p_j)$ than p_i, then the cost (in the scenario induced by p_i) of $SP(p_i)$ is less then or equal to the cost (in the scenario induced by p_j) of $SP(p_j)$.

Theorem 3 uses the result of Lemma 2 to provide an important criterion, which allows us to identify whether a path is dominated by another or not.

Theorem 3. *If $i > j$ and $p_i \cap SP(p_j) \subseteq p_j \cap SP(p_j)$ then $RC(p_i) \geq RC(p_j)$.*

Proof. From Lemma 2 we have:

$$C^{p_i}(SP(p_i)) \leq C^{p_j}(SP(p_j)) \tag{23}$$

By definition we have:

$$UC(p_i) \geq UC(p_j) \tag{24}$$

From (23) and (24) we can conclude:

$$RC(p_i) = UC(p_i) - C^{p_i}(SP(p_i)) \geq UC(p_j) - C^{p_j}(SP(p_j)) = RC(p_j) \tag{25}$$

□

Theorem 3 states that, given two paths p_i and p_j, with p_i after p_j in the path ranking in scenario u, such that p_j overlaps more with $SP(p_j)$ than p_i, then the robustness cost of p_i is higher than the robustness cost of p_j.

The result of Theorem 3 is used in Proposition 2 to give a formal criterion to decide whether it is possible to skip the calculation of the robustness cost of the path under investigation or not.

Proposition 2. *If $\exists p \in \{p_1, p_2, \ldots, p_{i-1}\}$ such that $p_i \cap SP(p) \subseteq p \cap SP(p)$ then $RC(p_i) \geq \min_{j \in \{1,\ldots,i-1\}} \{RC(p_j)\}$.*

Proof. From Theorem 3 we have:

$$RC(p_i) \geq RC(p) \tag{26}$$

By definition we have:

$$RC(p) \geq \min_{j \in \{1,\ldots,i-1\}} \{RC(p_j)\} \tag{27}$$

From (26) and (27) we can conclude:

$$RC(p_i) \geq \min_{j \in \{1,\ldots,i-1\}} \{RC(p_j)\} \tag{28}$$

$\square$

Proposition 2 states that if we know that p_i, the i-th shortest path in scenario u, is dominated (according to Theorem 3) by a path p_j with $j < i$, then we do not need to calculate $RC(p_i)$ (i.e. solving a shortest path problem in the scenario induced by p_i), because we already know that it will not improve the best result already available.

3.4 Pseudo-code

In this section the results described in Section 3.2 and Section 3.3 are summarized into an algorithm, whose pseudo-code is presented in Figure 3.

```
RelRobShortestPath(K)   // $K$ = number of paths to be considered
Calculate $LSP$;
$i := 0$;
$stop :=$ false;
$UB := \infty$;
While($i \leq K$ and $stop =$ false)
  Retrieve the $i$-th shortest path from $s$ to $t$ in scenario $u$;
  If($\nexists\, p_j$, $j < i$, such that $p_i \cap SP(p_j) \subseteq p_j \cap SP(p_j)$)    // Rule $B$
    If($p_i \cap LSP = \emptyset$)                                                  // Rule $A$
      $SP(p_i) := LSP$;                                                            // Rule $A$
    Else                                                                           // Rule $A$
      $SP(p_i) :=$ shortest path from $s$ to $t$ in the scenario induced by $p_i$;
    $RC(p_i) := UC(p_i) - C^{p_i}(SP(p_i))$;
    If($RC(p_i) < UB$)
      $UB := RC(p_i)$;
      $PathUB := p_i$;
  If($UB \leq UC(p_i) - UC(p_1)$)
    $stop :=$ true;
Return $PathUB$;
```

Figure 3: Algorithm for the robust deviation shortest path problem with interval data.

The algorithm starts by solving a shortest path problem in scenario l (the algorithm described in Dijkstra [3] is used) and by initializing some variables. An iterative statement is then entered. At each iteration i, the i-th shortest path from s to t in scenario u is retrieved (the algorithm described in Martins and dos Santos [7] is used). A test is carried out to check whether this path can be better than the best robust path currently available. According to the result of the test, another condition is eventually checked. The test suggests whether it is necessary to solve a shortest path problem in the scenario induced by p_i, or not (if the problem has to be solved then the algorithm described in Dijkstra [3] is used). At this point we are able to evaluate the robustness cost of p_i, and eventually to update the best robust path currently available. Finally, a test is carried out to see whether the exit condition is verified. Once the algorithm exits from the iterative statement, the best shortest path in terms of robust deviation, among those considered, is returned.

If the variable *stop* is *false* when the algorithm terminates, then the path returned is a heuristic solution to the problem, otherwise it is an exact solution.

4 Computational results

In this section some computational results are presented in order to evaluate the performance of the method described in Section 3.

The algorithm described in Figure 3 has been implemented in C++ and all the tests on our method have been carried out on a computer equipped with a Pentium 4 1.5GHz processor and 256MB of memory.

In Section 4.1 the benchmarks adopted in this paper are described. In Section 4.2 a study on the correlation between the length of robust paths and the performance of our algorithm is presented. In Section 4.3 the results obtained by the algorithm are discussed.

4.1 Description of the benchmarks

The weighted digraphs on which the benchmarks adopted in this paper are based, are described in this section. These graphs can be divided in three families, which are presented separately in the following subsections.

4.1.1 Graph *Sottoceneri*

This graph models the main roads of the Sottoceneri region, which is the southern part of Canton Ticino (Switzerland). Interval costs have been chosen in order to cover the road conditions of all the different times of a typical day. This graph has been adopted because it represents a typical road network, and consequently a typical application to real problems of the algorithm we propose.

4.1.2 *Karaşan* graphs

The structure of these graphs is the same as that of the randomly generated benchmarks adopted in Karaşan et al. [5]. As observed in Section 1, these graphs are acyclic and layered.

Names of the graphs give some indications about their characteristics, i.e. a graph of type *K-n-c-d-w* has n vertices; each interval cost $[l_{ij}, u_{ij}]$ is obtained by generating a random number $c_{ij} \in [1, c]$ and by randomly selecting l_{ij} from $[(1-d)c_{ij}, (1+d)c_{ij}]$ and u_{ij} from $[l_{ij}, (1+d)c_{ij}]$; lastly, w is the width of the graph (see Karaşan et al. [5]).

4.1.3 *Random* graphs

This family is composed of random graphs we have generated by setting up edges between random pairs of vertices, and by assigning random interval costs to them.

Also in this case an indication of the structure of the graphs is provided by their names. A graph of type *R-n-c-δ* has n vertices, its costs are always less than c and δ is an approximation for its arc density. This family of graphs has been considered in order to

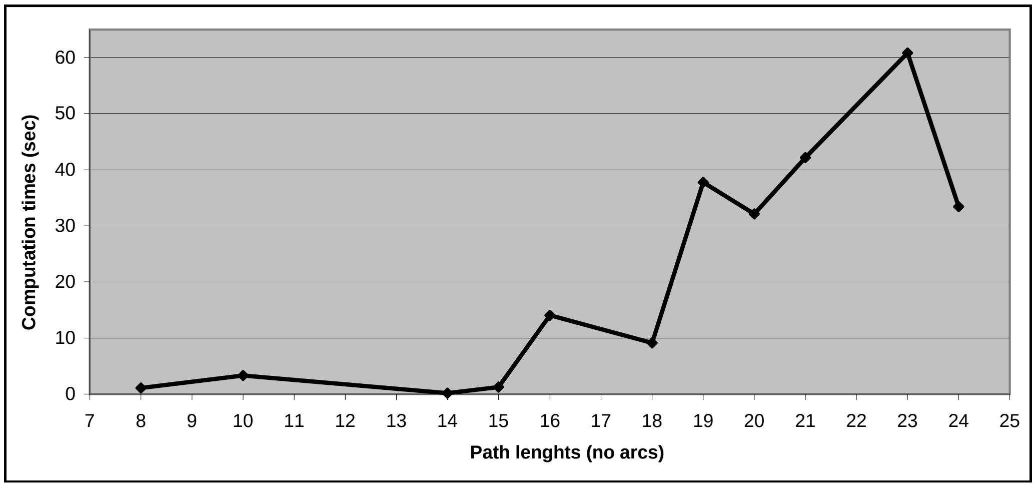

Figure 4: Correlation between path lengths and computation times on problem based on graphs of type *R-7000-100-0.0001*.

test our algorithm on problems without particular structural characteristics (i.e. different from road networks and different from acyclic, layered graphs).

4.2 Correlation between path length and algorithm performance

On graphs of families *Sottoceneri* and *Random* (and on all graphs in general), the length of the robust deviation path (in terms of number of arcs) is not known in advance. In this section we show how the performance of the algorithm we propose are correlated with the length of the robust deviation path.

We run our algorithm on graphs of type *R-7000-100-0.0001* with random pairs (s, t). In Figure 4 the results obtained are presented. (analogous results were obtained on graph *Sottoceneri*). On the x axis there are the lengths of robust deviation paths, while y axis is about computation times.

From Figure 4 it is possible to notice a correlation between the length of the robust deviation path and the time needed by our algorithm to solve the respective problem. This correlation depends mainly on two factors:

- when the robust path from s to t is long, all the paths from s to t will tend to be long, and consequently the shortest path algorithm will be slower (see Dijkstra [3]);
- when the robust path from s to t is long, more alternative paths will tend to exist between s and t, and consequently our algorithm will tend to need more iterations to converge.

4.3 Results

In this section we present the results obtained by the algorithm we propose.

For the problems based on *Karaşan* graphs, the starting vertex is always 0 and the destination vertex is always the last one (by definition, see Karaşan et al. [5]). For the problems based on *Random* graphs and on *Sottoceneri* the starting and destination points are randomly selected.

The maximum number of paths considered (parameter K in the pseudo-code of Figure 3) has been fixed at values which push the memory requirements near the limits of the computer used. In particular K has been fixed at 100000 for the tests on *Sottoceneri*, at 300000 for those on *K-90-20-0.9-2*, at 200000 for those on *K-180-20-0.9-3* and at 5000 for those on *R-7000-100-0.001* and *R-7000-100-0.0001*.

For each family of problems, 50 runs (which should be enough to overcome the effects of robust paths of different lengths, see Section 4.2), have been considered. The results are grouped in Table 1, where rows have the following meaning:

- Exact solutions (%): percentage of solutions for which optimality is confirmed by the algorithm;
- Gap (%): average gap between the lower bounds (LB) and the upper bounds (UB) retrieved by the algorithm ($\frac{UB-LB}{UB}$);
- Execution time: average total execution time (in seconds);

Table 1: Results. Averages over 50 runs.

	Sotto ceneri	*K-90-20-0.9-2*	*K-180-20-0.9-3*	*R-7000-100-0.001*	*R-7000-100-0.0001*
Exact solutions (%)	82.00	8.00	0.00	84.00	98.00
Gap (%)	6.85	58.81	88.41	2.04	0.24
Execution time	17.22188	34.67444	73.64907	103.67913	18.48302
Time last improvement	0.00284	0.00934	0.12596	0.42028	0.23530
Number of iterations	23236.84	293654.78	200000.00	1511.12	160.62
Iteration last improvement	2.56	80.86	349.64	1.10	1.06

- Time last improvement: average time (in seconds) of the last improvement to the best heuristic solution;
- Number of iterations: average total number of iterations;
- Iteration last improvement: average iteration number of the last improvement to the best heuristic solution.

Table 1 shows how the performance of the method changes when different families of problems are considered. The percentage of solutions whose optimality is confirmed is very high for problems based on *Sottoceneri* and *Random* graphs, but it is low for problems based on *Karaşan* graphs. The average gap between lower bounds and upper bounds remains large for the problems based on *Karaşan* graphs, while it is small for the other problems considered.

Observing the average execution times in Table 1, we can see that our method is always extremely fast. In particular, all the best solutions found are retrieved within half a second.

Another observation concerns the average number of iterations required to retrieve the best heuristic solutions. It is always very small, and this suggests the existence of a *correlation* between the cost of a path in scenario u and its robustness cost. This correlation confirms the intuition on which the algorithm is based, i.e. the path ranking on scenario u is also a good ranking in terms of robust deviation.

In this sense it is interesting to report that we have been able to solve to optimality, using the commercial solver *CPLEX 6.0* (see www.cplex.com), the mixed-integer program $RDSP$ for the problems of type *Sottoceneri* and *K-90-20-0.9-2* (the other problems have not been considered because it was impossible to solve them in less then 1 hour).

The results, obtained on a SUNW ULTRA-30 workstation, are very encouraging because optimality of the solution found by our algorithm has been confirmed for all of the problems considered, i.e. also for the 18% of the problems of type *Sottoceneri* and for the 92% of the problems of type *K-90-20-0.9-2* for which the algorithm we propose was not able to confirm optimality autonomously. This suggests that the quality of the heuristic solutions provided by our method is extremely high and that, on the other hand, the lower bound described in Theorem 1 is not tight enough for the problems based on *Karaşan* graphs.

The average time required by *CPLEX* to solve $RDSP$ was 1.02662 seconds for the problems of type *Sottoceneri* and 15.37751 seconds for those of type *K-90-20-0.9-2*. This means that when an exact (confirmed) solution is required, solving $RDSP$ is more convenient than running our algorithm for these two families of problems. For the other families of problems our method is however much better, because the mixed-integer program solver is simply not able to manage them in reasonable time.

It must finally be observed that using the preprocessing technique described in Karaşan et al. [5], our results would improve substantially for problems based on *Karaşan* graphs, because our algorithm would be applied on graphs with many less arcs. We have not implemented this preprocessing technique because it works for acyclic, layered graphs only, and we consider more general graphs.

The fact that our method obtains poor results on problems based on *Karaşan* graphs is however not a big limitation, because the benchmarks adopted in Karaşan et al. [5] are extremely peculiar due to their very regular structure. Real road and telecommunications networks do not present these characteristics.

We can conclude that the method we propose is competitive as an exact algorithm for large problems, for which it is not possible to solve the mixed-integer programming formulation in reasonable time. Our approach is however always very competitive as a heuristic technique, being able to find good solutions in very short times. This characteristic, which is connected with a correlation between the cost of each path in scenario u and the respective robustness cost, suggests an alternative use of the method we propose, i.e. to run it with small values of K, as a very fast heuristic algorithm.

5 Conclusion

The robust deviation shortest path problem with interval data has been studied in this paper.

After a formal description of this $\mathcal{NP}$-hard problem, an exact algorithm, which can be used as a heuristic method by truncating the execution before its natural end, has been described.

Computational results confirm the effectiveness of the approach we propose, and corroborate the intuition on which the method is based. The algorithm is able to guarantee optimal solutions for most of the problems of two of the three benchmark families considered. For the remaining problems it is able to retrieve high-quality heuristic solutions very quickly.

Acknowledgements

The authors would like to thanks Andrea Castelletti and Andrea Rizzoli for the useful discussions.

This work was co-funded by the European Commission IST project "MOSCA: Decision Support System For Integrated Door-To-Door Delivery: Planning and Control in Logistic

Chain", grant IST-2000-29557. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, 1993.

[2] L.C. Dias and J.N. Clímaco. Shortest path problems with partial information: models and algorithms for detecting dominance. *European Journal of Operational Research*, 121:16–31, 2000.

[3] E.W. Dijkstra. Note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[4] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.

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

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

[7] E.Q.V. Martins and J.L.E. dos Santos. A new shortest paths ranking algorithm. *Investigação Operacional*, 20(1):47–62, 2000.
Available at http://www.mat.uc.pt/˜eqvm/cientificos/investigacao/Artigos/K.ps.gz.

[8] H. Yaman, O.E. Karaşan, and M.Ç. Pinar. The robust spanning tree problem with interval data. *Operations Research Letters*, 29:31–40, 2001.

[9] G. Yu and J. Yang. On the shortest path problem. *Computers and Operations Research*, 25(6):457–468, 1998.

[10] P. Zieliński. The relative robust shortest path problem with interval data. Instytut Organizacji i Zarzadzania PWr., report serii PRE nr. 30, Submitted for publication to European Journal of Operational Research.
Available at http://mulhacen.ioz.pwr.wroc.pl/~pziel/publications/reports/ejor3.ps.