Journal of Applied Operational Research (2014) **6**(2), 124–133

ISSN 1735-8523 (Print), ISSN 1927-0089 (Online)

An exact algorithm for the *k*-robust shortest paths problem with interval data

Fabio Mastromatteo, Roberto Montemanni *, Luca Maria Gambardella and Andrea Emilio Rizzoli

Dalle Molle Institure for Artificial Intelligence (IDSIA), Manno-Lugano, Switzerland
University of Applied Sciences of Southern Switzerland (SUPSI), Manno-Lugano, Switzerland

Abstract. Companies operating in the textile sector aim at reducing the overall environmental impact of their products in order to obtain certification for them. This implies on optimization of the whole supply chain. The intrinsically high uncertainty that characterizes environmental impacts has to be taken into account during such an optimization. This problem can be modeled as a *k*-robust shortest paths problem with interval data, and we present an exact algorithm to tackle it. The method presented, based on some theoretical insights, is validated through some experimental results that show its effectiveness in solving problems arising in different sectors and not only in the supply chain optimization domain.

Keywords: *k*-shortest path; interval data; robust optimization; supply chain optimization; sustainability

Received October 2013. Accepted February 2014

Introduction

Within the EU-25, clothing and textiles account for approximately 3-5 % of our environmental impacts (Tukker *et al*. 2006), and globally this figure is even higher as Europe and the US have delocalised and outsourced most of the textile production, in an effort to reduce costs to the bare minimum. This has been possible at the expense of the quality of life of workers in developing countries, and of the environment, as risky and hazardous processes are subject to less stringent legislations. Recently things have been slowly changing: some segments of the customers' market are switching to the so-called *Lifestyle Of Health And Sustainability* (LOHAS, Cohen 2007). Key to LOHAS is the respect of the environment and of the workers' conditions. While in the mid to long-term we expect that LOHAS will be widely adopted, at present only some brands and sectors can afford to enter the LOHAS market, especially the prestigious fashion industry that holds high commercial gains by benefitting from the first mover advantage.

The Swiss textile and clothing industry has a tradition for quality and innovation. The Swiss customers are also sensible to environmental issues, as Switzerland has the highest per capita consumption of textiles associated with organic farming. Switzerland is therefore in the position to profile itself as the lead provider of knowledge on highly ecological, socially responsible, and competitive textile products and processes. However, in order to achieve this, we need to provide the Swiss textile industry with support tools to design and implement *sustainable supply chains*. This is the mission of the Swiss CTI project *EcoLogTex* (Montemanni *et al*. 2013). A *Supply chain designer* has to provide alternative supply chains to production managers according to the important she/he gives to different factors such as economic, time or environmental aspects. More in details, the manager will be asked to regulate the weights of the different factors involved into the optimization process leading to the most promising

* *Correspondence*: Roberto Montemanni, IDSIA/SUPSI, Galleria 2, CH-6928 Manno, Switzerland. E-mail: roberto@idsia.ch

supply chain. In such a way she/he can find the best supply chain design according to the policies of the company. Notice that such a process will normally be an iterative one, where the manager sees some proposal on the screen, modifies the weights of the different factors based on the results and the expectations and proceed likes this until she/he is satisfied. A central tool in such a context is a *k-shortest paths* calculator, able to provide the k most promising supply chains (modelled as shortest paths) according to some weighted combination of the different costs considered (Montemanni *et al*. 2013, Mastromatteo 2013). The costs considered usually cover socioeconomical and environmental impacts, while weights controlling the relative importance of the costs, are usually specified by the planner. Data such as environmental impacts are intrinsically subject to uncertainty (Finnveden *et al*. 2012), and it is important to model this inside the optimization model. For this reason *interval data* (Kouvelis and Yu 1997) are considered in our model: they have the advantage that no statistical information is required, and different tools are available in the literature.

Robust optimization based on interval data has been extensively studied in the last decades (Yu and Yang 1988, Kouveils and Yu 1997). In this paper we will present a new exact algorithm that can be seen as a generalization of a method originally introduced in (Montemanni and Gambardella 2004) for the *robust shortest path problem with interval data*, where only the best path is required, while in the present paper we look for the k best ones.

Problem description

In this paper, the robust deviation criterion (Kouvelis and Yu 1997) is applied to the k-shortest paths 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 for each arc. The target is to identify the k best shortest paths according to the robust deviation criterion. An example of graph with interval data costs 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 $c_{ij}^r \in [l_{ij}; u_{ij}]$ 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 set S of paths from s to t is said to be an optimal solution for the *k-robust (deviation) shortest paths problem* if it contains k paths with the smallest (among all paths from s to t) maximum (among all possible scenarios) robust deviation.

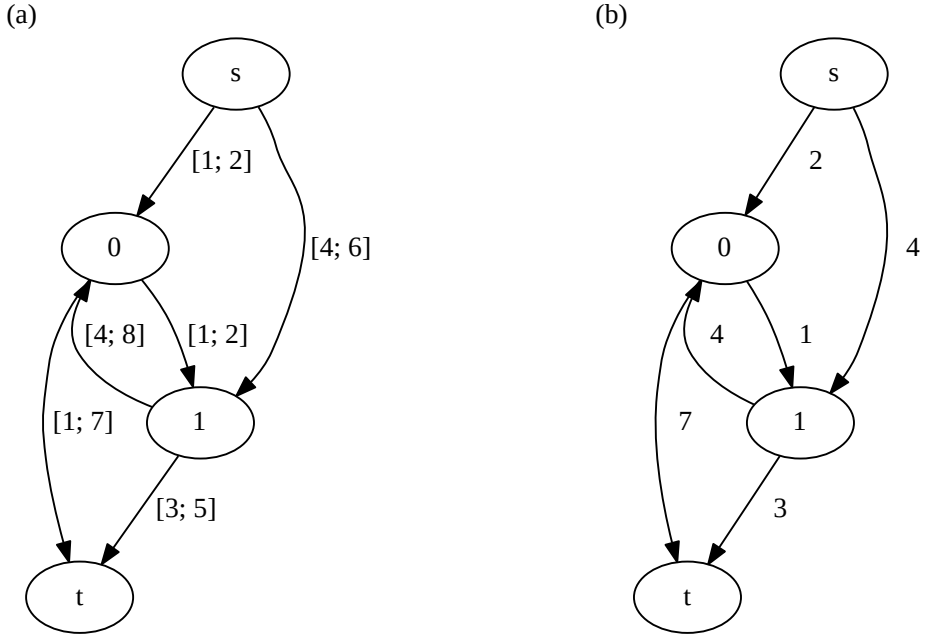

Fig. 1. (a) Example of a graph with interval data costs; (b) Scenario induced by path $p=\{s, 0, t\}$

A scenario can be seen as a snapshot of the network situation, and the *k* best robust shortest path are those typically more of interest for a decision maker in the context of the ExoLogTex project, being protected against uncertainty and consequently fulfilling the requirements – for example – of an environmental certification. More details about the mapping between the solutions of the *k*-robust shortest paths problem and the EcoLogTex *supply chain designer* can be found in Montemanni *et al*. 2013.

The following result is known from the literature:

Observation 1 (Karaşan *et al*. 2001): Given a path p from *s* to *t*, a scenario *r* which makes the robust deviation maximum is the one where each arc $(i,j) \in p$ has cost u_{ij} and each arc $(k,h) \notin p$ has cost l_{kh}.

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* in the most pessimistic scenario minus the cost of a shortest path in the scenario induced by *p* as the *robustness cost of p*. Figure 1(b) depicts the scenario induced by path $p = \{s, 0, t\}$ in the graph of Figure 1(a). The robustness cost of *p* is in this case (2+7) - (2+1+3) = 3.

A solving approach

In this paper we describe an extension of a method originally proposed in (Montemanni and Gambardella 2004) for the *robust shortest path problem with interval data*, where the objective is to retrieve only one path, while in our case we are interested in a set of paths. As far as we are aware, no other algorithm has been previously proposed for the *k*-robust shortest path problem with interval data. In detail, the approach we present is an ad-hoc exact algorithm that exploits some structural property of the problem under investigation. It is interesting to notice that this method can be used also as a heuristic algorithm, providing both upper and lower bounds for the robustness costs of the optimal paths, by stopping the computation before its natural end.

We first introduce the notation adopted in the remainder of the paper, then we present the main idea, supported by theoretical results. The algorithm is finally summarized.

Notation

- *l*: scenario in which $c_{ij}^{l} = l_{ij} \ \forall (i,j) \in A$
- *u*: scenario in which $c_{ij}^{u} = u_{ij} \ \forall (i,j) \in A$
- *P*: set of all 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*
- *UC(p)*: cost of path *p* in scenario *u*
- *LC(p)*: cost of path *p* in scenario *l*
- *RC(p)*: robust cost of path *p*
- p_i: *i*-th shortest path from *s* to *t* in scenario *u* (P_1 is the shortest path in scenario *u*)
- *kSet*: set containing the *k* paths with smallest robustness cost *RC* among those encountered up to now during the execution of the algorithm (*current solution*)

Main 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, the central idea is that when 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. This will be clear after having reviewed the following theoretical results.

Theorem 1 (Montemanni and Gambardella 2004)

$$RC(p_j) \geq UC(p_i) - UC(p_1) \quad \forall\, j \geq i \tag{1}$$

The result of Theorem 1 provides – when the paths are generated in non-decreasing order of $UC(p)$ – a lower bound for the robustness cost of the paths not yet examined. The result is used in the following Proposition 1, which is the main theoretical result of this paper and gives an effective exit criterion for the exact algorithm we present.

Proposition 1 If at iteration i of the algorithm, $UC(p_i) - UC(p_1) \geq RC(p_j) \forall p_j \in kSet$, then $kSet$ is an optimal solution of the k-robust shortest paths problem.

Proof.

From the hypothesis we have:

$$UC(p_i) - UC(p_1) \geq RC(p_j), \forall p_j \in kSet \tag{2}$$

Combining (1) and (2) we have:

$$RC(p_k) \geq RC(p_j), \forall p_j \in kSet \;\; \forall\, k \geq i \tag{3}$$

From (3), we can conlude that *kSet* contains an optimal solution for the k-robust shortest path problem.

■

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 k best paths retrieved so far, in order to decide whether to continue examining paths or to stop the computation.

According to (Martins and dos Santos 2000) ranking the first φ shortest paths in a fixed scenario is, in practice, an easy task with a computational complexity of $O(\varphi|A|)$, making our approach viable for reasonable values of φ.

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. In our implementation, we have adopted the procedure described in Dijkstra (1959), 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 (φ) has been examined (heuristic solution).

A more algorithmic description of the approach we propose will be given in the following section.

The algorithm

A pseudo-code for the algorithm summarizing the findings described in the previous section is presented in Figure 2. The algorithm starts 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 2000 is used). The robustness cost of path p_i is calculated by solving a deterministic shortest path problem (the algorithm described in Dijkstra 1959 is used) a test is carried out to check whether the new path should enter the solution set $kSet$ containing the current k-robust shortest paths. Finally, the exit condition is tested to understand whether an optimal solution has been retrieved, and the computation can be interrupted. At the end of the computation, the Boolean variable *heuristicSolution* states if the solution retrieved is heuristic of proven optimal.

```
k-RobustSP(k, φ)

// k is the number of robust paths to retrieve
// φ is the number of paths produced by the ranking (k ≤ φ)

  i := 0;
  kSet := ∅;
  heuristicSolution := false;
  while (i ≤ φ)
    i := i+1;
    Retrieve the i-th shortest path from s to t in scenario u;
    if (There are only i-1 paths from s to t in scenario u)
       return kSet;
    SP(p_i) := shortest path from s to t in the scenario induced by path p_i;
    RC(p_i) := UC(p_i) - C^{p_i}(SP(p_i));
    if (RC(p_i) < max_{p_j∈kSet}{RC(p_j)} or |kSet| < k)
      if (|kSet| = k)
        Remove argmax_{p_j∈kSet}{RC(p_j)} from kSet;
      Add p_i to kSet;
      if (UC(p_i) − UC(p_1) ≥ max_{p_j∈kSet}{RC(p_j)}) {
        return kSet;
  heuristicSolution := true;
  return kSet;
```

Fig. 2. An algorithm for the *k*-robust shortest paths problem with interval data

Computational experiments

In this section, we discuss some computational results. The algorithm has been implemented in C++ and for our experiments is run on a virtual host with a vCPU assigned from an Intel i5-3570@3.4GHz processor (reserved) and 4GB of RAM.

We will start by describing the characteristics of the graphs used for the test then we will present experiments to show a correlation between the length of the paths and the performance of the algorithm. Later we show how the value of k (the number of paths required) affects the convergence speed of the algorithm. Finally extensive tables of results will be proposed and discussed.

Description of the benchmarks

The weighted digraphs on which the benchmarks adopted in this paper are based, are described in the remainder of this section. These graphs can be divided in three families based on their characteristics. They are presented separately in the following subsection.

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 potential application to real problems of the algorithm we propose. This graph has 386 vertices and 1038 arcs.

Karaşan graphs

The structure of these graphs is the same as that of the randomly generated benchmarks adopted in Karaşan *et al*. 2001. 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, measuring the maximum number of nodes in each layer of the graph (see Karaşan *et al*. 2001). We proposed two very complex graphs (K-90-20-0.9-2, K-180-20-0.9-3) resembling telecommunications networks (Karaşan *et al*. 2001) and two graphs reproducing a typical supply chain of a textile company (K-60-100-0.5-6) and an extremely complex (*stress-test*) supply chain from the same sector (K-135-100-0.5-15), see Montemanni *et al*. 2013.

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, costs are positive integers less than or equal to c, and δ is an approximation for its arc density. This family of graphs has been considered in order to test our algorithm on problems without particular structural characteristics (i.e. different from road networks and different from acyclic, layered graphs).

Correlation between path length and algorithm performance

For instances based on the Sottoceneri and Random graphs (and on all general graphs), 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 is correlated with the length of the robust deviation path.

We run our algorithm 1000 times on graphs of type R-7000-100-0.0001 with random pairs (s, t) and $k = 1$. In Figure 3 the results obtained are presented (analogous results, with the same behavior, has been obtained on graph Sottoceneri). On the x-axis the lengths of robust deviation paths (in number of vertices) are reported, while y-axis is about computation times.

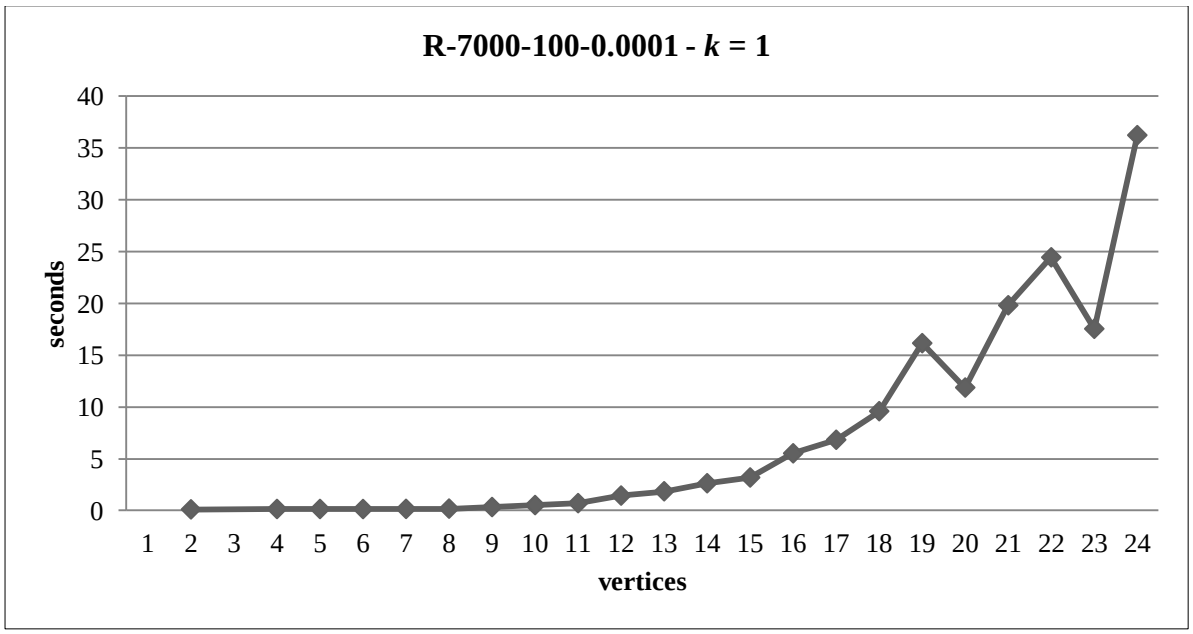

Fig. 3. Correlation between path length and algorithm performance.

From Figure 3 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. In particular, the computation time seems to increase more than linear with respect to the number of vertices in the optimal solution. This correlation intuitively depends mainly on two distinct factors:

- if the robust path from *s* to *t* is long, all the reasonably good paths from *s* to *t* will tend to be long, and consequently the shortest path algorithm will be slower (see Dijkstra 1959);
- if 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.

Correlation between *k* and the convergence speed of the algorithm

In this section we aim to show the correlation between the value of parameter *k* (number of paths required) and the convergency speed of the algorithm. We run 50 different problems for each R-7000-100-0.0001 (Figure 4) and K-60-100-0.5-6 (Figure 5) graphs for every value of *k* between 1 and 15.

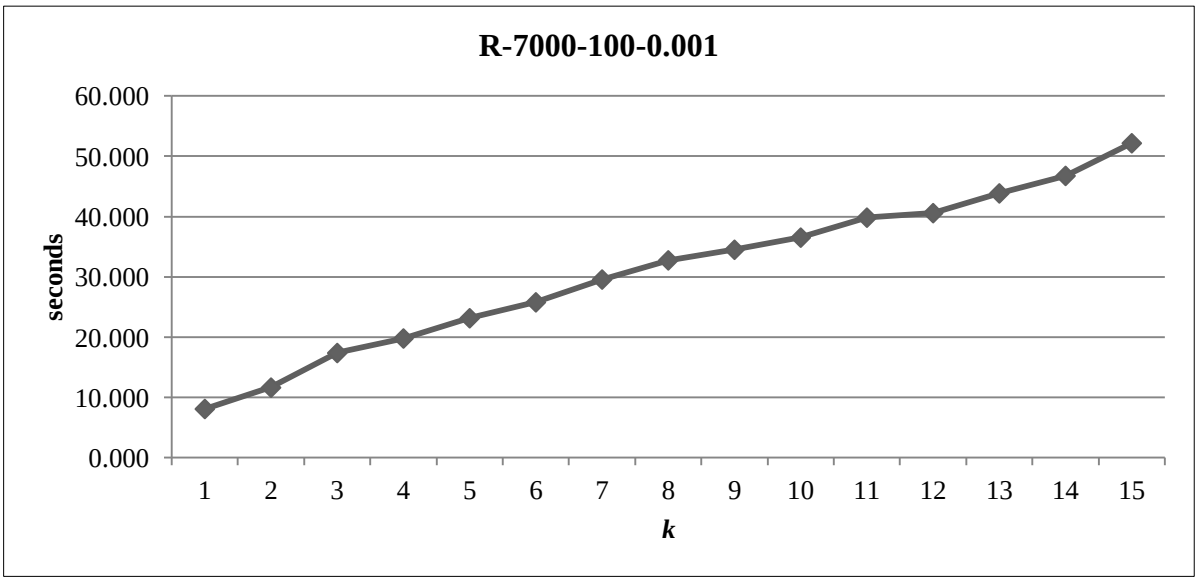

Fig. 4. Correlation between the value of *k* and the convergence speed of the algorithm.

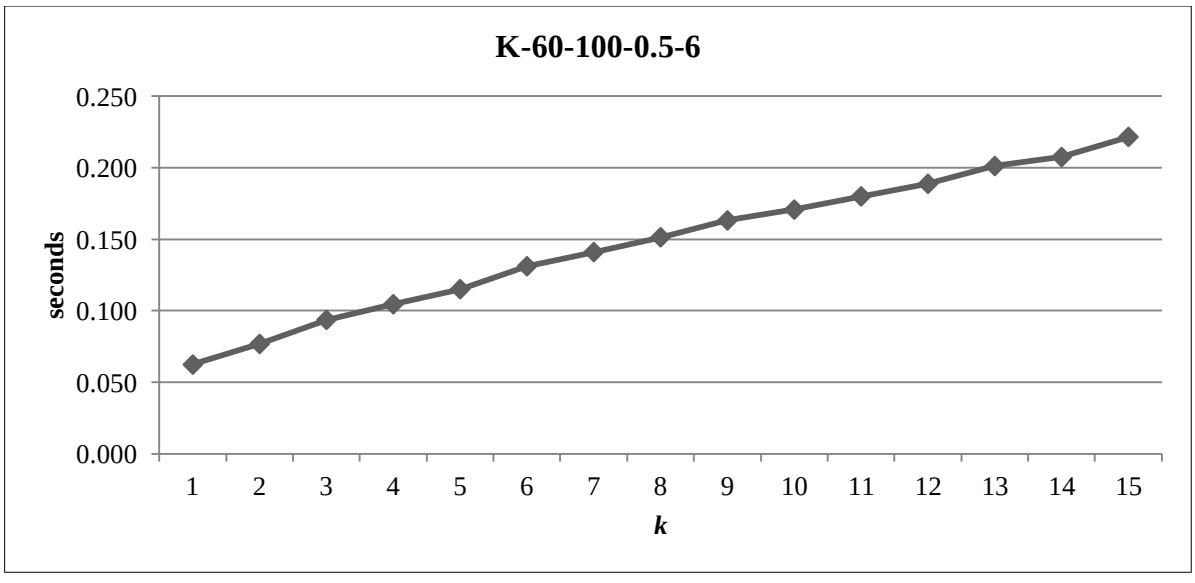

Fig. 5. Correlation between the value of *k* and the convergence speed of the algorithm.

The charts show how for both the different graphs tested, increasing the size of k leads to a clearly linear variation of the execution time of the algorithm. This property grants the possibility of using the algorithm efficiently also for future problems that may require higher values of k, and ensures confidence over the calculation times. The results also suggest that the intuition on which the algorithm is based (to examine paths according to a ranking on the so-called scenario u) makes sense, since the time required to converge is proportional to the number of paths that has to be examined by the algorithm to prove optimality.

Detailed experimental results

In this section, the detailed results obtained by the new algorithm on instances of various type will be presented and discussed. The value φ, representing the maximum paths generated by the algorithm (and reached in case optimality is not proved) is instance-depending, according to the following Table 1, where in column two the domain of each instance is also reported.

Table 1. Maximum number of paths generated by the algorithm for each instance considered.

Instance	*Domain*	φ
Sottoceneri	Transportations	100 000
K-90-20-0.9-2	Telecommunications	300 000
K-180-20-0.9-3	Telecommunications	200 000
K-60-100-0.5-6	Textile supply chain	300 000
K-135-100-0.5-15	Textile supply chain	300 000
R-7000-100-0.001	Random	2 000
R-7000-100-0.0001	Random	2 000

For Karaşan (K) instances, the starting vertex s is always that with index 0 and the destination vertex t is always the last one by definition (Karaşan *et al*. 2001). For the remaining instances (Sottoceneri and Random), s and t are chosen at random. We make tests for different values of k (3, 10, 15) and with φ values given in Table 1 for each instance considered. For each experiment, average values over 50 runs are reported in the following tables 2 to 8. Rows have the following meaning:

- *Optimal solutions (%):* Percentage of instances for which optimality has been proven by the algorithm;
- *Gap (%):* Average percentage difference between upper (UB) and lower (LB) bounds, calculated as (UB-LB)/UB;
- *Average Execution Time (sec):* Average computation time required by the algorithm, in seconds;
- *Number of Iteractions:* Average number of iterations required by the algorithm;
- *Iteraction Last Improvement:* Average iteration number in which the last change to set *kSet* happened. This measure, together with the previous one, indicates how many iterations, on average, are spent to prove optimality of a solution already settled;
- *Max Iteraction Last Improvement:* Maximum iteration number in which the last change to set *kSet* happened over the 50 runs considered. This measure, together with the previous one, measures how the algorithm is affected by differences in the instances.

Table 4. Experimental results for Sottoceneri (Transportations) and Karaşan (Telecommunications) graphs, $k = 3$.

	Sottoceneri	K-90-20-0.9-2	K-180-20-0.9-3
Optimal Solution (%)	56.00	10.00	0.00
Gap (%)	22.91	54.37	87.91
Average Execution Time (sec)	29.96	30.55	63.76
Average Number of iteractions	44 886.34	286 924.88	200 000.00
Iteraction last improvement	7.12	653.84	4 720.70
Max Iteraction last improvement	114	12 869	105 727

Table 5. Experimental results for Sottoceneri (Transportations) and Karaşan (Telecommunications) graphs, $k = 10$.

	Sottoceneri	K-90-20-0.9-2	K-180-20-0.9-3
Optimal Solution (%)	52.00	10.00	0.00
Gap (%)	24.00	54.73	87.97
Average Execution Time (sec)	32.76	34.35	64.62
Average Number of iteractions	48 299.02	286 924.88	200 000.00
Iteraction last improvement	338.04	1 125.04	6 795.28
Max Iteraction last improvement	8 609	23 640	144 188

Table 6. Experimental results for Random and Karaşan (Textile supply chain) graphs, $k = 3$.

	R-7000-100-0.001	R-7000-100-0.0001	K-60-100-0.5-6
Optimal Solutions (%)	64.00	98.00	100.00
Gap (%)	5.92	0.17	0.00
Average Execution Time (sec)	104.35	17.39	0.094
Average Number of iteractions	1 127.68	229.12	574.20
Iteraction last improvement	6.48	4.56	5.12
Max Iteraction last improvement	31	17	13

Table 7. Experimental results for Random and Karaşan (Textile supply chain) graphs, $k = 10$.

	R-7000-100-0.001	R-7000-100-0.0001	K-60-100-0.5-6
Optimal Solution (%)	34.00	96.00	100.00
Gap (%)	9.35	0.05	0.00
Average Execution Time (sec)	149.72	37.05	0.171
Average Number of iteractions	1 613.76	523.98	980.60
Iteraction last improvement	12.52	15.12	20.00
Max Iteraction last improvement	54	53	65

Table 8. Experimental results for Karaşan (Extremely complex textile supply chain) graphs, $k = 15$.

	K-135-100-0.5-15
Optimal Solution (%)	100.00
Gap (%)	0.00
Average Execution Time (sec)	10.65
Average Number of iteractions	17 900.44
Iteraction last improvement	38.58
Max Iteraction last improvement	343

From the tables we can observe how the percentage of the number of instances where we are able to prove optimality of the solutions is strictly related to the type of the graph under investigation. In particular, the method we propose is extremely effective on Random graphs and low-density Karaşan graphs (Textile supply chain), fairly effective on Sottogeneri graphs (Transportations) and not very good on dense Karaşan graphs (Telecommunications). When the optimality gap is analized, the same conclusions about the applicability of our algorithm can be drawn. It is however interesting to observe how for some problems, like R-7000-100-0.001 in Table 6,

even when the percentage of proven optimal solutions is not extremely high (64%), the average gap i slow (5.92%), suggesting that optimality is *almost* proven in most of the cases. Execution times are always extremely low (never above 2.5 minutes). This also suggests that larger values of φ could be used in case the user is interested in proven optimality and is prone su trade more computation time for optimality proofs.

From the tests reported it also emerges that many iterations are spent (on average) to prove optimality of solutions, while the optimal solution had been found many iteration before, already (comparison of rows *Number of Iteractions* and *Iteraction Last Improvement*). This implies that the idea of ranking the scenarions on scenario u and examining them in terms of robustness in this order, is promising. On the other hand, this indicates that the lower bound provided by Proposition 1 tends to stagnate and does not appear to be extremely tight. Another further implication of this phenomena, is that heuristic solutions generated with very quick run of the algorithm (small values of φ) are likely to be very close to optimality, even if this cannot be formally proven by the algorithm itself. Finally, a comparison of the last two rows suggests that sometimes improvements are found later (even if always far from the maximum number of iterations φ). This is true especially for the Telecommunications-like instances, and indicates that the algorithm is not fully robust.

A final comment has finally to be spent about the results of the algorithm on instances resembling textile supply chains (K-60-100-0.5-6 and K-135-100-0.5-15). Both small and high values of k (3 and 15) were optimally solved by the algorithms in short computation times (0.094, 0.171 seconds for typical instances and 10.65 seconds for extremely complex instances). This suggests that the method proposed in this paper can be efficiently used within the *EcoLogTex supply chain designer*, which was the main aim of our study.

Conclusions

The k-robust shortest paths problem with interval data has been studied, and an exact algorithm has been proposed. This algorithm is based on the extension of previous theoretical results found for a combinatorial optimization problem for which the k-shortest paths problem represents a generalization. The new approach has been tested on a wide selection of instances with different characteristics, ranging from transportation, telecommunication and general graphs. Of particular interest is the fact that the proposed method is able to perform extremely efficiently on graphs obtained from supply chains in the textile sector, which was the inspiring application for the present study.

Acknowledgments. The work was supported by the Swiss CTI project "EcoLogTex – Tool for Ecologic-Economic Supply Chain Analysis and Optimization in the Textile Industry".

References

Cohen MJ (2007). Consumer credit, household financial management, and sustainable consumption. International Journal of Consumer Studies 31: 57–65.

Dijkstra E (1959). Note on two problems in connection with graphs. Numerische Mathematlk 1: 269-271.

Finnveden G, Hauschild M, Ekvall T, Guinee J, Heijungs R, Hellweg S, Koehler A, Pennington D and Suh S (2012). Recent developments in Life Cycle Assessment. Journal of Environmental Management 91: 1–21.

Karaşan OE, Pinar MÇ and Yaman H (2001). The robust shortest path problem with interval data. Technical report, Bilkent University.

Kouvelis P and Yu G (1997). Robust discrete optimization and its applications. Kluwer, Dordrecht.

Martins EQV and dos Santos JLE (2000). A new shortest path ranking algorithm. Investigação Operational 20: 47-62.

Mastromatteo F (2013). Ottimizzazione robusta per supply chain del settore tessile. Bachelor dissertation, University of Applied Sciences of Southern Switzerland, Manno-Lugano, Switzerland (in Italian).

Montemanni R and Gambardella LM (2004). An exact algorithm for the robust shortest path problem with interval data. Computers & Operations Research 3: 1667–1680.

Montemanni R, Valeri C, Nesic S, Gambardella LM, Gioacchini M, Fumagalli T, Zeller H, Meyer K, Faist M and Rizzoli AE (2013). Supply chain and sustainability in the textile sector. In: Yeomans JS *et al.* (eds). Proc. 5th Int'l Conf. on Applied Operational Research, Lecture Notes in Management Science 5: 67–73.

Tukker A, Huppes G, Guinee J, Heijungs R, de Koning A, van Oers L, Suh S, Geerken T and Nielsen P (2006). Environmental impact of products (EIPro). Tech. Report EUR 22284 EN, Institute for Prospective Technological Studies, Joint Research Centre, European Commission.

Yu G and Yang J (1988). On the shortest path problem. Compuers & Operations Research 25(6): 457–468.