RECOVERABLE ROBUST SHORTEST PATH PROBLEMS

CHRISTINA PUHL*

* *Technische Universität Berlin - Fakultät II, Institut für Mathematik, Straße des 17. Juni 136, 10623 Berlin, Germany, E-mail: puhl@math.tu-berlin.de*

17.02.2009

Abstract. Recoverable robustness is a concept to avoid over-conservatism in robust optimization by allowing a limited recovery after the full data is revealed.

We investigate two settings of recoverable robust shortest path problems. In both settings the costs of the arcs are subject to uncertainty. For the first setting, at most k arcs of the chosen path can be altered in the recovery. In the second setting, we commit ourselves to a path before the costs are fully known. Deviating from this choice in the recovery comes at extra costs. For each setting we consider three different classes of scenario sets.

We show that both problems are NP-hard. For the second setting we give an approximation algorithm depending on the inflation factor and the rental factor.

1. Introduction

The shortest path problem asks for a shortest path with respect to a cost function between two designated nodes s and t in a directed graph. This problem is one of the most studied combinatorial optimization problems and can be solved efficiently in its deterministic version with nonnegative arc length. But in real-world applications like transportation, network design or telecommunication, some data might be subject to uncertainty. Nevertheless, a decision, in our setting a path, has to be taken beforehand without the knowledge of the specific scenario that occurs in the future. We expect uncertainty to be given via a set of scenarios, each defining a cost function.

There are two classical approaches when dealing with uncertainties: stochastic programming and robust optimization. In stochastic programming one assumes to have perfect knowledge about the probability distribution on the scenarios and seeks for a solution that optimizes some stochastic function. A special case, the 2-stage stochastic programming, defines a first stage decision, which is fixed for all scenarios, and a second stage decision, taken after all data are known. Together they must form a feasible solution for the scenario. The general aim is to minimize the costs for the first decision and the expected costs for the second part. For example, Minkoff [5] and Ravi [7] applied this method to the shortest path problem assuming uncertainty not only in the costs but also in the origin and destination. Yet, in practice many problems tend to be solved only once, therefore the expected value loses its relevance. Furthermore, a scenario might appear in which the total costs are much higher then the expected costs. This also results from the fact that in many applications no stochastic information is given.

Robustness addresses those two problems by neglecting the distribution and using a min-max-criterion. The robust shortest path problem has been studied,

Research supported by the Research Training Group „Methods for Discrete Structures“ (DFG-GRK 1408) and the Berlin Mathematical School.

among others, by Bertsimas and Sim [2], Yu and Yang [9] and Aissi et al. [1]. In those cases the goal is to find a path that minimizes its maximal costs over all scenarios. The drawback in those settings are the unacceptably high costs of an optimal solution. They also ignore the fact that in most problems a recovery involving at least a minor change to the previously determined solution is possible.

Recoverable robustness has been introduced by Liebchen et al. [6]. This concept combines and generalizes robust optimization and 2-stage stochastic programming. In a first stage some decision has to be made. This decision leads to first stage decision costs and limitations of the feasible solutions in the second stage. We call those the *recovery set* of a decision. In the second stage, when the scenario is known, any solution might be taken from the recovery set. For this solution the scenario costs have to be paid. An optimal recoverable robust solution is a first stage decision that minimizes the first stage costs and the maximal scenario costs by taking the best solution from its recovery set. In contrast to the robust approach, there exists no unique setting of recoverable robustness. We will introduce two settings, one in which the recovery set is very limited and one in which the decision influences the arising cost functions.

k-Arc-RRSP. A natural application of recoverable robustness to the shortest path problem is to define an (s,t)-path in the first stage, while in the second stage, a path can be chosen that differs only by k arcs from the first stage path.

Definition 1.1 (k-Arc-RRSP)**.** Let $G = (V, A)$ be a directed graph and s, t two vertices in V. Furthermore, a *first stage cost* function $c^D : A \to \mathbb{R}^+$, a set of scenarios $\mathcal{S}$, and a recovery constant $k \in \mathbb{N}$ are given. Each scenario defines *scenario costs* $c^S : A \to \mathbb{R}^+$. Let $p \in \mathcal{P}$, where $\mathcal{P}$ contains all directed (s,t)-paths in G. The *recovery set* $\mathcal{P}_p^k$ of p is defined as the set of all paths $p' \in \mathcal{P}$ with $|p' \setminus p| \leq k$ and the *robust recovery costs* $c_{RR}(p)$ as

$$c_{RR}(p) := \max_{S \in \mathcal{S}} \min_{p' \in \mathcal{P}_k^p} c^S(p').$$

An optimal solution p to the *k-Arc Recoverable Robust Shortest Path problem (k-Arc-RRSP)* minimizes the total costs $c(p)$ over all (s,t)-paths $\mathcal{P}$, where $c(p)$ is given by

$$c(p) = c^D(p) + c_{RR}(p).$$

Note that for $k = 0$ the k-Arc-RRSP is equivalent to the robust shortest path problem.

The analysis of the problem highly depends on the given scenario set. We distinguish three settings: the discrete scenario set $\mathcal{S}_D$, the interval scenario set $\mathcal{S}_I$ and the Γ-scenario set $\mathcal{S}_\Gamma$. In the *discrete scenario set* every scenario is explicitly given with its cost function [5, 7, 9]. The *interval scenario set* is an indirect description of all possible scenarios, where for each arc a a lower and an upper cost bound $\underline{c}_a$ and $\overline{c}_a$ with $0 \leq \underline{c}_a \leq \overline{c}_a$ are given. For any cost function $c : A \to \mathbb{R}^+$ with $c_a \in [\underline{c}_a, \overline{c}_a]$ for all $a \in A$, a scenario with this cost function exists in $\mathcal{S}_I$. For the *Γ-scenario set* again lower and upper cost bounds for the arc costs are given. The set $\mathcal{S}_\Gamma$ contains any scenario S, which cost function deviates at most Γ arc costs from the lower bound. This set has been introduced by Bertsimas and Sim and has been extensively studied in robust optimization [2].

We show that the decision version of the k-Arc-RRSP is weakly **NP**-complete for $|\mathcal{S}_D| = 2$ and constant k by a reduction of 2-Partition. If k is part of the input, however, we prove that the k-Arc-RRSP with $|\mathcal{S}_D| \geq 1$ is strongly **NP**-hard and inapproximable. Since the k-Arc-RRSP with interval scenario set is equivalent to the k-Arc-RRSP with just one discrete scenario, this problem is also strongly

NP-hard and inapproximable. For the special graph class of series parallel graphs we introduce a polynomial algorithm to solve the k-Arc-RRSP with $\mathcal{S}_I$.

Concerning the k-Arc-RRSP with $\mathcal{S}_\Gamma$ the computation of the total costs for a given path is already **NP**-hard, i.e., solving the problem $\max_{S\in\mathcal{S}_\Gamma} \min_{p'\in\mathcal{P}_p^k} c^S(p')$ is **NP**-hard. This problem remains **NP**-hard for $\mathcal{P}_p^k = \mathcal{P}$ due to a reduction from exact-one-in-three 3SAT. Modifying the reduction to the decision to choose the optimal first stage path shows the **NP**-hardness for constant k and gives a lower bound of $\sqrt{2}$ to the approximation factor for any efficient approximation algorithm, unless $\mathbf{P} = \mathbf{NP}$.

Rent-RRSP. Another setting for the recoverable robust shortest path problem is the Rent-RRSP. In the first stage an (s,t)-path is chosen for which rental costs, depending on the *rental factor* α and the revealed scenario, have to be payed. After the scenario is known any other path might be chosen as recovery path. For an arc a that was part of the first stage decision, we have to pay in the second stage the difference between the scenario costs and the first stage costs of this arc, i.e., $(1-\alpha)\cdot c_a^S$. For any other arc we get extra inflation costs given by the factor $(1+\beta)$.

Definition 1.2 (Rent-RRSP)**.** Let $G = (V, A)$ be a directed graph and s, t two vertices in V. Furthermore, a *rental factor* $\alpha \in]0, 1[$, an *inflation factor* $\beta \geq 0$, and a set of scenarios $\mathcal{S}$ each defining a *scenario cost function* $c^S : A \to \mathbb{R}^+$ are given. As before $\mathcal{P}$ contains all (s,t)-paths in G. For a path $p \in \mathcal{P}$ the *rent costs* $c_R^S(p)$ in scenario S are defined by $c_R^S(p) = \alpha \cdot c^S(p)$ and the *implementation costs* $c_I(p)$ by $c_I^S(p) = \min_{p'\in\mathcal{P}}(1-\alpha)c^S(p') + (\alpha+\beta)\sum_{e\in p'\setminus p} c_e^S$. The goal is to find a path with minimal *total costs* $c(p)$, defined as

$$c(p) = \max_{S\in\mathcal{S}} \left(c_R^S(p) + c_I^S(p)\right).$$

As the k-Arc-RRSP, the Rent-RRSP with $\mathcal{S}_D$ is weakly **NP**-complete for bounded $|\mathcal{S}_D| \geq 2$ and strongly **NP**-complete otherwise. The interval case is solvable in polynomial time, since any shortest path due to the upper costs $\overline{c}$ yields an optimal solution.

Furthermore, another adjustment of the reduction from exact-one-in-three 3SAT to $\max_{S\in\mathcal{S}_\Gamma} \min_{p\in\mathcal{P}} c^S(p)$ shows that the Rent-RRSP is **NP**-hard for $0 < \alpha < \frac{2}{3}$ and $3\alpha + \beta < 2$. A lower bound for the approximation factor can be given by solving a nonlinear optimization problem. We introduce a $\min(\frac{1}{\alpha}, 2+\beta)$-approximation algorithm, which is tight for $\alpha \geq 0.5$.

Overview. Section 2 covers the complexity results of the k-Arc-RRSP. In Section 3 we give an overview of the complexity results for the Rent-RRSP and the approximation algorithm for Γ-scenarios.

2. The Complexity of the k-Arc-RRSP

Discrete Scenario Sets. The decision version of the k-Arc-RRSP is in **NP**: Given an (s,t)-path p, the total costs can be calculated by solving a constrained shortest path (CSP) problem for every scenario S. The cost functions of this CSP-problem are the scenario cost function and a distance cost function

$$d(a) = \begin{cases} 0 & \text{if } a \in p \\ 1 & \text{otherwise} \end{cases}.$$

The cost function d computes $|p'\setminus p|$ for any path p' and the given (s,t)-path p and is bounded in the CSP-instance by k. In general the CSP-problem is weakly **NP**-hard and can be solved by a labeling Dijkstra in pseudo-polynomial time $\mathcal{O}(n^2L^2)$, where L is the upper bound on the second costs. Since the bound k is in our case smaller than n (otherwise the problem is trivial), this CSP-problem is solvable in

polynomial time. As in the robust setting [9], the k-Arc-RRSP problem is weakly **NP**-complete for constant k and bounded scenario set.

Theorem 2.1. *The k-Arc-RRSP is weakly* **NP***-complete for constant k and $|\mathcal{S}_D| \geq 2$.*

Proof. Let I be an instance of 2-Partition with n elements $a_1, \ldots, a_n$, $a_i \in \mathbb{N}$ and $\sum_{i=1}^{n} a_i = 2b$. The corresponding k-Arc-RRSP instance I' consists of the graph G and the scenarios S_1 and S_2. The graph G is a chain graph with $n + k$ links, each link consisting of two parallel arcs (an upper and a lower arc). The first stage cost function assigns to the last k lower arcs costs of $M > 2b$. All other arcs get costs of 0. The cost function to scenario S_1 attaches costs a_i to the i^{th} upper arc, $i = 1, \ldots, n$ and costs M to the other upper arcs. All lower arcs get 0 costs. The scenario S_2 flips the first n arc costs, i.e., the lower arcs get costs a_i and the upper ones 0 (Fig. 2.1). Choosing $M = 2b + 1$ the size of the instance I' lies polynomial in I.

In this setting any partition $A_1 \dot{\cup} A_2 = \{a_1, \ldots, a_n\}$ and can be represented by a path $p_{(A_1,A_2)}$, which uses the upper arc of the i^{th} link, if and only if $a_i \in A_1$ and the last k upper arcs. Its total costs are

$$c(p_{(A_1,A_2)}) = \max\{\sum_{b \in A_1} b, \sum_{c \in A_2} c\}.$$

On the other hand every path p with total costs smaller then M, defines a partition of the elements due to the use of upper and lower arcs. Observe that every (s,t)-path with costs smaller then M uses the last k upper arcs and exchanges them for the lower arcs in both scenarios. Therefore, it holds: The 2-Partition instance I is an yes-instance if and only if there exists an optimal solution in I' with costs b.

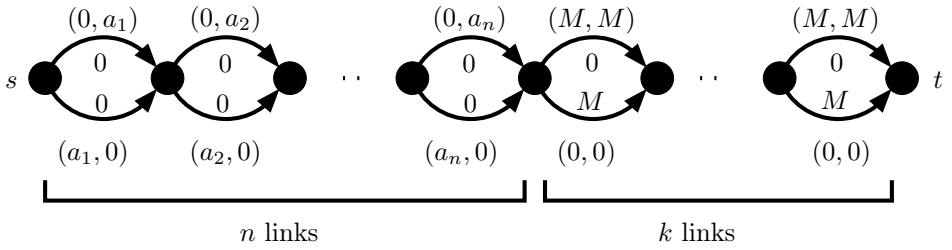

Figure 2.1. The cost functions of S_1, S_2 are given by the vector $(c_a^{S_1}, c_a^{S_2})$ for every arc, whereas the first stage costs are given as single values.

□

In [1] Aissi et al. show, that the robust shortest path problem cannot be approximated with a factor better then 0.5. This proof can be transferred to the k-Arc-RRSP to show the same lower bound on the approximability for constant k and unbounded $\mathcal{S}_D$.

Theorem 2.2. *There exists no approximation algorithm with a factor $\gamma < 2$ for the k-Arc-RRSP with unbounded $\mathcal{S}_D$ and constant k, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. We start with a reduction of the Path with Forbidden Pairs on acyclic graphs to the k-Arc-RRSP with constant k. Note that the reduction from 3SAT to Path with Forbidden Pairs induces a graph, which is acyclic (see [4]). Hence, we can restrict ourselves to this graph class, in which every (s,t)-path is a simple path. Let I be an instance of Paths with Forbidden Pairs, i.e., $G = (V, A)$

be a directed acyclic graph, $s, t \in V$, and $\mathcal{C} \subseteq V\backslash\{t\} \times V\backslash\{t\}$ a set of node pairs. The instance I' of the k-Arc-RRSP contains a graph G' and a scenario set $\mathcal{S}$. To construct G', we extend G by adding a new source s' which is connected to s via a chain of length k (Fig. 2.2). The first stage costs assign costs of 2 to all upper arcs of the added chain graph, and costs of 0 otherwise. The set of scenarios $\mathcal{S}$ contains for every pair $\{a_i, b_i\} \in \mathcal{C}$ a scenario S_i. Every scenario adds costs of 0 to the upper arc of the chain from s' to s and costs of 2 to the lower arcs. The costs for the other arcs in G' are defined in the following way:

$$c_a^{S_i} = \begin{cases} 1 & a = (a_i, v) \text{ or } a = (b_i, v)\ \forall v \in V \\ 0 & \text{otherwise} \end{cases}.$$

Due to this construction it holds: there exists a feasible path for I if and only if the optimal path for I' has costs smaller then 2.

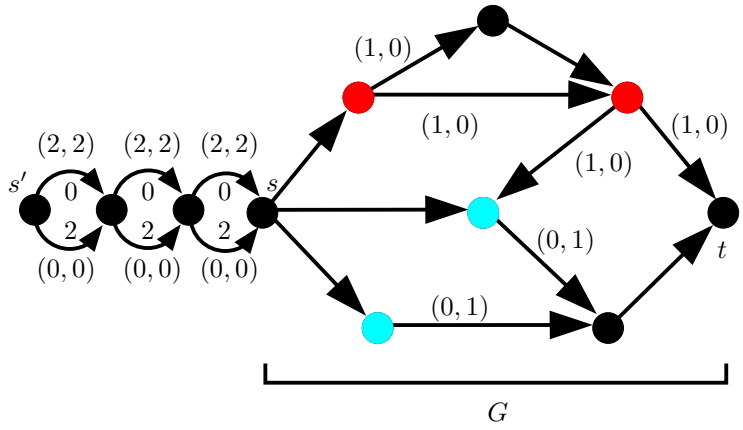

FIGURE 2.2. In the given PATH WITH FORBIDDEN PAIRS instance the red nodes and the blue nodes are the forbidden pairs.

If there exists a feasible path p in I, then the path p' using the lower arcs of the chain and the arcs of p from s to t has at most costs of 1: the first stage costs are 0, and in every scenario, the path p' can exchange the k lower arcs for the k upper arcs of the chain. Hence, in this part the costs are 0. Since p does not contain a pair $\{a_i, b_i\}$, in every scenario the costs for $p'_{[s,t]}$ are smaller then two.

On the other hand, every path with costs smaller then 2 has to use the k lower arcs in the chain and recover theses arcs in every scenario. If the optimal path p' has costs smaller then 2, then it can not contain a forbidden pair $\{a_i, b_i\}$. Otherwise the scenario costs of p' in the scenario S_i are greater or equal to 2. Due to this gap there exists no approximation algorithm with a factor $\gamma < 2$, unless $\mathbf{P} = \mathbf{NP}$. □

If k is part of the input, however, the decision version of the k-Arc-RRSP is **NP**-complete. For the optimization version no approximation algorithm exists, unless $\mathbf{P} = \mathbf{NP}$. This is due to a reduction from 3SAT, in which a feasible assignment to a given 3SAT instance exists if and only if an optimal solution of the constructed k-Arc-RRSP instance has total costs of 0.

Theorem 2.3. *The k-Arc-RRSP with one scenario, $c^D, c^S \in \{0, 1\}$ and $k \leq \frac{1}{4}|V(G)|$ is strongly* **NP***-complete. No efficient approximation algorithm exists, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. We reduce from 3SAT. Let I be an instance of 3SAT with $x_1, \dots, x_n$ variables and $C_1, \dots, C_m$ clauses,

$$C_j = y_{j1} \vee y_{j2} \vee y_{j3}$$

and w.l.o.g. $n = m$.

The graph G' of the k-Arc-RRSP instance is composed of three parts: the variable part, the clause part and the connecting part. In the variable part we introduce for each variable x_i two nodes s_i and t_i, which are connected by two paths of length two (Fig. 2.3). We call the node in the upper path the x_i *docking node* and the node on the lower path the $\overline{x}_i$ *docking node*. Furthermore, the nodes t_i and s_{i+1}, $i = 1, \dots, n-1$, and t_n with t are connected via an arc, while s is connected with s_1 via a path of length 5. All arcs in the variable part get costs of $(c^D, c^S) = (0, 1)$.

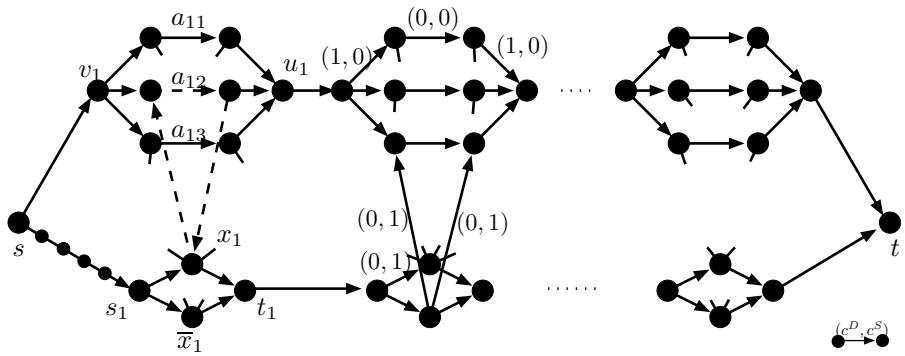

FIGURE 2.3. The lower part of the graph represents the variables of I, the upper part its clauses. The dashed arcs form the cycle $C[a_{12}, z_{12}]$ for the clause $C_1 = x_2 \vee x_1 \vee x_3$, i.e., $z_{12} = x_1$, and the literal arc a_{12}.

The clause part of G' contains for every clause C_j two nodes v_j and u_j, which are connected via three paths of length 3. We call the middle arc $a_{j\ell}$ in the ℓ^{th} path the (j, ℓ)-*literal arc*. The nodes u_j and v_{j+1}, $j = 1, \dots n-1$, s and v_1, u_n and t are connected. The literal arcs have a cost structure of $(c^D, c^S) = (0, 0)$, while all other arcs in the clause part of G' have costs of $(1, 0)$.

The last part, the connecting part of G', connects the clause part with the variable part. In general we define the cycle $C[a, b]$ with $a = (u, v)$ being an arc and b being a node as the cycle $(b, u) \cup a \cup (v, b)$ of length 3. The connecting part of G' consists of all cycles $C[a_{j\ell}, z_{j\ell}]$ with $a_{j\ell}$ being the (j, ℓ)-literal arc and $z_{j\ell}$ the $y_{j\ell}$ docking node, $j = 1, \dots, n$ and $\ell \in \{1, 2, 3\}$. The arcs in the connecting part, besides the literal arcs, get costs of $(c^D, c^S) = (0, 1)$. Hence, the graph G' contains $12 \cdot n + 6$ nodes and $21 \cdot n + 6$ arcs. For $k \leq \frac{1}{4} \cdot |V(G')|$, we are allowed to change the first stage path p_1 by at most $3n + 1$ arcs, as soon as the scenario costs reveal.

An optimal solution to I' with total costs 0, i.e., two (s, t)-paths p_1 and p_2 with $|p_2 \backslash p_1| \leq 3n + 1$, exists if and only if I is a yes-instance.

Let (p_1, p_2) be an optimal solution with total costs 0. Since p_1 has first stage costs of zero, it just contains arcs of the variable part and of the $C[a_{j\ell}, z_{j\ell}]$ cycles. Therefore, it crosses for every variable x_i either the x_i docking node or the $\overline{x}_i$ docking node. We define an assignment x according to p_1 by

$$x_i = \begin{cases} \text{true} & \text{if } p_1 \text{ crosses the } x_i \text{ docking node} \\ \text{false} & \text{if } p_1 \text{ crosses the } \overline{x}_i \text{ docking node} \end{cases}.$$

The path p_2 just crosses arcs in the clause part of G' (otherwise it induces recovery costs greater than 0). Since p_2 is a simple path, it has a length of $4n+1$ and contains exactly n literal arcs. The only arcs in the clause part of G' with first stage costs of 0, are the literal arcs. Therefore, p_1 has to crosses for every clause C_j one literal arc $a_{j\ell}$, $\ell \in \{1,2,3\}$. If the path p_1 contains a literal arc $a_{j\ell}$, then it also crosses the $y_{j\ell}$ docking node. Hence, x verifies every clause C_j, if (p_1, p_2) have total costs of 0.

Let now be x an feasible assignment to I. We define the first decision path p_1 in the following way: If $x_i = \text{true}$, the path p_1 contains the x_i docking node and all cycles $C[a_{j\ell}, z_{j\ell}]$ with $y_{j\ell} = x_i$; If $x_i = \text{false}$, the path p_1 contains the $\overline{x}_i$ docking node and all cycles $C[a_{j\ell}, z_{j\ell}]$ with $y_{j\ell} = \overline{x}_i$. This path is well defined, has first stage costs of 0 and crosses one literal arc $a_{j\ell}$ for each clause j. In addition, any simple (s,t)-path p_2 in the clause part containing those literal arcs is a feasible recovery path of p_1 with recovery costs 0. Therefore, a feasible solution (p_1, p_2) with total costs 0 exists in I', if I is a yes-instance. □

Interval Scenario Sets. Obviously the k-Arc-RRSP with interval scenario sets is equivalent to the k-Arc-RRSP with one discrete scenario, namely $S_{\max}$ with $c_a^{S_{\max}} = \overline{c}_a$. Hence, the problem can be reduced to finding a fist stage path p and a recovery path p' with $|p' \backslash p| \leq k$ minimizing $c(p) = c^D(p) + c^{S_{\max}}(p')$.

As a consequence of Theorem 2.3 the k-Arc-RRSP problem with interval scenarios is inapproximable for k being part of the input. Nevertheless, by restricting the instances to series parallel graphs, the k-Arc-RRSP with $\mathcal{S}_I$ can be solved in polynomial time. Let G be a series composition of G_1 and G_2, two series parallel graphs. Any optimal solution in G using k arcs as recovery consists of an optimal solution to G_1 using i arcs as recovery and an optimal solution to G_2 using j arcs as recovery with $i+j=k$. If G is a parallel composition of G_1 and G_2, then either the optimal first stage path p and its recovery path p' are both part of G_1 (or G_2), or p is in G_i and p' in G_j, $j \neq i$. In the second case, p is a shortest path according to c^D and p' is a shortest path according to $c^{S_{\max}}$ with a maximal length of k arcs. A decomposition of a given series parallel graph into parallel and series compositions starting from simple arcs can be computed in linear time.

Theorem 2.4. *An optimal solution of an k-Arc-RRSP with $\mathcal{S}_I$ can be calculated in polynomial time on series parallel graphs.*

Γ-Scenario Sets. Considering the k-Arc-RRSP with Γ-scenario sets the computation of the robust recovery costs for a given path p is already **NP**-hard, i.e., computing $\max_{S \in \mathcal{S}} \min_{p \in \mathcal{P}_p^k} c^S(p)$ with Γ scenarios. We call this problem the Max-Scenario-problem, which is a sub-problem of the recoverable robust shortest path problems.

Definition 2.5 (Max-Scenario-Problem)**.** Let $G = (V, A)$ be a directed graph, $s, t \in V$, and let $\mathcal{S}$ be a set of scenarios each defining a cost function $c^S : A \to \mathbb{R}_{\geq 0}$. The value value$(S)$ of a scenario S is determined through the shortest path according to c^S, i.e.

$$\text{value}(S) = \min_{p \in \mathcal{P}} c^S(p).$$

An optimal solution to the *Max-Scenario-problem* is a scenario $S \in \mathcal{S}$ with a maximal value.

The Max-Scenario-problem is easy to solve for discrete scenarios and interval scenarios. For Γ-scenarios the problem is similar to the discrete time-cost tradeoff (DTCT) problem with negative processing times and the goal to maximize the makespan. The proof for the **NP**-hardness of the DTCT [3] can be transferred to the Max-Scenario-problem with $\mathcal{S}_\Gamma$.

Theorem 2.6. *The the Max-Scenario-problem with* Γ*-scenarios is* **NP***-complete.*

Proof. In polynomial time the feasibility of any scenario can be tested and its value value(S) can be computed. Therefore, the decision version of the Max-Scenario-problem is in **NP**.

We reduce the **NP**-hard EXACT-ONE-IN-THREE 3SAT problem [8] to the Max-Scenario-problem with $\mathcal{S}_\Gamma$. Let I be an EXACT-ONE-IN-THREE 3SAT instance with $x_1, \dots, x_n$ variables and $C_1, \dots, C_m$ clauses. Each clause C_j consists of three literals $y_{j1}, y_{j2}, y_{j3} \in \{x_1, \overline{x}_1, \dots, x_n, \overline{x}_n\}$, i.e.,

$$C_j = y_{j1} \vee y_{j2} \vee y_{j3}.$$

W.l.o.g. x_i or $\overline{x}_i$ is contained in a least one clause. A *feasible solution* to I is a vector $x \in \{\text{true}, \text{false}\}^n$, such that exactly one literal in every clause is fulfilled with true. We construct a Max-Scenario-instance I' with Γ-scenarios, i.e., we define a graph G, lower and upper cost-bounds and Γ. We start with the graph G. For each variable x_i the graph G contains a fork G_{x_i} with $s_i = s$, the origin node in G. A *fork* is a graph G_{x_i} defined by three arcs $a_i, a_{x_i}, a_{\overline{x}_i}$ and four nodes $s_i, y_i, v_{x_i}, v_{\overline{x}_i}$, with $a_i = (s_i, y_i)$, $a_{x_i} = (y_i, v_{x_i})$ and $a_{\overline{x}_i} = (y_i, v_{\overline{x}_i})$. The arcs a_i and $a_{\overline{x}_i}$ are block-arcs. A *block-arc* (v, w) is an arc representing M parallel (v, w) arcs each having the same properties, e.g., the lower and upper cost-bound. We call a_i the *handle* of a fork, a_{x_i} the *true arm* of a fork and $a_{\overline{x}_i}$ the *false arm* of a fork (Fig. 2.4).

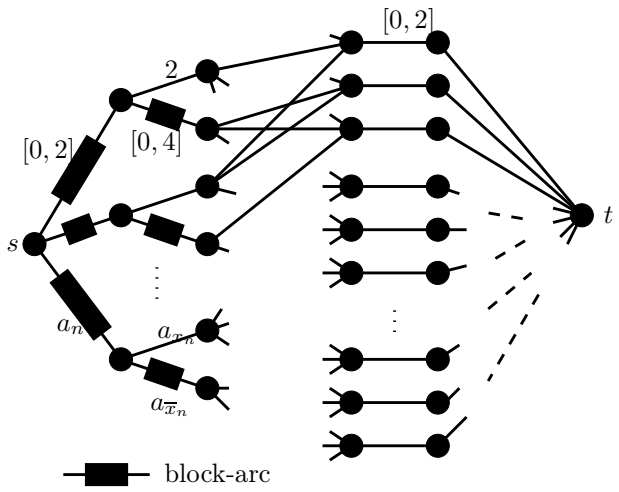

FIGURE 2.4. The arcs a_n, a_{x_n} and $a_{\overline{x}_n}$ form the fork G_{x_n}. For every clause C_j, there exist three clause-arcs a_{j1}, a_{j2} and a_{j3}.

Furthermore, G has three parallel arcs a_{j1}, a_{j2} and a_{j3} for each scenario C_j. Each arc represents a true assignment for C_j, where for a_{ji} the i^{th} literal is true. We call those arcs the *clause-arcs*. Each clause-arc is connected with t, the destination node in G. We finish the construction of G by defining the arcs between the fork arms and clause-arcs. Let a_{ji} be a clause-arc to the clause $C_j = y_{j1} \vee y_{j2} \vee y_{j3}$. For $\ell \neq i$ and $y_{j\ell} = \overline{x}_k$, we connect the true arm of the fork G_{x_k} with a_{jk} and if $y_{j\ell} = x_k$ we connect the false arm of G_{x_k} with a_{ji}. For $\ell = i$, we add an arc between the true arm of G_{x_k} and a_{ji} if $y_{ij} = x_k$. If $y_{ij} = \overline{x}_k$, we connect the false arm of G_{x_k} with a_{ji} (Fig. 2.5).

We continue with the upper and the lower cost bounds in G. The handles, the true arms and the clause-arcs get upper cost bounds of 2 and the false arms get bounds of 4. Furthermore, the lower bounds of the true arms are set to 2, i.e., the

costs of those arcs are not subject to uncertainties. Every other cost bound is set to 0 (Fig. 2.4). Note that the size of G is polynomial in the input for $M = 2m+1$. We set $\Gamma = M \cdot n + 2m$.

We will prove, that there exists a Γ-scenario S^* with value$(S^*) = 4$ in I' if and only if there is a feasible solution for the instance I.

Let x^* be a feasible solution to I. We define the cost function of S^* for all arcs with uncertainty in the following way: If x_i^* is true, S^* assigns upper costs to the handle of G_{x_i} and lower costs to the false arm. If x_i^* is false, the false arm gets the upper costs and the handle the lower costs. Note that any (s,t)-path already has a length of 2 due to this cost assignments. Since x^* is a feasible solution, in every clause C_j, there exists exactly one literal $i(j) \in \{1,2,3\}$ which has a true assignment. Scenario S^* puts the costs of all clause-arcs a_{ji} with $i \neq i(j)$ to their upper bounds and leaves the costs of $a_{ji(j)}$ at the lower bound (Fig. 2.5). In total S^* changes n block-arcs and $2m$ clause-arcs, i.e., $n \cdot M + 2m$ arc costs. Therefore, S^* is a Γ-scenario. It remains to show that any shortest path in G with c^{S^*} has a length of 4.

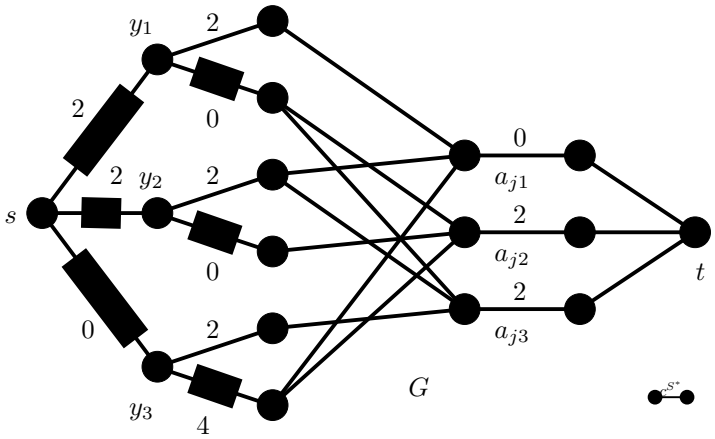

FIGURE 2.5. This graph G is constructed for the instance I with $C_1 = x_1 \vee \overline{x}_2 \vee x_3$. The scenario S^* to a feasible solution $x^* = (\text{true}, \text{true}, \text{false})$ has value$(S^*) = 4$. In C_j the first variable x_1 verifies the clause. Therefore, the costs of a_{j1} are not raised.

Assume that there exists a path p with costs of 2. Then p has to cross a clause-arc $a_{ji(j)}$, since all paths to a clause-arc have already a length of 2. If $y_{ji(j)} = x_\ell$, then x_ℓ has a true assignment. Therefore, any path traversing the true arm of G_{x_ℓ} has, due to the definition of S^*, length of 4 or more. The same argument works for $y_{ji(j)} = \overline{x}_\ell$. If $y_{ji} = x_\ell$ for $i \neq i(j)$, then $a_{ji(j)}$ is connected to the false handle of G_{x_ℓ}. Since the literal y_{ji} is false, the variable x_ℓ^* is set to false. Therefore, any path crossing this arm, has length of at least 4. The same conclusions are valid for $y_{ji} = \overline{x}_\ell$. Hence, paths traversing $a_{ji(j)}$ have already a length of 4 before they pass the clause-arc. This is a contradiction.

Let S^* be a Γ-scenario in I' with value$(S^*) = 4$. Before we start with a construction of x^*, we need some observations.

1. Observation: The scenario S^* assigns in every fork exactly one block-arc to the upper cost bound.

Proof: Assume that there is a fork G_{x_ℓ} in which no block-arc is assigned to $\overline{c}$. Then in the handle block-arc and in the false arm block-arc exists an arc with costs of 0. An (s,t)-path traversing these two arcs has at most costs of 2. This is

a contradiction to value(S^*) = 4. Since $2m < M$, at most n block-arcs can have upper bound costs. △

2. Observation: Exactly two clause-arcs of each clause are moved to their upper bounds.

Proof: Assume there exists a clause C_j, in which only one clause-arc is changed to the upper costs. Each one of the three clause-arcs a_{j1}, a_{j2} and a_{j3} is connected to the same forks G_{x_a}, G_{x_b} and G_{x_c}. Since in every fork one of the block-arcs has been assigned to the upper costs, either a shortest path to the end of the true arm or a shortest path to the end of the false arm has length of 4. The other one has length of 2. Let a_{j1} w.l.o.g. be the one clause, in which the costs have been moved up. Since the shortest path from s to t has a length of 4 and the other two clause-arcs a_{j2} and a_{j3} have costs of 0, both must be connected to the three arms with the higher costs (Fig. 2.6). This is a contradiction to the construction of G.

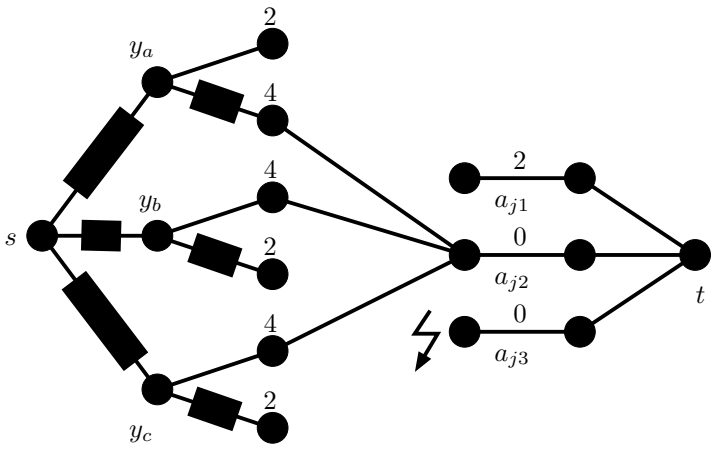

FIGURE 2.6. If a scenario S moves just one of three clause-arcs, then there exists an (s,t)-path in G of length 2.

Since S^* already changed $n \cdot M$ arc costs, there are just $2m$ possibilities left; two for every clause. △

Now we define a solution x^* to the scenario S^*

$$x_i^* = \begin{cases} \text{true} & \text{if } c^{S^*}(a_i) = 2 \\ \text{false} & \text{otherwise} \end{cases}.$$

For every clause C_j, there is one clause-arc $a_{ji(j)}$ with costs 0. W.l.o.g. $i(j) = 1$. If $y_{j1} = x_\ell$, then a_{j1} is connected to the true arm. Every path crossing this arm has to have a length of 4. Therefore, the handle arc a_ℓ has to have costs at the upper bound and hence $x_a^* = \text{true}$. The same argumentation works for $y_{j1} = \overline{x}_a$. Furthermore, for $y_{ji} = x_{b_i}$ or $y_{ji} = \overline{x}_{b_i}$ with $i \in \{2,3\}$ the two variables are set such that they neglect the clause. Hence, x^* is a feasible solution.

This completes the proof of the **NP**-completeness of the Max-Scenario-problem. □

In the following we denote the graph G of the reduction from an EXACT-ONE-IN-TREE 3SAT instance I as G_I. Furthermore, we just summarize the most important facts about the reduction: Let I be an instance of EXACT-ONE-IN-THREE 3SAT. We can construct a graph G_I with some cost uncertainties modeled by the intervals $[0,2]$ and $[0,4]$ and $c^D = 0$, such that

(1) if there exists a scenario $S \in \mathcal{S}$ with $\min_{p\in\mathcal{P}} c^S(p) = 4$, the instance I is a yes-instance
(2) if for every $S \in \mathcal{S}$ there exists a path with costs at most 2, the instance I is a no-instance
(3) every simple (s,t)-path in G_I has a length of 4.

In this reduction any scenario is allowed to change almost half of all uncertain values to the upper interval costs.

Adding one (s,t) arc a_1 to G_I with fixed scenario costs 6, it is **NP**-hard to compute the total costs of $p = a_1$ for $k \geq 4$: In this case $\mathcal{P}_p^k$ contains all (s,t)-paths in G_I. For any scenario $S \in \mathcal{S}_\Gamma$ the shortest path according to c^S has at most costs of 4. Hence, it is better to switch in the second phase to this path. The total costs of p are equal to $6 + 4$ if and only if I is a yes-instance. Therefore, the costs of a given path in the k-Arc-RRSP cannot be efficiently calculated, if $\mathbf{P} \neq \mathbf{NP}$. Since for a decision problem in **NP** the costs of any feasible solution have to be calculated in polynomial time, the decision version of the k-Arc-RRSP is not in **NP**.

Theorem 2.7. *The decision version of the* k-Arc-RRSP *with* $\mathcal{S}_\Gamma$ *is not in* **NP**, *unless* $\mathbf{P} = \mathbf{NP}$.

An exact algorithm for the k-Arc-RRSP optimization problem constructs an optimal path, but not the total costs of this path. Nevertheless, the following theorem holds:

Theorem 2.8. *For constant* $k \geq 4$ *the* k-Arc-RRSP *with* $\mathcal{S}_\Gamma$ *is strongly* **NP***-hard.*

Proof. We reduce from exact-one-in-tree 3SAT. Let I be an exact-one-in-tree 3SAT instance. To construct an instance of the k-Arc-RRSP problem I', we add a simple (s',s)-path $p_{(s',s)}$ with length $k-3$ and an (s',t)-path $p_{(s',t)}$ of length $k+1$ to G_I (Fig. 2.7). The first stage costs adds costs of 5 to the first $k-3$ arcs on the $p_{(s',s)}$ and the $p_{(s',t)}$ path. The cost bounds on the scenario costs are shifted by 4 for all arcs in G_I, i.d. the costs of the fork arms are changed from $[0,2]$ to $[4,6]$, from 2 to 6, from $[0,4]$ to $[4,8]$, the scenario arcs are changed from $[0,2]$ to $[4,6]$, and all other arcs get costs of 4. The lower bounds also define the first stage costs. Note that the scenario costs of any (s,t)-path increases by 16 compared to the costs of this path with the original scenario costs. The last tree arcs on the $p_{(s',t)}$ path get costs of $[4,5]$ and the last arc the costs of 4. The cost function and all scenario costs satisfy the $\alpha_{[0,1]}$-deviation condition and the size of I' is polynomial in I.

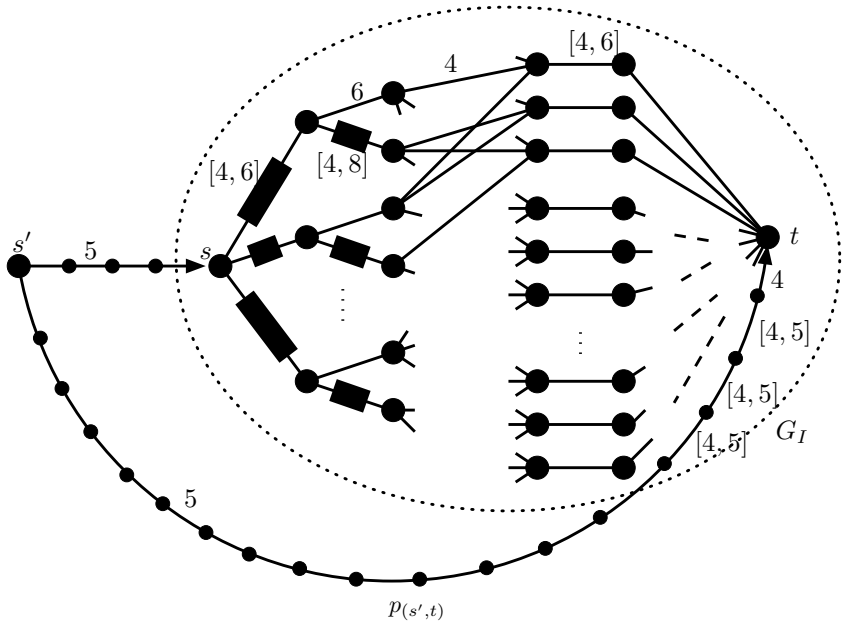

FIGURE 2.7. The first stage costs are defined by the lower bound of the interval costs.

To show that the optimal solution of I' is the simple (s',t)-path $p_{(s',t)}$ if and only if I is a Yes instance, we need three observations:

- Every simple (s',t)-path has a length of $k+1$ and first stage costs of $\overline{c} = 5 \cdot (k-3) + 16$.
- Since $k \geq 4$, the recovery of a simple (s,t)-path $p \neq p_{(s',t)}$ contains all paths crossing s.
- The recovery of $p_{(s',t)}$ consists just of $p_{(s',t)}$.

If I is a yes-instance, the (s',t)-path $p_{(s',t)}$ has total costs of $\overline{c} + 5 \cdot (k-3) + 19$, while any other simple path has recovery costs of $\overline{c} + 5 \cdot (k-3) + 20$. Any other first decision $G' \in \mathcal{G}$ has at least costs of $(\overline{c}+4) + 5 \cdot (k-3) + 19$. On the other hand, if I is a No-instance, any simple (s',t)-path crossing s gets costs of $\overline{c} + 5 \cdot (k-3) + 18$, while $p_{(s',t)}$ has total costs of $\overline{c} + 5 \cdot (k-3) + 19$ and any other subgraph in $\mathcal{G}$ has costs of at least $(\overline{c}+4) + 5 \cdot (k-3) + 18$. Hence, $p_{(s',t)}$ is an optimal solution of the k-Arc-RRSP instance I' if and only if I is a yes-instance.

Note that for k being part of the input the problem is inapproximable due to Theorem 2.3. □

3. The Rent-RRSP And Its Complexity

Discrete Scenario Sets. The Rent-RRSP with $\mathcal{S}_D$ is weakly **NP**-hard for $\alpha > 0$ and $|\mathcal{S}_D| = 2$ and strongly **NP**-hard for unbounded $|\mathcal{S}_D|$.

Theorem 3.1. *The* Rent-RRSP *with* $\mathcal{S}_D$ *is strongly* **NP***-hard for* $\alpha > 0$ *and weakly* **NP***-hard for* $|\mathcal{S}| = 2$.

The result of weak **NP**-hardness is due to a reduction from 2 − Partition, similar to the reduction to Theorem 2.1. The reduction graph G is again a chain graph with n links of two parallel arcs, where n is the number of elements of the 2-Partition instance. Scenario S_1 assigns costs a_i to the i^{th} upper arc, 0 otherwise. The other scenario S_2 flips these costs, i.e., the upper arcs get 0 costs. Note that for any scenario S_i there exists a path p_i with costs 0. Hence, for every path and

every scenario the implementation costs are equal to 0. Therefore, the total costs of a path p reduce to the rent costs $c(p) = \max_{S\in\mathcal{S}} \alpha \cdot c^S(p)$. The strong **NP**-hardness results from a similar reduction – this time from 3-Partition. Furthermore, we can replace any arc a of costs $\tilde{a}_i = \max c_a^{S_i}$ by a path of length $\tilde{a}_i$. Hence the problem remains strongly **NP**-hard for $c_a^S \in \{0, 1\}$.

A lower bound on the approximation factor can also be achieved by a reduction from Path with Forbidden Pairs.

Theorem 3.2. *There exists no approximation algorithm with a factor $\gamma < 2$ for the* Rent-RRSP *with $\mathcal{S}_D$ and $\alpha > 0$, unless $\mathbf{P} = \mathbf{NP}$.*

Proof. Reduction from Path with Forbidden Pairs. Let I be an instance of Paths with foridden pairs, i.e., $G = (V, A)$ be a directed graph, $s, t \in V$, and $\mathcal{C} \subseteq V\backslash\{t\} \times V\backslash\{t\}$ a set of node pairs. For every pair $\{v_i, u_i\} \in \mathcal{C}$ we add an (s, t)-arc a_i to the graph G and a scenario S_i to $\mathcal{S}$ to define a Rent-RRSP instance I'. The scenario S_i has the cost function

$$c_a^{S_i} = \begin{cases} 1 & a = (v_i, v) \text{ or } a = (u_i, v)\ \forall v \in V \\ 0 & \text{otherwise.} \end{cases}.$$

Due to this construction, in any scenario there exists a path of costs 0. Hence the total costs of a path reduce to

$$c(p) = \max_{S\in\mathcal{S}} \alpha \cdot c^S(p).$$

With the same arguments as in the proof to Theorem 2.2 a shortest path in I' has costs less or equal $\alpha \cdot 1$ if and only if there exists a feasible path to I. Otherwise the shortest path in I' has a length of greater or equal $\alpha \cdot 2$. □

Interval Scenario Sets. In the Rent-RRSP with interval scenario sets the scenario $S_{\max}$ with $c_a^{S_{\max}} = \overline{c}_a$ dominates all other scenarios. Hence, any shortest path in terms of this cost function yields an optimal solution for the Rent-RRSP.

Theorem 3.3. *Any shortest path in terms of the cost function $\overline{c}$ yields an optimal solution for the* Rent-RRSP*.*

Proof. Let $p_{\max}$ be a shortest path due to the cost function $\overline{c}$ and $p \in \mathcal{P}$. Then

$$\begin{aligned} c(p) &= \max_{S\in\mathcal{S}} \min_{p'\in\mathcal{P}} [\alpha c^S(p) + (1-\alpha)c^S(p') + (\alpha+\beta) \sum_{a\in p'\setminus p} c_a^S] \\ &\geq \min_{p'\in\mathcal{P}} [\alpha\overline{c}(p) + (1-\alpha)\overline{c}(p')] \\ &\geq \overline{c}(p_{\max}) = c(p_{\max}) \end{aligned}$$

□

Note that in this case the recovery option is not used at all.

Γ-Scenario Sets. As stated in Theorem 2.7 for the k-Arc-RRSP, in the Rent-RRSP the total costs for a given path are strongly **NP**-hard to compute. The reduction is based on the proof mentioned in Section 2 for solving $\max_{S\in\mathcal{S}_\Gamma} \min_{p\in\mathcal{P}} c^S(p)$. But even without returning the total costs for a given path, the problem remains **NP**-hard.

Theorem 3.4. *The* Rent-RRSP *with $\mathcal{S}_\Gamma$ is strongly* **NP***-hard for $0 < \alpha < \frac{2}{3}$ and $3\alpha + \beta < 2$.*

Proof. We reduce from exact-one-in-three 3SAT. Let I be an instance of that problem with n variables and m clauses containing the variables x_q, x_r, x_s which are only used in the clauses $C_a = x_q \vee x_r \vee x_s$, $C_b = \overline{x}_q \vee \overline{x}_r \vee x_s$ and $C_c = \overline{x}_q \vee x_r \vee \overline{x}_s$. These three clauses are also part of I. Note that the only assignment verifying those

clauses is x_q = true, x_r = false and x_s = false. Recall from the proof of Theorem 2.6, that for every variable, the graph G_I contains a *fork* (Fig. 2.4). A fork consists of two block-arcs, which represent M parallel arcs with the same cost structure as for the block-arc, and one normal arc. One of the block-arcs and a normal arc represent the two *fork arms*, the other block-arc the *handle*. Furthermore, G_I contains for every clause three parallel arcs, each one representing a feasible assignment to the variables for this clause. We call those $3m$ arcs *clause-arcs*. Each one of those clause-arcs is connected to three fork arms and to t. Remember that there exists a scenario $\tilde{S}$ with a cost function such that the shortest path has length 4 if and only if I is a yes-instance. The scenario $\tilde{S}$ is allowed to have at most a little more than half of all arc costs at their upper bounds, i.e., $n \cdot M + 2m$. To this graph G_I we add an arc (s,t) with fixed costs $c_{(s,t)} = a$ and $6 > a > \max\{4, 6\alpha + 2 \cdot (1+\beta)\}$, denoting (s,t) also as path $\tilde{p}$. Since $3\alpha + \beta < 2$, such a value a exists.

1. Observation: If I is a no-instance, then the total costs for every path $p \in \mathcal{P}\backslash\{\tilde{p}\}$ are bounded by

$$c(p) \geq \alpha \cdot 6 + 2 \cdot (1+\beta).$$

Proof: We define the scenario S_p to a path p in the following way: raise the costs of every clause-arc, of all block-arcs (i.e., all M parallel arcs) the path p is crossing, and of all fork handles which are connected to the clause-arc p traverses. Altogether scenario S_p changes the costs of at most $4M + 3m$ arcs. In this scenario the rent costs of p are $c_R^{S_p}(p) \geq \alpha \cdot 6$ and for every other path we have at least implementation costs of $2 \cdot (1+\beta)$. Since $3\alpha + \beta < 2$ and I is a no-instance, i.e., in every scenario there exists a shortest path of length 2, the minimal rent and implementation costs for S_p are $\alpha \cdot 6 + 2 \cdot (1+\beta)$. △

2. Observation: If I is a yes-instance, there exists a path $\overline{p} \in \mathcal{P}\backslash\tilde{p}$ with total costs

$$c(\overline{p}) = \max\{\alpha \cdot 6 + 2 \cdot (1+\beta), 4\}.$$

Proof: Consider the path $\overline{p}$ crossing the handle of G_{x_r}, the true arm of G_{x_r} and the clause-arc a_{a1}. We divide all scenarios $S \in \mathcal{S}_\Gamma$ according to their cost assignment to this path, i.e., $\mathcal{S}_{\Gamma,6}$ contains all scenarios with $c^S(\overline{p}) = 6$, etc. We assume there exists a scenario $S \in \mathcal{S}_{\Gamma,6}$ with $c^S(p') \geq 4$ for all $p' \in \mathcal{P}\backslash\{\tilde{p}\}$. Hence, this scenario defines a true assignment to I, as shown in the proof of Theorem 2.6. But the only valid assignment sets x_r = false. Therefore, S has to move the costs of the false arm to the upper cost bound. Since $S \in \mathcal{S}_{\Gamma,6}$, at least one arc of the handle has to be moved to the upper cost bound. This is a contradiction to observation 1 in Theorem 2.6. Hence, for all $S \in \mathcal{S}_{\Gamma,6}$ there exists a path p' with $c^S(p') = 2$.

If $S \in \mathcal{S}_{\Gamma,4}$, the total costs of $\overline{p}$ are 4 and if $S \in \mathcal{S}_{\Gamma,2}$, the total costs are 2. Thus,

$$c(\overline{p}) = \max\{\alpha \cdot 6 + 2 \cdot (1+\beta), 4\}.$$

△

3. Observation: If I is a no-instance, the path $\tilde{p}$ has total costs

$$c(\tilde{p}) = \alpha \cdot a + 2 \cdot (1+\beta)$$

and if I is a yes-instance, the total costs are

$$c(\tilde{p}) = \min\{a, \alpha \cdot a + 4 \cdot (1+\beta)\}.$$

Proof: If I is a no-instance, in every scenario S exists a path $p \in \mathcal{P}\backslash\{\tilde{p}\}$ with costs 2. Since $a > \alpha \cdot 6 + 2 \cdot (1+\beta)$ but $a < 6$, the total costs of $\tilde{p}$ are

$$c(\tilde{p}) = \alpha \cdot a + 2 \cdot (1+\beta) < \alpha \cdot 6 + 2 \cdot (1+\beta) < a.$$

If I is a yes-instance, there exists a scenario S^* with all paths $p \in \mathcal{P}\backslash\{\tilde{p}\}$ having at least costs of 4. A scenario with all of those paths having length of 6 does not

exist. Therefore, the total costs of $c(\tilde{p})$ are

$$c(\tilde{p}) = \min\{a, \alpha \cdot a + 4 \cdot (1+\beta)\}.$$

△

Due to the previous three observations we get: If I is a no-instance, $\tilde{p}$ is the optimal solution, i.e.,

$$c(\tilde{p}) = \alpha \cdot a + 2 \cdot (1+\beta) < \alpha \cdot 6 + 2 \cdot (1+\beta) \leq c(p) \quad \forall p \in \mathcal{P} \backslash \{\tilde{p}\}.$$

If I is a yes-instance, due to the restrictions on a, the path $\tilde{p}$ is not an optimal solution:

$$\begin{aligned} c(\tilde{p}) &= \min\{a, \alpha \cdot a + 4 \cdot (1+\beta)\} \\ &> \max\{\alpha \cdot 6 + 2 \cdot (1+\beta), 4\} \\ &= c(\overline{p}), \end{aligned}$$

with $\overline{p}$ defined as in the second observation. Therefore, any exact algorithm for the RENT-RRSP solves the EXACT-ONE-IN-THREE 3SAT. □

Since an optimal solution cannot be constructed efficiently if $\mathbf{P} \neq \mathbf{NP}$, we are interested in an approximation algorithm. An approximation algorithm constructs a first solution $p \in \mathcal{P}$ and gives for every first solution p and scenario $S \in \mathcal{S}$ a recovery strategy, i.e., a rule how to compute the second solution.

Algorithm 1 Optimal Recovery

Input:	Directed graph $G = (V, A)$, $s, t \in V$, $\underline{c}_a$ and $\overline{c}_a$ $\forall a \in A$, Γ, α, β.
Output:	First decision path p and recovery strategy.
1. Step:	Calculate $p \in \mathcal{P}$ with $p = \arg\min_{p \in \mathcal{P}} \max_{S \in \mathcal{S}_\Gamma} c^S(p)$.
Recovery:	Choose the shortest path $p' \in \mathcal{P}$ to the cost function

$$\tilde{c}_a = \begin{cases} (1-\alpha)c_a^S & \forall a \in p \\ (1+\beta)c_a^S & \forall a \notin p \end{cases}.$$

Theorem 3.5. *The approximation algorithm* Optimal Recovery *calculates a solution p of total costs*

$$c(p) \leq \min\{(2+\beta), \frac{1}{\alpha}\} \cdot \mathrm{OPT}\,.$$

Proof. The problem $\min_{p \in \mathcal{P}} \max_{S \in \mathcal{S}_\Gamma} c^S(p)$ is a robust shortest path problem with Γ-scenario sets. This problem is solvable in polynomial time by solving $m+1$ shortest path problems [2].

Two lower bounds for the optimal solution are

$$\begin{aligned} \mathrm{OPT} &= \min_{p \in \mathcal{P}} \max_{S \in \mathcal{S}} \min_{p' \in \mathcal{P}} [\alpha c^S(p) + (1-\alpha)c^S(p') + (\alpha+\beta) \cdot \sum_{a \in p' \backslash p} c_a^S] \\ &\geq \alpha \cdot \min_{p \in \mathcal{P}} \max_{S \in \mathcal{S}} c^S(p) \end{aligned}$$

and

$$\mathrm{OPT} \geq \max_{S \in \mathcal{S}} \min_{p' \in \mathcal{P}} c^S(p').$$

On the other hand, in each scenario S

$$c_R^S(p_A) \leq \alpha \cdot \min_{p \in \mathcal{P}} \max_{S \in \mathcal{S}} c^S(p) \quad \text{and} \quad c_I^S(p_A) \leq (1+\beta) \cdot \min_{p' \in \mathcal{P}} c^S(p').$$

Therefore, the total costs for the path p_A calculated by Algorithm 1 are bounded by

$$c(p_A) = \max_{S\in\mathcal{S}}[c_R^S(p_A) + c_I^S(p_A)] \leq \mathrm{OPT} + (1+\beta)\cdot\mathrm{OPT} = (2+\beta)\cdot\mathrm{OPT}\,.$$

Since the recovery chooses the shortest path of the cost function $\tilde{c}$ defined in Algorithm 1

$$c(p_A) \leq \min_{p\in\mathcal{P}} \max_{S\in\mathcal{S}} c^S(p_A) \leq \frac{1}{\alpha}\cdot\mathrm{OPT}\,.$$

Together those bounds give an an approximation factor of $\min\{\frac{1}{\alpha}, (2+\beta)\}$. □

For $\alpha \geq 0.5$ the approximation factor is tight. In the Rent-RRSP instance given by a graph G composed of an (s,t)-arc with the cost-interval $[0,1]$, also denoted as path $\tilde{p}$, and a path p from s to t with two arcs, each one having a cost interval of $[0,0.5]$, and $\Gamma = 2$, the algorithm 1 could choose path $\tilde{p}$. This results in total costs $c(\tilde{p}) = \min\{1, \alpha + (1+\beta)\frac{1}{2}\}$ whereas the path p yields the optimal costs $c(p) = \max\{\alpha, 0.5\}$. For $\alpha \geq 0.5$ we get $\mathrm{ALG} \leq \frac{1}{\alpha}\cdot\mathrm{OPT}$.

4. Conclusions

We considered two different settings for the recoverable robust shortest path problem and investigated their complexity with respect to the most common scenario settings in literature. For all those settings, and k being part of the input, the k-Arc-RRSP problem is strongly **NP**-hard and not approximable, unless $\mathbf{P} = \mathbf{NP}$. If k is constant, we introduce a polynomial algorithm to solve the problem with interval scenarios on series parallel graphs. Whether a polynomial algorithm for general graphs exists, or the problem is **NP**-hard with interval scenarios, should be the focus of further research.

For Γ-scenario sets the k-Arc-RRSP and the Rent-RRSP are **NP**-hard optimization problems. The strict robust version on the other hand, can be solved in polynomial time. We give an approximation algorithm for the Rent-RRSP, which chooses a robust shortest path as first stage decision. In the recovery stage either this path or any other with less costs is taken. Therefore, in any application it is better to use the recovery than to stay with the robust solution.

Acknowledgment

We thank Christian Liebchen, Rolf H. Möhring, Martin Skutella and Sebastian Stiller for fruitful discussions.

References

[1] H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation complexity of min-max (regret) versions of shortest path, spanning tree, and knapsack. *Proceedings of the 13th Annual European Symposium on Algorithms (ESA 2005)*, pages 862–873, 2005.

[2] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming B*, 98:49–71, 2003.

[3] P. De, J. Dunne, and Ch. Wells. Complexity of the discrete time-cost tradeoff problem for project networks. *Operations Research*, 45(2):302–306, 1997.

[4] H. Gabow, S. Maheshwari, and L. Osterweil. On two problems in the generation of program test paths. *IEEE Transactions on software engineering*, SE-2(3):227–231, 1976.

[5] D. Karger and M. Minkoff. Building steiner trees with incomplete global knowledge. *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[6] C. Liebchen, M. Lübbecke, R. H. Möhring, and S. Stiller. Recoverable robustness. *Tech. Report ARRIVAL-TR-0066, ARRIVAL-Project*, 2007.

[7] R Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Math. Program.*, 108:97–114, 2006.

[8] T. Schaefer. The complexity of satisfiability problems. *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, 1978.
[9] G. Yu and J. Yang. On the robust shortest path problem. *Computer and Operations Research*, 25(6):457–468, 1998.