Having a Plan B for Robust Optimization

André Chassein *

Fachbereich Mathematik, Technische Universität Kaiserslautern, Germany

Abstract

We extend the standard concept of robust optimization by the introduction of an alternative solution. In contrast to the classic concept, one is allowed to chose two solutions from which the best can be picked after the uncertain scenario has been revealed. We focus in this paper on the resulting robust problem for combinatorial problems with bounded uncertainty sets. We present a reformulation of the robust problem which decomposes it into polynomially many subproblems. In each subproblem one needs to find two solutions which are connected by a cost function which penalizes if the same element is part of both solutions. Using this reformulation, we show how the robust problem can be solved efficiently for the unconstrained combinatorial problem, the selection problem, and the minimum spanning tree problem. The robust problem corresponding to the shortest path problem turns out to be NP-complete on general graphs. However, for series-parallel graphs, the robust shortest path problem can be solved efficiently. Further, we show how approximation algorithms for the subproblem can be used to compute approximate solutions for the original problem.

Keywords: Robust Optimization, Combinatorial Optimization, Bounded Uncertainty, k-Adaptability, Approximation

1 Introduction and Motivation

Bertsimas and Sim present in their seminal paper [4] the concept minmax robustness with bounded uncertainty sets, in contrast to the previously published paper of Ben-Tal and Nemirovski [3] in which the use of ellipsoidal uncertainty sets for linear programming problems is presented. Bertsimas and Sim show that their concept can be applied to combinatorial optimization problems without changing the complexity of the problem. A variety of other robustness concepts has been developed for combinatorial optimization problems. The concepts differ either in the shape of the uncertainty set or in the notion of robustness. Other notions of robustness are minmax regret [1, 7, 15], recoverable robustness [8, 14, 18], adjustable robustness [2], or light robustness [11]. Recently the concept of k-adaptability has been proposed in [6]. The idea of this concept is to

*Email: chassein@mathematik.uni-kl.de

choose in advance a set of k solutions from which the best can be chosen after the uncertain scenario has been revealed. This idea is related to recoverable or adjustable robustness where one is allowed to modify a preselected solution, taking into account a recovery budget, after the uncertain scenario has been revealed. In [6], it is shown that the k-adaptability concept is equivalent to the classic minmax concept if the set of feasible solutions is convex. Further, it is shown that the concept applied to combinatorial problems boils down to the classic minmax problem of the their linear relaxation if k is large enough.

In this paper, we focus on the special case in which one is allowed to pick only two solutions in advance. Following the notation defined in [6], we call our problem 2-adaptability (2-adapt). Formally, the problem is defined as

$$\min_{x,y\in\mathcal{X}} \max_{\tilde{c}\in\mathcal{U}} \min\left(\tilde{c}^T x, \tilde{c}^T y\right) \tag{2-adapt}$$

where $\mathcal{X} \subset \{0,1\}^n$ is the feasible set of a combinatorial optimization problem and the uncertainty set $\mathcal{U}$ is defined as

$$\mathcal{U} = \left\{ \tilde{c} \in \mathbb{R}^n : \tilde{c}_i \in [c_i, c_i + d_i], \sum_{i=1}^{n} \frac{\tilde{c}_i - c_i}{d_i} \leq \Gamma \right\}$$

as proposed by Bertsimas and Sim [5]. The parameter $\Gamma \in \mathbb{N}$ in the definition of $\mathcal{U}$ specifies the size of the uncertainty set. It can be interpreted as the budget of uncertainty which can be spent on the components of the cost vector. If Γ is set to n, $\mathcal{U}$ corresponds to the full interval uncertainty set $[c, c+d]$. In the original definition of $\mathcal{U}$ in [4] also fractional values for Γ are allowed. We restrict us in this paper to integral values, but an extension to fractional values is conceivable.

The paper is organized as follows. We begin by presenting, in Section 2, a reformulation procedure which converts problem (2-adapt) to polynomially many subproblems. In Section 3, we analyze the computational complexity of problem (2-adapt) for the unconstrained combinatorial, the selection, the minimum spanning tree, and the shortest path problem. In Section 4, we show how an approximation algorithm for the subproblems can be used to approximate problem (2-adapt). We present in Section 5 an application of the (2-adapt) concept to the minimum spanning tree problem. We conclude the paper in Section 6 and post future research questions.

2 General Reformulation

In a first step, we derive a compact nonlinear reformulation of the problem. We begin by formulating the adversary problem $\max_{\tilde{c}\in\mathcal{U}} \min(\tilde{c}^T x, \tilde{c}^T y)$ as a linear program. We introduce a new variable z to replace $\min(\tilde{c}^T x, \tilde{c}^T y)$ with the two constraints $z \leq \tilde{c}^T x$ and $z \leq \tilde{c}^T y$. Further, we introduce variables $\delta_i \in [0,1]$ for $i \in [n]$ ($[n] := \{1, \ldots, n\}$) and set $\tilde{c}_i = c_i + d_i \delta_i$. After these transformations we arrive at the following linear programming formulation for the adversary problem.

$$
\begin{aligned}
\max_{\delta,z}\ & z \\
\text{s.t. } & z \le c^T x + \sum_{i=1}^{n} d_i x_i \delta_i \\
& z \le c^T y + \sum_{i=1}^{n} d_i y_i \delta_i \\
& \sum_{i=1}^{n} \delta_i \le \Gamma \\
& 0 \le \delta_i \le 1 \qquad\qquad \forall i \in [n]
\end{aligned}
$$

The corresponding dual of the problem is given by

$$
\begin{aligned}
\min_{\alpha,\gamma,\theta}\ & c^T x\alpha + c^T y(1-\alpha) + \sum_{i=1}^{n} \gamma_i + \Gamma\theta && \text{(Eval}(x,y)\text{)} \\
\text{s.t. } & d_i x_i \alpha + d_i y_i (1-\alpha) - \theta \le \gamma_i && \forall i \in [n] \\
& 0 \le \gamma_i && \forall i \in [n] \\
& 0 \le \alpha \le 1 \\
& 0 \le \theta
\end{aligned}
$$

The optimal value of the linear program Eval(x, y) is equal to the objective value of (2-adapt) of the solution pair (x, y). Using this reformulation of the adversary problem, we get the following nonlinear reformulation of (2-adapt). Note that we can add the constraint $\alpha \le 0.5$ without changing the optimal value due to the symmetry of the problem.

$$
\begin{aligned}
\min_{x,y,\alpha,\gamma,\theta}\ & c^T x\alpha + c^T y(1-\alpha) + \sum_{i=1}^{n} \gamma_i + \Gamma\theta && \text{(NL)} \\
\text{s.t. } & d_i x_i \alpha + d_i y_i (1-\alpha) - \theta \le \gamma_i && \forall i \in [n] \\
& 0 \le \gamma_i && \forall i \in [n] \\
& 0 \le \alpha \le 0.5 \\
& 0 \le \theta \\
& x \in \mathcal{X} \\
& y \in \mathcal{X}
\end{aligned}
$$

We claim that problem (NL) is equivalent to the following problem

$$
\begin{aligned}
\min_{x,y,\alpha}\ & c^T x\alpha + c^T y(1-\alpha) + \left\| \begin{pmatrix} d_1 x_1 \alpha + d_1 y_1 (1-\alpha) \\ \vdots \\ d_n x_n \alpha + d_n y_n (1-\alpha) \end{pmatrix} \right\|^{(\Gamma)} && \text{(NL}'\text{)} \\
\text{s.t. } & 0 \le \alpha \le 0.5 \\
& x \in \mathcal{X} \\
& y \in \mathcal{X}
\end{aligned}
$$

where $||v||^{(\Gamma)}$ is the sum of the Γ largest values of v. Lemma 2.1 helps in proving that (NL) and (NL$'$) are equal.

Lemma 2.1. *The optimal value of the optimization problem*

$$\min_{\theta \geq 0} \Gamma\theta + \sum_{i=1}^{n} \max(0, v_i - \theta)$$

is equal to $||v||^{(\Gamma)}$*, where* $v \in \mathbb{R}^n_+$.

Proof. We assume without loss of generality that the entries of v are sorted and pairwise different such that $v_1 < v_2 < \cdots < v_n$. Consider the function $f(\theta) = \Gamma\theta + \sum_{i=1}^{n} \max(0, v_i - \theta)$. Note that f is a piecewise linear function with $f(0) = \sum_{i=1}^{n} v_i$. Further, the breakpoints of f are given by $v_0, v_1, v_2, \ldots, v_n$ where $v_0 = 0$. The different slopes of f on the linear pieces are $\Gamma - n$, $\Gamma - (n-1)$, $\Gamma - (n-2), \ldots, \Gamma$. Hence, the function f is convex and its minimum is attained if the slope of the linear piece is 0. Note that the $(n - \Gamma)$th breakpoint of the function is part of the linear piece with slope 0. Hence, setting $\theta^* = v_{n-\Gamma}$ minimizes f. This proves the lemma, since $f(\theta^*) = \Gamma v_{n-\Gamma} + \sum_{i=1}^{n} \max(0, v_i - v_{n-\Gamma}) = \Gamma v_{n-\Gamma} + \sum_{i=v_{n-\Gamma}}^{n} v_i - v_{n-\Gamma} = ||v||^{(\Gamma)}$. □ □

Note that, in an optimal solution of (NL), we can set $\gamma_i = \max(0, d_i x_i \alpha + d_i y_i (1 - \alpha) - \theta)$. Further, if we fix x, y, and α, (NL) reduces to

$$c^T x\alpha + c^T y(1 - \alpha) + \min_{\theta \geq 0} \left(\Gamma\theta + \sum_{i=1}^{n} \max(0, d_i x_i \alpha + d_i y_i (1 - \alpha) - \theta) \right)$$

Using Lemma 2.1, it is immediate that (NL) is equivalent to (NL$'$). For fixed x and y, we define

$$h_{x,y}(\alpha) = c^T x\alpha + c^T y(1 - \alpha) + \left\| \begin{pmatrix} d_1 x_1 \alpha + d_1 y_1 (1 - \alpha) \\ \vdots \\ d_n x_n \alpha + d_n y_n (1 - \alpha) \end{pmatrix} \right\|^{(\Gamma)}$$

We define $H(\alpha) = \min_{x,y \in \mathcal{X}} h_{x,y}(\alpha)$. Then, problem (NL$'$) reduces to $\min_{\alpha \in [0, 0.5]} H(\alpha)$. Further, note that for all x and y, the function $h_{x,y}$ is piecewise linear in α. For all α for which the order of the values $d_i x_i \alpha + d_i y_i (1 - \alpha)$ is constant, $h_{x,y}$ is an affine function. Hence, every breakpoint $\hat{\alpha}$ of $h_{x,y}$ is found at the points at which the order of the values $d_i x_i \alpha + d_i y_i (1 - \alpha)$ changes. All these points can be found as solutions of the following equations:

- $\hat{\alpha} d_i = d_j$ for $i \neq j$
- $(1 - \hat{\alpha}) d_i = d_j$ for $i \neq j$
- $(1 - \hat{\alpha}) d_i = \hat{\alpha} d_j$ for $i \neq j$

Since H is the minimum of a set of piecewise linear functions with at most $O(n^2)$ many breakpoints $\mathcal{A} = \{\alpha_1, \ldots, \alpha_l\} \cup \{0, 0.5\}$, we can conclude that H is a piecewise concave function with breakpoints contained in $\mathcal{A}$ (recall that the

minimum of linear functions is concave). Further, the minimum of H is found at a point contained in $\mathcal{A}$. Therefore, we can simplify (NL′) to

$$\min_{\alpha\in\mathcal{A}} \min_{x,y\in\mathcal{X}} h_{x,y}(\alpha)$$

Hence, it suffices to focus on the problem $\min_{x,y\in\mathcal{X}} h_{x,y}(\alpha)$ for a fixed value of α. In the following, α denotes a fixed value and no longer a variable. We introduce again the variable θ to obtain the following formulation for the problem under consideration.

$$\begin{aligned}\min_{x,y,\theta}\ & c^T x\alpha + c^T y(1-\alpha) + \sum_{i=1}^{n} \max(0, d_i x_i \alpha + d_i y_i (1-\alpha) - \theta) + \Gamma\theta \\ \text{s.t. }\ & 0 \le \theta,\ x \in \mathcal{X},\ y \in \mathcal{X}\end{aligned}$$

Defining $g_{x,y}(\theta) = c^T x\alpha + c^T y(1-\alpha) + \sum_{i=1}^{n} \max(0, d_i x_i \alpha + d_i y_i (1-\alpha) - \theta)$ and $G(\theta) = \Gamma\theta + \min_{x,y\in\mathcal{X}} g_{x,y}(\theta)$ the problem reduces to

$$\min_{\theta\ge 0}\ G(\theta)$$

Note that for all x and y, the function $g_{x,y}$ is piecewise linear. Further, the set of all breakpoints of $g_{x,y}$ is contained in the set

$$\Theta(\alpha) = \{0\} \cup \bigcup_{i=1}^{n} \{d_i\alpha, d_i(1-\alpha), d_i\}.$$

Using the same reasoning as for the function H, we infer that function G is piecewise concave and all breakpoints are contained in $\Theta(\alpha)$. Note that the size of $\Theta(\alpha)$ is in $O(n)$. We conclude that we can simplify problem $\min_{x,y\in\mathcal{X}} h_{x,y}(\alpha)$ to

$$\min_{\theta\in\Theta(\alpha)} \Gamma\theta + \min_{x,y\in\mathcal{X}} g_{x,y}(\theta)$$

Hence, it suffices to focus on the problem $\min_{x,y\in\mathcal{X}} g_{x,y}(\theta)$ for a fixed value of θ. In the following, θ denotes a fixed value and no longer a variable. The problem $\min_{x,y\in\mathcal{X}} g_{x,y}(\theta)$ can be reformulated in the following compact form

$$\begin{aligned}\min_{x,y}\ & \sum_{i=1}^{n} f_i(x_i, y_i, \alpha, \theta) \\ \text{s.t. }\ & x \in \mathcal{X} \\ & y \in \mathcal{X}\end{aligned} \qquad (\mathrm{P}(\alpha,\theta))$$

The values of $f_i(x_i, y_i, \alpha, \theta)$, given in Table 1, are derived from the definition of $g_{x,y}(\theta)$ for fixed θ. We complete this section with the theorem that summarizes our findings.

Theorem 2.2. *Problem (2-adapt) is equivalent to*

$$\min_{\alpha\in\mathcal{A}} \min_{\theta\in\Theta(\alpha)} \Gamma\theta + \min_{\substack{x\in\mathcal{X}\\ y\in\mathcal{X}}} \sum_{i=1}^{n} f_i(x_i, y_i, \alpha, \theta).$$

$f_i(x_i, y_i, \alpha, \theta)$	$x_i = 0$ $y_i = 0$	$x_i = 1$ $y_i = 0$	$x_i = 0$ $y_i = 1$	$x_i = 1$ $y_i = 1$
$0 \le \theta \le d_i\alpha$	0	$(c_i + d_i)\alpha - \theta$	$(c_i + d_i)(1-\alpha) - \theta$	$c_i + d_i - \theta$
$d_i\alpha \le \theta \le d_i(1-\alpha)$	0	$c_i\alpha$	$(c_i + d_i)(1-\alpha) - \theta$	$c_i + d_i - \theta$
$d_i(1-\alpha) \le \theta \le d_i$	0	$c_i\alpha$	$c_i(1-\alpha)$	$c_i + d_i - \theta$
$d_i \le \theta$	0	$c_i\alpha$	$c_i(1-\alpha)$	c_i

Table 1: Values of the function $f_i(x_i, y_i, \alpha, \theta)$, used in the definition of (P(α, θ)).

The next corollary is an immediate consequence of Theorem 2.2.

Corollary 2.3. *If problem (P(α, θ)) can be solved in time T, problem (2-adapt) can be solved in $O(n^3T)$. In consequence, if (P(α, θ)) is polynomial solvable, (2-adapt) is also polynomial solvable.*

Proof. Enumerating over all values in $\mathcal{A}$ requires $O(n^2)$ time and enumerating all values in $\Theta(\alpha)$ requires $O(n)$. In summary, $O(n^3)$ many values need to be enumerated, i.e., $O(n^3)$ many problems (P(α, θ)) need to be solved. □ □

Remark 2.4. *On the first sight, one may expect that the factor of n^3 in the run time analysis prohibits to use the enumeration procedure to solve large scale problems. But taking a closer look at the sets $\mathcal{A}$ and $\Theta(\alpha)$ reveals that the size of these sets depends only on the number of different values of d_i. Hence, if the number of different uncertainty levels d_i is kept fix the procedure scales well even for large scale problems.*

We summarize the described reformulation procedure in Algorithm 1.

Algorithm 1 Algorithm to solve problem (2-adapt).

Compute $\mathcal{A}_1 = [0, 0.5] \cap \bigcup_{\substack{i\in[n]\\ j\in[n]}} \left\{ \frac{d_j}{d_i} \right\}$.
Compute $\mathcal{A}_2 = [0, 0.5] \cap \bigcup_{\substack{i\in[n]\\ j\in[n]}} \left\{ \frac{d_i - d_j}{d_i} \right\}$.
Compute $\mathcal{A}_3 = [0, 0.5] \cap \bigcup_{\substack{i\in[n]\\ j\in[n]}} \left\{ \frac{d_i}{d_j + d_i} \right\}$.
Set $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \{0, 0.5\}$.
for all $\alpha \in \mathcal{A}$ **do**
    Compute $\Theta(\alpha) = \bigcup_{i\in[n]} \{\alpha d_i, (1-\alpha)d_i, d_i\} \cup \{0\}$.
    for all $\theta \in \Theta(\alpha)$ **do**
        Solve $P(\alpha, \theta)$. Optimal solution: $(x_{(\alpha,\theta)}, y_{(\alpha,\theta)})$, optimal value: $f_{\alpha,\theta}$.
        Define $\mathrm{val}(\alpha, \theta) = \Gamma\theta + f_{\alpha,\theta}$.
    end for
end for
Set $(\alpha^*, \theta^*) = \mathrm{argmin}_{\alpha\in\mathcal{A}, \theta\in\Theta(\alpha)} \mathrm{val}(\alpha, \theta)$.
return $(x_{(\alpha^*,\theta^*)}, y_{(\alpha^*,\theta^*)})$

Remark 2.5. *Note that val(α, θ) (computed in Algorithm 1) is in general only an upper bound for the value of Eval$(x_{(\alpha,\theta)}, y_{(\alpha,\theta)})$. However, Theorem 2.2 ensures that val(α, θ) is equal to Eval$(x_{(\alpha,\theta)}, y_{(\alpha,\theta)})$ for the optimal choice of (α, θ).*

We set vectors $a, b, p \in \mathbb{R}^n$ in the following way:

$$\begin{aligned} a_i &= f_i(1,0,\alpha,\theta) && \forall i \in [n] \\ b_i &= f_i(0,1,\alpha,\theta) && \forall i \in [n] \\ p_i &= f_i(1,1,\alpha,\theta) - (a_i + b_i) && \forall i \in [n] \end{aligned}$$

Observe that, by definition, $a, b, p \geq 0$. Note that we can post problem $(\mathrm{P}(\alpha,\theta))$ in a more general form using the vectors a, b, and p:

$$\begin{aligned} \min_{x,y,z}\ & a^T x + b^T y + p^T z && && (P_{Coord}) \\ \text{s.t. }\ & x_i + y_i \leq z_i + 1 && \forall i \in [n] \\ & z_i \in \mathbb{B} && \forall i \in [n] \\ & x \in \mathcal{X} \\ & y \in \mathcal{X} \end{aligned}$$

A solution of the coordination problem (P_{Coord}) consists of two separate solutions of the original problem. Each solution has its own linear cost vector a and b. The solutions are connected by penalty costs $p_i \geq 0$ that need to be paid if both solutions contain the same element i. This is expressed by variable z_i which is set to 1 in an optimal solution if and only if $x_i = y_i = 1$. On the basis of Corollary 2.3, we investigate, in the next section, problem (2-adapt) for specific optimization problems. Instead of problem $(\mathrm{P}(\alpha,\theta))$, we consider directly the more general problem (P_{Coord}) which seems to be an interesting problem on its own.

3 Specific Problems

3.1 The Unconstrained Problem

The unconstrained combinatorial problem can be seen as the simplest combinatorial optimization problem. The feasible set of the unconstrained combinatorial problem consists of all binary vectors, i.e., $\mathcal{X} = \{0,1\}^n$. Due to its simple structure, it is a good starting point for the study of complex robustness settings (see [9]).

Lemma 3.1. *Problem (P_{Coord}) can be solved in $O(n)$ for the unconstrained combinatorial problem.*

Proof. Since $\mathcal{X} = \{0,1\}^n$ and the objective function is additively separable, the choice of (x_i, y_i) has no influence on (x_j, y_j). Hence, we find the global minimum by minimizing each component separately. Minimizing index i can be done by enumerating all four possible values of (x_i, y_i). □ □

Combining Corollary 2.3 and Lemma 3.1, we obtain the next theorem.

Theorem 3.2. *Problem (2-adapt) of the unconstrained combinatorial problem can be solved in $O(n^4)$.*

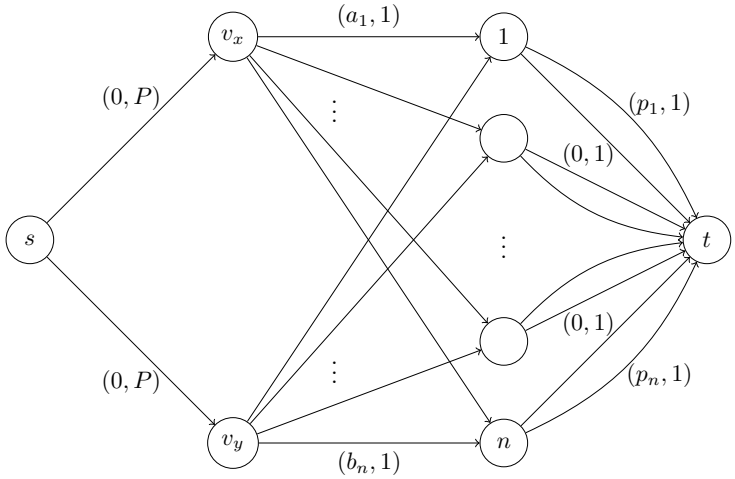


Figure 1: The instance of the minimum cost flow problem used in the proof of Lemma 3.3. The labels $(\overline{c}, \overline{u})$ denote the cost $\overline{c}$ and the capacity $\overline{u}$ of each edge. The costs and the capacity of edges (v_x, i) $((v_y, i))$ are given by $(a_i, 1)$ $((b_i, 1))$. Two edges connect every node i with the target node t, one with label $(0, 1)$, the other with $(p_i, 1)$. The excess of node s is $2P$ and of node t is $-2P$. The excess of all other nodes is 0.

3.2 The Selection Problem

A solution of the selection problem is a subset of P out of n items. The feasible set of the selection problem is given as $\mathcal{X} = \{x \in \{0,1\}^n : \sum_{i=1}^n x_i = P\}$, where $x_i = 1$ if and only if item i is selected. The problem has been frequently studied in the field of robust optimization (see [16]).

Lemma 3.3. *Problem (P_{Coord}) can be solved in $O(n^2 \log(n)^2)$ for the selection problem.*

Proof. We convert the problem to a minimum cost flow problem which can be solved efficiently. We describe the constructed instance in Figure 1.

Since all capacities of the minimum cost flow problem are integral, there exists an optimal solution which is integral and can be found in $O(n^2 \log(n)^2)$ (see [19]). Since $2P$ units of flow leave node s, P units of flow leave node v_x on exactly P edges $(v_x, i_1), \ldots, (v_x, i_P)$ causing costs of $a_{i_1} + \cdots + a_{i_P}$. These P edges correspond to the items $i_1, \ldots, i_P$ selected by solution x. The other P units of flow leave node v_y on exactly P edges $(v_y, j_1), \ldots, (v_y, j_P)$ causing costs of $b_{j_1} + \cdots + b_{j_P}$. Further, if only one unit of flow enters some node i, no additional costs occur for this unit since it can use the free edge from i to t. But if two units of flow enter node i, one of these two units uses the edge with costs p_i. This resembles the cost structure of problem (P_{Coord}). Hence, an optimal solution of the minimum cost flow problem leads to an optimal solution of (P_{Coord}). □ □

Combining Corollary 2.3 and Lemma 3.3, we obtain the next theorem.

Theorem 3.4. *Problem (2-adapt) of the selection problem can be solved in $O(n^5 \log(n)^2)$.*

3.3 The Minimum Spanning Tree Problem

Consider an undirected graph $G = (V, E)$ with vertex set V and edge set E. The minimum spanning tree problem (with cost vector c) can be formulated by the following linear program.

$$\begin{aligned} \min_x \; & c^T x && && \text{(LP-MST)} \\ & \sum_{e \in E} x_e = n - 1 && && \\ & \sum_{e \in E(S)} x_e \leq |S| - 1 && \forall S \subset V, S \neq \emptyset \\ & x_e \geq 0 && \forall e \in E \end{aligned}$$

where $E(S)$ denotes the set of all edges $e = (u, v)$ with $u, v \in S$. Further, we denote by $E(S, S')$ the set of all edges $e = (u, v)$ with $u \in S$ and $v \in S'$. It is not necessary to post integrality requirements for the problem since each vertex of the feasible polytope is integral. The constraints defined for each subset $S \subset V$ are called subtour elimination constraints, see [17] for more information about this formulation. We start by showing a structural lemma of the feasible set of (LP-MST). Denote by $\delta(e) \in \mathbb{B}^m$ the incidence vector of edge e. For a given solution x of (LP-MST), we say two different edges e_1 and e_2 are *compatible* with each other if there exists an $\epsilon > 0$ such that $x^+ := x + \epsilon\delta(e_1) - \epsilon\delta(e_2)$ and $x^- := x - \epsilon\delta(e_1) + \epsilon\delta(e_2)$ are both feasible for (LP-MST). We denote this definition of x^+ and x^- also as *shifting* solution x on edges e_1 and e_2 with shifting parameter ϵ. Denote by $C_x(e)$ the set of all compatible edges of edge e with respect to solution x. We say an edge e is fractional if $x_e \in (0, 1)$. Note that only fractional edges can be compatible to each other.

Lemma 3.5. *Let x be a solution of (LP-MST). Then, for each fractional edge e the set of compatible edges $C_x(e)$ is not empty.*

Proof. First, note that the constraint $\sum_{e \in E} x_e = n - 1$ holds for all choices of edges e_1 and e_2, since $\sum_{e \in E} x_e^+ = \sum_{e \in E} x_e^- = n - 1$. Denote by $\mathcal{F}$ the set of all tight subtour elimination constraints for solution x, i.e., $\sum_{e \in E(F)} x_e = |F| - 1$ for all $F \in \mathcal{F}$. We call $F \in \mathcal{F}$ a *tight set*. These are the critical constraints for the definition of compatible edges since all non-tight subtour elimination constraints stay fulfilled if we choose the shifting parameter ϵ small enough. Next, we show that the set $\mathcal{F}$ fulfills the following property which is essential for the rest of the proof. The claim and its proof can be found in [17] (we state it for the sake of completeness).

Claim 3.6. *If $F_1, F_2 \in \mathcal{F}$ and $F_1 \cap F_2 \neq \emptyset$, then $F_1 \cap F_2 \in \mathcal{F}$.*

Proof. Denote by $x(M) = \sum_{e \in E(M)} x_e$ the sum over all edges of $E(M)$ and by $x(M_1, M_2) = \sum_{e \in E(M_1, M_2)} x_e$ the sum over all edges of $E(M_1, M_2)$. For any two subsets $S_1, S_2 \subset V$, we have $x(S_1) + x(S_2) = x(S_1 \cup S_2) + x(S_1 \cap S_2) - x(S_1 \backslash S_2, S_2 \backslash S_1)$. Hence, $x(S_1) + x(S_2) \leq x(S_1 \cup S_2) + x(S_1 \cap S_2)$. For $F_1, F_2 \in \mathcal{F}$

and $F_1 \cap F_2 \neq \emptyset$, we have that

$$\begin{aligned}|F_1| - 1 + |F_2| - 1 = x(F_1) + x(F_2) &\leq x(F_1 \cup F_2) + x(F_1 \cap F_2) \\ &\leq |F_1 \cup F_2| - 1 + |F_1 \cap F_2| - 1 \\ &= |F_1| - 1 + |F_2| - 1.\end{aligned}$$

The second inequality holds since x is feasible and $F_1 \cap F_2 \neq \emptyset$. Hence, all inequalities must hold with equality, which shows that $F_1 \cap F_2 \in \mathcal{F}$. □ □

In the following, *inclusionwise minimal tight sets* will play a crucial role. A set is inclusionwise minimal tight if each strict subset of it is not a tight set. The idea is to contract all inclusionwise minimal tight sets. Contracting a set $V' \subset V$ means that we replace it with a single node v'. Further, every edge $e = (u, w)$ with $u \in V'$ and $w \in V \backslash V'$ is replaced with edge $e' = (v', w)$. All edges $e = (u, v)$ with $u, v \in V'$ are removed.

We repeat the following: Select an inclusionwise minimal tight set $F^* \in \mathcal{F}$ and contract it.

First, we show that the structure of the set of tight sets does not change if a minimal tight set F^* is contracted to a node v^*. We claim that we can split $\mathcal{F}$ into two disjoint parts $\mathcal{F}_1 = \{F \in \mathcal{F} : F^* \subset F\}$ and $\mathcal{F}_2 = \{F \in \mathcal{F} : F^* \cap F = \emptyset\}$. By definition, these two sets are disjoint. Assume that $\mathcal{F}_1 \cup \mathcal{F}_2 \neq \mathcal{F}$, i.e., there exists a set $\hat{F} \in \mathcal{F}$ with $F^* \cap \hat{F} \neq \emptyset$ and $F^* \not\subset F$. Then, $F^* \cap \hat{F}$ would also be a tight set, due to Claim 3.6, contradicting the fact that F^* is an inclusionwise minimal tight set. Hence, we conclude that $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$. Since $\mathcal{F}_2$ is not changed by the contraction of F^*, it is still a set of tight sets. The set $\mathcal{F}_1$ is changed by the contraction to the set $\mathcal{F}_1' = \{(F \backslash F^*) \cup \{v^*\} : F \in \mathcal{F}_1\}$. We claim that $\mathcal{F}_1'$ only consists of tight sets for the contracted graph. Let $F \in \mathcal{F}_1$ be a set which is contracted to $F' \in \mathcal{F}_1'$, i.e., $F' = (F \backslash F^*) \cup \{v^*\}$, then we have

$$\begin{aligned}\sum_{e \in E(F')} x_e &= \sum_{e \in E(F' \backslash \{v^*\})} x_e + \sum_{e \in E(F' \backslash \{v^*\}, v^*)} x_e \\ &= \sum_{e \in E(F' \backslash \{F^*\})} x_e + \sum_{e \in E(F' \backslash \{F^*\}, F^*)} x_e \\ &= \sum_{e \in E(F' \backslash \{F^*\})} x_e + \sum_{e \in E(F' \backslash \{F^*\}, F^*)} x_e + \sum_{e \in E(F^*)} x_e - (|F^*| - 1) \\ &= \sum_{e \in E(F)} x_e - |F^*| + 1 \\ &= |F| - 1 - |F^*| + 1 \\ &= |F'| - 1,\end{aligned}$$

where we use that F^* and F are tight sets. We conclude that the set of tight sets for the graph which is obtained after one contracting F^* is given by $\mathcal{F}' = \mathcal{F}_1' \cup \mathcal{F}_2$.

Note that the contraction process terminates with a graph that consists of a single node since the complete set $V \in \mathcal{F}$ is a tight set. Hence, each edge will be contracted in this process. Using the next claim, we can define the set of compatible edges $C_x(e)$ for an edge e.

Claim 3.7. *All fractional edges contracted in the same step are compatible.*

Proof. Let e_1 and e_2 be two different fractional edges which were contracted when we contracted the set F^*. We define $x^+ := x + \epsilon\delta(e_1) - \epsilon\delta(e_2)$ and $x^- := x - \epsilon\delta(e_1) + \epsilon\delta(e_2)$ for a sufficiently small $\epsilon > 0$. By construction, $x^+(F^*) = x^-(F^*) = x(F^*) = |F^*| - 1$. Hence, F^* is still feasible for x^+ and x^-. In conclusion, also each tight constraint $F \in \mathcal{F}_1$ is feasible. For all other tight constraints $F \in \mathcal{F}_2$, we have that $F^* \cap F = \emptyset$ and, therefore, $e_1, e_2 \notin E(F)$. In conclusion, $x^+(F) = x^-(F) = x(F) = |F| - 1$ for all $F \in \mathcal{F}_2$. Hence, x^+ and x^- are feasible for all tight constraints. If the shifting parameter ϵ is chosen small enough, all other subtour elimination constraints which are not tight, and the sign constraints $x \geq 0$ are also fulfilled for x^+ and x^-. This proves that x^+ and x^- are feasible for (LP-MST), which shows that e_1 and e_2 are compatible edges. □ □

We start the contraction process by contracting each edge $e = (u, v)$ with $x_e = 1$, since $\{u, v\}$ forms an inclusionwise minimal tight set. After these contractions only fractional edges remain. Note that a single fractional edge e cannot form a tight set. Hence, we conclude that in each additional contraction step at least two fractional edges must be contained.

To summarize, we have found a procedure which iteratively contracts sets of edges until all edges are contracted. In each contraction step either none or at least two fractional edges are contracted. Further, all edges that were contracted in a single step are compatible with each other. Combining these arguments proves that $C_x(e)$ is not empty for each fractional edge e. □ □

Remark 3.8. *It is easy to show that the iterative contraction process defines the maximal sets $C_x(e)$ for each edge e, i.e., if two edges are not contracted in the same step of the procedure they are also not compatible to each other. Note that Lemma 3.5 can be used to prove that each vertex of the feasible set of (LP-MST) is integral.*

It is important to note that the set of tight constraints $\mathcal{F}$ is the same for x, x^+, and x^-. Since the definition of compatible edges depends purely on $\mathcal{F}$, we conclude that, if two edges are compatible for x, they are also compatible for x^+ and x^-. This allows to shift multiple pairs simultaneously and still preserve feasibility. We summarize this in the following observation.

Observation 3.9. *Let x be a solution of (LP-MST) and $(e'_1, e''_1), \dots (e'_k, e''_k)$ a list of compatible pairs. Set $E' = \{e'_1, \dots, e'_k\}$ and $E'' = \{e''_1, \dots, e''_k\}$ and define $x^+ := x + \epsilon\delta(E') - \epsilon\delta(E'')$ and $x^- := x - \epsilon\delta(E') + \epsilon\delta(E'')$, where $\delta(E)$ denotes the incidence vector of set E. For sufficiently small ϵ, x^+ and x^- are both feasible for (LP-MST).*

The set of compatible edges also guarantees the following *transitivity* property.

Observation 3.10. *Let e', e'', and e''' be pairwise different edges. Let e' be compatible with e'' and let e'' be compatible with e''', then is e' also compatible with e'''.*

Next, we consider the integer programming formulation of (P_{Coord}) for the minimum spanning tree problem.

$$\min_{x,y,z} a^T x + b^T y + p^T z \qquad \text{(IP-2MST)}$$

$$\sum_{e \in E} x_e = n - 1$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad \forall S \subset V, S \neq \emptyset$$

$$\sum_{e \in E} y_e = n - 1$$

$$\sum_{e \in E(S)} y_e \leq |S| - 1 \qquad \forall S \subset V, S \neq \emptyset$$

$$x_e + y_e \leq z_e + 1 \qquad \forall e \in E$$

$$x_e \in \mathbb{B} \qquad \forall e \in E$$

$$y_e \in \mathbb{B} \qquad \forall e \in E$$

$$z_e \geq 0 \qquad \forall e \in E$$

Lemma 3.11. *The vertices of the linear relaxation of the feasible set of (IP-2MST) are integral.*

Proof. We denote the linear relaxation of (IP-2MST) as (LP-2MST). Assume that (x, y, z) is a fractional solution of (LP-2MST). The idea of the proof is to find two other feasible solutions (x^+, y^+, z^+) and (x^-, y^-, z^-) of (LP-2MST) by shifting some components of (x, y, z) by ϵ such that $(x, y, z) = 0.5((x^+, y^+, z^+) + (x^-, y^-, z^-))$. This shows that (x, y, z) cannot be a vertex of the feasible set.

The feasible set of (LP-2MST) breaks into two feasible sets of (LP-MST) if the constraints including the z variables are removed. Hence, we have to analyze how the z variables relate the x and y variables. To this end, we define the following edge sets for solution (x, y, z):

- $I_x^0 = \{e \in E : x_e \in (0, 1), z_e = 0, y_e = 0\}$
- $I_y^0 = \{e \in E : y_e \in (0, 1), z_e = 0, x_e = 0\}$
- $I_{x,z} = \{e \in E : x_e \in (0, 1), z_e \in (0, 1]\}$
- $I_{y,z} = \{e \in E : y_e \in (0, 1), z_e \in (0, 1]\}$
- $I_{\text{conflict}} = \{e \in E : x_e \in (0, 1), y_e \in (0, 1), z_e = 0\}$
- $I_{\text{agree}} = I_{x,z} \cup I_{y,z} \cup I_x^0 \cup I_y^0$

Note that each fractional edge (either x_e or y_e is fractional) is contained in at least one set. At the start of the analysis, we focus on the constraints $x_e + y_e \leq z_e + 1$. Later, we will deal with the subtour elimination constraints for x and y.

Consider an edge $e \in I_{x,z}$ (for $I_{y,z}$ it works analogously). If $z_e < 1$ and we increase or decrease x_e by a small enough amount, we can just increase or decrease z_e by the same amount and the corresponding constraint $x_e + y_e \leq z_e + 1$ is still fulfilled. If $z_e = 1$, the corresponding constraint is not tight, hence it is not violated if x_e is shifted by a small enough amount and y_e and z_e are kept fix.

Next, consider an edge $e \in I_x^0$ (for I_y^0 it works analogously). For this edge, the constraint $x_e + y_e \leq z_e + 1$ is not tight, hence it is not violated if x_e is shifted by a small enough amount and y_e and z_e are kept fix.

Combining these two cases, we obtain: If we change the value x_e (y_e) of a fractional edge from I_{agree}, there exists an adaption of the z_e value such that $x_e + y_e \leq z_e + 1$ is still fulfilled. Hence, x_e can be changed without influencing y_e and vice versa. This does not hold for I_{conflict}.

Consider an edge $e \in I_{\text{conflict}}$. If x_e is increased by a small amount, we might have to decrease y_e by a small amount to maintain constraint $x_e + y_e \leq z_e + 1$ (if the constraint is tight), and, vice versa, if y_e is increased, x_e needs to be decreased. Note that z_e cannot be shifted since it is already 0. In the following, we assume without loss of generality that all constraints corresponding to edges $e \in I_{\text{conflict}}$ are tight.

We summarize: We can change x_e for all edges in the set I_{agree} without influencing y_e and vice versa. Further, if we shift the value x_e for an element of I_{conflict}, we have to shift y_e in the opposite direction and vice versa.

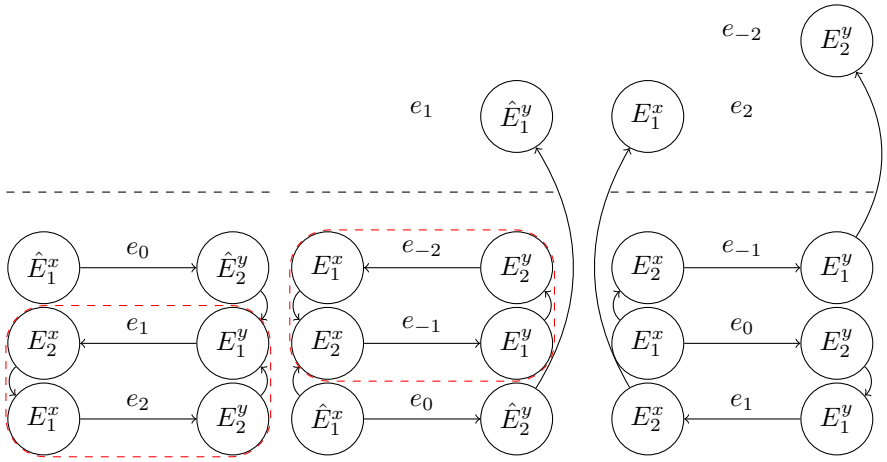
In the next step, we also take the subtour elimination constraints into account. Using Lemma 3.5, we define a sequence of shifting moves on x and y such that feasibility is maintained. Without loss of generality, we assume that at least one value x_e is fractional. We only define how to shift x and y. The z variables needs to be adapted accordingly. If we increase or decrease x_e for an edge $e \in I_{x,z}$, we also increase or decrease z_e, analogously for y_e and edges $e \in I_{y,z}$. Note that these two adaption effects may interfere if $e \in I_{x,z} \cap I_{y,z}$. In this case, either no change for z_e is required or a change that is twice as large as the shifting parameter. For all other edges, the changes of x_e or y_e have no influence on the value of z_e. For simplicity, in the following, we give no formal definition of the change in z since it is straightforward.

We begin with the easy case where $I_{\text{conflict}} = \emptyset$. In this case, we select a fractional edge e from I_{agree}. Due to Lemma 3.5, we find another edge $e' \in C_x(e)$ which is compatible to e with respect to the solution x (Observe that if (x, y, z) is feasible for (LP-2MST), then x and y are also feasible for (LP-MST)). We define $x^+ := x + \epsilon\delta(e) - \epsilon\delta(e')$ and $x^- := x - \epsilon\delta(e) + \epsilon\delta(e')$, we set $y^+ = y^- = y$. If z^+ and z^- are adapted accordingly, we get that (x^+, y^+, z^+) and (x^-, y^-, z^-) are both feasible for (LP-2MST) and $(x, y, z) = 0.5((x^+, y^+, z^+) + (x^-, y^-, z^-))$.

Next, consider the more complicated case where $I_{\text{conflict}} \neq \emptyset$. In this case, it may not suffice to only shift the x, y, and z values of two edges. Instead, we will define a procedure that defines four sets of edges E_1^x, E_2^x, E_1^y, and E_2^y which need to be changed simultaneously. To maintain feasibility, we have to ensure that E_1^x and E_2^x consist of pairs of compatible edges with respect to solution x and that E_1^y and E_2^y consist of pairs of compatible edges with respect to solution y. This ensures feasibility of the subtour elimination constraints. To guarantee that all constraints $x_e + y_e \leq z_e + 1$ are fulfilled, we have to make sure that for all $e \in I_{\text{conflict}}$, we have that $e \in E_1^x \Leftrightarrow e \in E_2^y$ and $e \in E_2^x \Leftrightarrow e \in E_1^y$. We will then define:

$$
\begin{aligned}
x^+ &:= x + \epsilon\delta(E_1^x) - \epsilon\delta(E_2^x) \\
x^- &:= x - \epsilon\delta(E_1^x) + \epsilon\delta(E_2^x) \\
y^+ &:= y + \epsilon\delta(E_1^y) - \epsilon\delta(E_2^y) \\
y^- &:= y - \epsilon\delta(E_1^y) + \epsilon\delta(E_2^y)
\end{aligned}
$$

Again, z^+ and z^- have to be chosen accordingly to x^+, x^- and y^+, y^-. Recall Observation 3.9, which is necessary to show that (x^+, y^+, z^+) and (x^-, y^-, z^-) are feasible for (LP-2MST). It is immediate that $(x, y, z) = 0.5((x^+, y^+, z^+) + (x^-, y^-, z^-))$.
In the following, we will define the procedure to find E_1^x, E_2^x, E_1^y, and E_2^y.



(a) Generating a cycle in the forward execution.

(b) Generating a cycle in the backward execution.

(c) Generating no cycle.

Figure 2: We show three sample executions of the procedure that is used to define the sets E_1^x, E_2^x, E_1^y, and E_2^y. Each row corresponds to one edge. The left column represents the sets E_1^x,E_2^x and the right column the sets E_1^y,E_2^y. Note that the sets $\hat{E}_1^x, \hat{E}_2^x, \hat{E}_1^y$, and $\hat{E}_2^y$ are not used for the definition of the shifted solutions. The straight arrows correspond to relations which arise due to the connection of x and y via the z variables. The curved arrows represent the selection of edges from $C_x(e)$ or $C_y(e)$. The set I_{agree} is above the dashed line and the set I_{conflict} below of it. The protocols of the forward and backward execution generating these examples is shown in Table 2.

The procedure consists of two parts, a *forward* and a *backward execution*. The procedure is guaranteed to terminate after at most m steps. However, the procedure might terminate already in the forward execution. In this case, the backward execution is not necessary.

The procedure defines four temporary sets $\hat{E}_1^x, \hat{E}_2^x, \hat{E}_1^y$, and $\hat{E}_2^y$ which will be used after the procedure to define the desired sets E_1^x, E_2^x, E_1^y, and E_2^y.

At the beginning of the process, we choose an arbitrary fractional edge $e_0 \in I_{\text{conflict}}$ and add this edge to $\hat{E}_1^x$. Edge e_0 defines the starting point of the forward and the backward execution.

Each loop of the forward execution consists of two steps, a y- and a x-step. Each step begins with some starting edge e.

We begin the y-step with starting edge $e_0 \in \hat{E}_1^x$. Since $e_0 \in I_{\text{conflict}}$, we add e_0 to $\hat{E}_2^y$. In the next step, we choose an edge $e_1 \in C_y(e_0)$ which is compatible to e_0 with respect to solution y. We have to distinguish four cases:

Case y1: $e_1 \in I_{\text{agree}}$. We add e_1 to $\hat{E}_1^y$. We terminate the forward execution

and continue with the backward execution.

Case y2: $e_1 \notin \hat{E}_1^y \cup \hat{E}_2^y$. We add e_1 to $\hat{E}_1^y$. We continue with the x-step with e_1 as the starting edge.

Case y3: $e_1 \in \hat{E}_1^y$. In this case, we terminate the complete process since we have found a cycle. This cycle will be used to define the final sets E_1^x, E_2^x, E_1^y, and E_2^y.

Case y4: $e_1 \in \hat{E}_2^y$. In this case, we know from the transitivity property (see Observation 3.10) that there exists another edge $e_1' \in \hat{E}_1^y$ which is also compatible to e_0. We select e_1' instead of e_1 and continue as in Case y3.

Note that only Cases y1 and y2 can happen in the first iteration of the forward execution. Next, we define the x-step with starting edge e_1.

Since $e_1 \in I_{\text{conflict}}$, we add e_1 to $\hat{E}_2^x$. In the next step, we choose an edge $e_2 \in C_x(e_1)$ which is compatible to e_1 with respect to solution x. We have to distinguish four cases:

Case x1 $e_2 \in I_{\text{agree}}$. We add e_2 to $\hat{E}_1^x$. We terminate the forward execution and continue with the backward execution.

Case x2: $e_2 \notin \hat{E}_1^x \cup \hat{E}_2^x$. We add e_2 to $\hat{E}_1^x$. We continue with the y-step with e_2 as the starting edge.

Case x3: $e_2 \in \hat{E}_1^x$. In this case, we terminate the complete process since we have found a cycle. This cycle will be used to define the final sets E_1^x, E_2^x, E_1^y, and E_2^y.

Case x4: $e_2 \in \hat{E}_2^x$. In this case, we know from the transitivity property (see Observation 3.10) that there exists another edge $e_2' \in \hat{E}_1^x$ which is also compatible to e_1. We select e_2' instead of e_2 and continue as in Case x3.

Note that only Cases x1,x2, and x3 can happen in the first iteration of the forward execution (Case x3 might happen since already one y-step was executed). After at most m iterations of y- and x-steps, we have either found a cycle, in this case, we can skip the backward execution, or no cycle has been found. In the latter case, we start the backward execution with starting edge e_0.

The main difference of the backward execution to the forward execution is that it starts with an x-step, compared to the forward execution which started with an y-step. The description of the x- and the y-step of the backward execution is almost identical to the forward execution. Just interchange $\hat{E}_1^x$ with $\hat{E}_2^x$ and interchange $\hat{E}_1^y$ with $\hat{E}_2^y$. Further, the Cases y1 and x1 needs to be adapted.

In Cases y1 and x1, we terminate the backward execution and set E_1^x, E_2^x, E_1^y, and E_2^y equal to the temporary sets $\hat{E}_1^x, \hat{E}_2^x, \hat{E}_1^y$, and $\hat{E}_2^y$.

Next, we consider the case in which neither Case x1 nor Case y1 has happened in any of the x- and y-steps of the backward execution. In this case, we must have found a cycle (either in the forward or in the backward execution). We obtain E_1^x, E_2^x, E_1^y, and E_2^y from the temporary sets $\hat{E}_1^x, \hat{E}_2^x, \hat{E}_1^y$, and $\hat{E}_2^y$ by

		Step	Starting Edge	Selected Edge	Case
a)	Forward	y	e_0	e_1	y2
		x	e_1	e_2	x2
		y	e_2	e_1	y3
b)	Forward	y	e_0	e_1	y1
	Backward	x	e_0	e_{-1}	x2
		y	e_{-1}	e_{-2}	y2
		x	e_{-2}	e_{-1}	x3
c)	Forward	y	e_0	e_1	y2
		x	e_1	e_2	x1
	Backward	x	e_0	e_{-1}	x2
		y	e_{-1}	e_{-2}	y1

Table 2: Protocols of the forward and backward executions shown in Figure 2.

removing all edges which are not contained in this cycle.

Note that in both cases, independent of the fact if we have found a cycle or not, we have that E_1^x and E_2^x as well as E_1^y and E_2^y consist of pairs of compatible edges. Further, the forward and backward execution of the process ensure that we have for all edges $e \in I_{\text{conflict}}$ that $e \in E_1^x \Leftrightarrow e \in E_2^y$ and $e \in E_2^x \Leftrightarrow e \in E_1^y$. Hence, we have found the desired sets E_1^x, E_2^x, E_1^y, and E_2^y which completes the proof.

We provide in Figure 2 three examples of the forward and backward executions for a better understanding of the procedure. We list in Table 2 the protocols of the forward and backward executions for each of the three examples. □ □

As shown by Lemma 3.11, we can relax the integrality constraints of (IP-2MST) since the linear relaxation is integral. Note that the exponentially many subtour elimination constraints can be separated efficiently [17]. Therefore, we can use the ellipsoid method to efficiently solve the linear relaxation of (IP-2MST) (see [13]). In conclusion, we obtain the main lemma of the section.

Lemma 3.12. *Problem (P_{Coord}) can be solved in polynomial time for the minimum spanning tree problem.*

Combining Corollary 2.3 and Lemma 3.12, we obtain the next theorem.

Theorem 3.13. *Problem (2-adapt) of the minimum spanning tree problem can be solved in polynomial time.*

3.4 The Shortest Path Problem

Unfortunately, it turns out that the problem (2-adapt) for the shortest path problem is strongly NP-complete for general graphs. Hence, problem (P_{Coord}) cannot be solved efficiently on general graphs unless P=NP.

Theorem 3.14. *Problem (2-adapt) is strongly NP-complete for the shortest path problem, even if* $\Gamma = 1$.

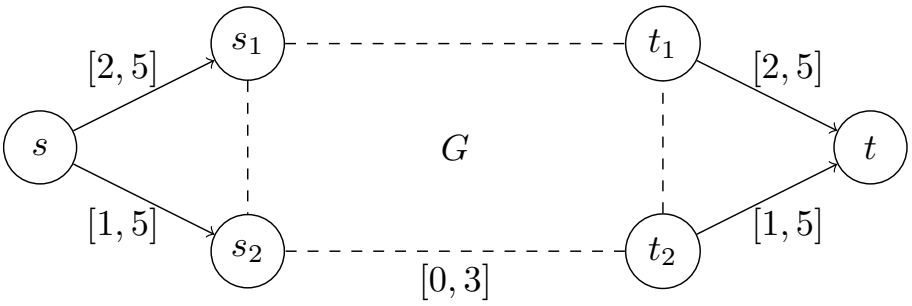


Figure 3: We construct graph G' by adding nodes s and t to G and by connecting them appropriately to s_1, s_2, t_1 and t_2. The intervals on the edges define the corresponding cost intervals. The cost interval of all edges from G are set to $[0, 3]$.

Proof. We give a reduction from the 2-disjoint path problem, which is known to be strongly NP-complete (see [12]). A instance of the 2-disjoint path problem (2DPP) consists of a graph $G = (V, E)$ with two source nodes s_1 and s_2 and two sink nodes t_1 and t_2. The problem is to decide if there exists an s_1-t_1 path P and an $s_2 - t_2$ path Q such that P and Q are disjoint. Given graph G, we construct a graph $G' = (V \cup \{s, t\}, E \cup E_{new})$, where $E_{new} = \{(s, s_1), (s, s_2), (t_1, t), (t_2, t)\}$ as shown in Figure 3. The uncertainty budget Γ is set to 1. The cost of the different edges are defined by the following intervals:

- $[2, 5]$ for the edges (s, s_1) and (t_1, t),
- $[1, 5]$ for the edges (s, s_2) and (t_2, t),
- and $[0, 3]$ for all other edges.

Denote by OPT the optimal value of the created 2-adaptability instance. We claim that $OPT \leq \frac{34}{7}$ if and only if the 2DPP is a yes-instance, i.e., there exists an $s_1 - t_1$ path P and an $s_2 - t_2$ path Q such that P and Q are disjoint.

First, we assume that such paths P and Q exist in G. We define the $s - t$ paths $P' = (s, s_1) \circ P \circ (t_1, t)$ and $Q' = (s, s_2) \circ Q \circ (t_2, t)$ in G'. Next, we consider the problem $\max_{c \in \mathcal{U}} \min(c(P'), c(Q'))$, where $c(P')$ denotes the cost of path P' under scenario c. We call this the adversary problem. The optimal solution of the adversary problem defines the value of solution (P', Q'). Since P' and Q' are edge disjoint, we have to split the uncertainty budget into two parts and distribute it among P' and Q'. Note that all cost intervals of the edges of P' have length 3, whereas Q' contains the edge (s, s_2) with a interval length of 4. Further, the nominal cost ($\Gamma = 0$) of P' is 4 and of Q' is 2. Hence, the adversary problem can be rewritten as:

$$\max_{\alpha \in [0,1]} \min\left(4 + 3\alpha, 2 + 4(1 - \alpha)\right) = \frac{34}{7}$$

This shows the first direction of the claim. For the second direction, assume that we have an optimal solution (P', Q') of the 2-adaptability problem such that $OPT \leq \frac{34}{7}$. To show that the restriction of (P', Q') onto G, gives a solution of the 2DPP, we have to distinguish two cases:

Case 1: P' and Q' share at least one edge.
Denote by e' the edge contained in P' and Q'. If $e' \in E_{new}$, we can spent the complete uncertainty budget on this edge to increase its cost to 5. It is an immediate consequence that the value of the solution (P', Q') is at least 5, which is a contradiction. On the other hand, if $e' \in E$, we can also spent the complete uncertainty budget on this edge to increase its cost to 3. Since the nominal cost of each $s-t$ path are at least 2, we conclude that the cost of (P', Q') is at least 5.

Case 2: P' and Q' are edge disjoint.
Without loss of generality, we assume that P' contains (s, s_1). Further, we assume that P' contains (t_2, t). In consequence, Q' contains (s, s_2) and (t_1, t). Note that in this case the restriction of (P', Q') onto G gives no feasible solution of the 2-disjoint path problem since the two disjoint paths are not correctly connected to s_1, t_1 and s_2, t_2. The nominal cost of both paths is 3 and both paths contain an edge with cost interval $[1, 5]$ (an interval with length 4). We obtain again a contradiction since the adversary problem can be rewritten as:

$$\max_{\alpha \in [0,1]} \min\left(3 + 4\alpha, 3 + 4(1-\alpha)\right) = 5$$

In summary, we conclude that P' and Q' must be edge disjoint and that P' contains (s, s_1) and (t_1, t) and that Q' contains (s, s_2) and (t_2, t). This guarantees that the restriction of (P', Q') onto G gives a feasible solution of the 2DDP, which concludes the proof. □ □

Remark 3.15. *A very similar reduction can be used to show that problem (P_{Coord}) is NP-complete for the shortest path problem.*

Corollary 3.16. *If $\mathsf{P} \neq \mathsf{NP}$, there is no β-approximation algorithm for problem (2-adapt) of the shortest path problem with $\beta < \frac{35}{34}$, even if $\Gamma = 1$.*

Proof. Consider again the reduction used in the proof of Theorem 3.14. If the 2DPP instance is a yes-instance, i.e., there exist two disjoint $s_1 - t_1$ and $s_2 - t_2$ paths, then the optimal objective value of the (2-adapt) problem is $\frac{34}{7}$. Contrary, if the 2DPP instance is a no-instance, we have seen that each solution of the (2-adapt) problem has objective value of at least 5. We conclude that no approximation algorithm with approximation guarantee $< \frac{35}{34}$ exists for problem (2-adapt) unless $\mathsf{P} = \mathsf{NP}$. □ □

Note that Corollary 3.16 implies that no PTAS can exist for this problem if $\mathsf{P} \neq \mathsf{NP}$. Because of these negative results on general graphs, we consider a restricted graph classes which leads to a polynomial solvable problem. Due to convention, we use m instead of n to denote the number of edges of the underlying graph.

Lemma 3.17. *Problem (P_{Coord}) can be solved in $O(m)$ for the shortest path problem on series-parallel graphs.*

Proof. At first, we construct a binary decomposition tree of the series-parallel graph G, which can be done in linear time (see [10]). The vertices of the decomposition tree $\mathcal{V}$ represent a subgraph $G_{\mathcal{V}}$ of G. The leafs of the decomposition tree represent single edges and the graph corresponding to node $\mathcal{V}$ with children nodes $\mathcal{V}_1$ and $\mathcal{V}_2$ is defined by either series or parallel composing $G_{\mathcal{V}_1}$ and $G_{\mathcal{V}_2}$.

The algorithm computes, for each node $\mathcal{V}$ of the decomposition tree, a triple $T_{\mathcal{V}} = ((P_X, P_Y), P_a^*, P_b^*)$. The pair (P_X, P_Y) is the optimal solution of problem (P_{Coord}) on $G_{\mathcal{V}}$ and P_a^* and P_b^* are the shortest paths with respect to cost a and b on $G_{\mathcal{V}}$. The definition of the correct triples for the leaves of the decomposition tree are trivial, since the leaves represent single edges.

Consider a node $\mathcal{V}$ of the decomposition tree with children nodes $\mathcal{V}_1$ and $\mathcal{V}_2$. We can assume that the algorithm has already computed the triples $T_{\mathcal{V}_1} = ((P_X, P_Y), P_a^*, P_b^*)$ and $T_{\mathcal{V}_2} = ((Q_X, Q_Y), Q_a^*, Q_b^*)$ for $\mathcal{V}_1$ and $\mathcal{V}_2$.
If $\mathcal{V}$ is generated by a series composition, we set

$$T_{\mathcal{V}} = \begin{pmatrix} (P_X \circ Q_X, P_Y \circ Q_Y) \\ P_a^* \circ Q_a^* \\ P_b^* \circ Q_b^* \end{pmatrix}.$$

If $\mathcal{V}$ is generated by a parallel composition, we set

$$T_{\mathcal{V}} = \begin{pmatrix} (P_X^*, Q_X^*) \\ \operatorname{argmin}\{a(P_a^*), a(Q_a^*)\} \\ \operatorname{argmin}\{b(P_b^*), b(Q_b^*)\} \end{pmatrix}.$$

To obtain (P_X^*, Q_X^*), we compute for each of the four pairs (P_X, P_Y), (Q_X, Q_Y), (P_a^*, Q_b^*), and (Q_a^*, P_b^*) the objective value with respect to the objective function of (P_{Coord}) and choose the best one. Note that the triple that corresponds to the root of the decomposition tree contains the optimal solution of problem (P_{Coord}) on graph G. Since each triple can be computed in constant time, the complete algorithm can be implemented in linear time. □ □

Combining Corollary 2.3 and Lemma 3.17, we obtain the next theorem.

Theorem 3.18. *Problem (2-adapt) of the shortest path problem on series-parallel graphs can be solved in $O(m^4)$.*

4 Approximation

In this section, we show how approximation algorithms for (P_{coord}) can be used to approximate (2-adapt). Assume we have given an β-approximation algorithm for problem (P_{coord}) with $\beta \geq 1$. Recall that each problem $P(\alpha, \theta)$ for $\alpha \in \mathcal{A}$ and $\theta \in \Theta(\alpha)$ can be formulated as a coordination problem P_{coord}. Hence, we can use the β-approximation algorithm to obtain approximate solutions of $P(\alpha, \theta)$. Denote by $(x_{(\alpha,\theta)}, y_{(\alpha,\theta)})$ the approximate solution of $P(\alpha, \theta)$.

Theorem 4.1. *Algorithm 2 is a β-approximation algorithm for problem (2-adapt).*

Proof. Denote by OPT the optimal value of problem (2-adapt) and by ALG the objective value of the approximate solution returned by Algorithm 2. From Theorem 2.2, we know that there exists $\alpha' \in \mathcal{A}$ and $\theta' \in \Theta(\alpha')$ and a solution (x^*, y^*) such that

$$\text{OPT} = \Gamma\theta' + \sum_{i=1}^{n} f_i(x_i^*, y_i^*, \alpha', \theta').$$

Algorithm 2 Approximation algorithm for problem (2-adapt).

for all $\alpha \in \mathcal{A}$ **do**
  for all $\theta \in \Theta(\alpha)$ **do**
    Compute a β-approximate solution $(x_{(\alpha,\theta)}, y_{(\alpha,\theta)})$ of $P(\alpha, \theta)$.
    Compute $\text{val}(\alpha, \theta) = \text{Eval}(x_{(\alpha,\theta)}, y_{(\alpha,\theta)})$.
  end for
end for
Set $(\alpha^*, \theta^*) = \text{argmin}_{\alpha \in \mathcal{A}, \theta \in \Theta(\alpha)} \text{val}(\alpha, \theta)$.
return $(x_{(\alpha^*,\theta^*)}, y_{(\alpha^*,\theta^*)})$

Denote by (x', y') the β-approximate solution computed for $P(\alpha', \theta')$ by Algorithm 2 and by $(\tilde{x}, \tilde{y})$ the solution returned by Algorithm 2. From the definition of the algorithm it follows that

$$Eval(\tilde{x}, \tilde{y}) \leq Eval(x', y')$$

where $Eval(x, y)$ denotes the optimal value of $\text{Eval}(x, y)$. Further, since (x', y') is a β-approximate solution and (x^*, y^*) is an optimal solution of $P(\alpha', \theta')$ it holds that

$$\sum_{i=1}^{n} f_i(x'_i, y'_i, \alpha', \theta') \leq \beta \sum_{i=1}^{n} f_i(x^*_i, y^*_i, \alpha', \theta')$$

Lastly, we have to observe that

$$Eval(x', y') =$$

$$\min_{\substack{0 \leq \alpha \leq 1 \\ 0 \leq \theta}} c^T x' \alpha + c^T y'(1-\alpha) + \Gamma\theta + \sum_{i=1}^{n} \max(0, d_i x'_i \alpha + d_i y'_i (1-\alpha) - \theta) \leq$$

$$c^T x' \alpha^* + c^T y'(1-\alpha^*) + \Gamma\theta^* + \sum_{i=1}^{n} \max(0, d_i x'_i \alpha^* + d_i y'_i (1-\alpha^*) - \theta^*) =$$

$$\Gamma\theta^* + \sum_{i=1}^{n} f_i(x'_i, y'_i, \alpha^*, \theta^*)$$

Using that $\beta \geq 1$ and combining all previous inequalities, we can prove the claimed approximation guarantee:

$$\begin{aligned}
\text{ALG} &= Eval(\tilde{x}, \tilde{y}) \\
&\leq Eval(x', y') \\
&\leq \Gamma\theta^* + \sum_{i=1}^{n} f_i(x'_i, y'_i, \alpha^*, \theta^*) \\
&\leq \Gamma\theta^* + \beta \sum_{i=1}^{n} f_i(x^*_i, y^*_i, \alpha^*, \theta^*) \\
&\leq \beta \left(\Gamma\theta^* + \sum_{i=1}^{n} f_i(x^*_i, y^*_i, \alpha^*, \theta^*) \right) \\
&= \beta \text{OPT}
\end{aligned}$$

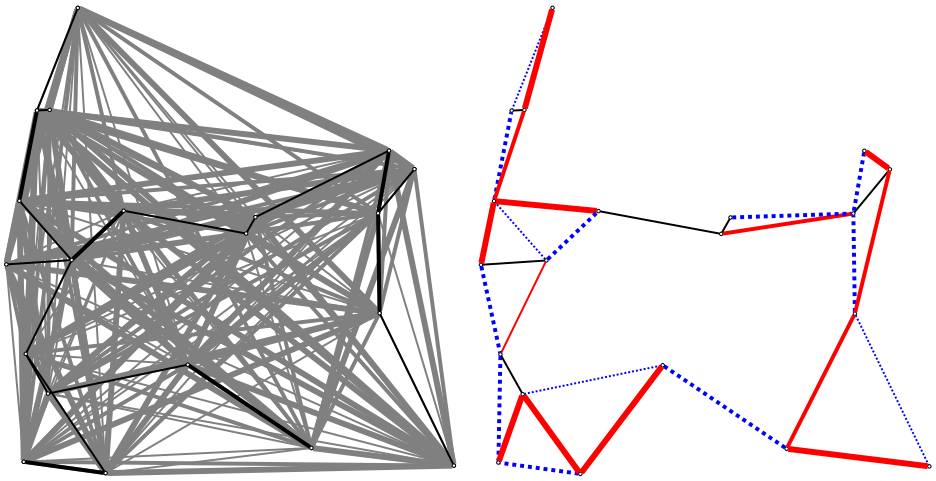
□ □

Corollary 4.2 is an immediate consequence of Theorem 4.1.

Corollary 4.2. *An FPTAS (PTAS) for problem (P_{coord}) can be transferred to an FPTAS (PTAS) for problem (2-adapt).*

5 Case Study

In this case study, we demonstrate the application of the concept to the minimum spanning tree problem. As test graph, we use a complete graph with 20 nodes. To generate the vertex set, we choose uniformly at random 20 points in the square $[0, 100]^2$. The cost c_e of each edge $e = (u, v)$ is set to be the Euclidean distance between u and v. The uncertain cost d_e are chosen uniform at random from the set $\{10, 20, 30\}$. The parameter Γ which specifies the size of the uncertainty set is set to 10. The graph is shown in Figure 4a in gray. The thickness of an edge indicates the amount of uncertainty which is assigned to this edge. The thin, average, thick edges are the edges with $d_e = 10, 20, 30$. First, we solved the classic minmax minimum spanning tree problem for the generated instance. The classic solution is shown in Figure 4a in black. In Figure 4b, we present the optimal pair of spanning trees for problem (2-adapt). To compare



(a) The complete graph is shown in gray. The classic robust solution is shown in black.

(b) The first solution of the solution pair is shown in red (straight), the second in blue (dashed). All edges shared by both solutions are shown in black.

Figure 4: Sample instance of the minimum spanning tree problem.

the performance of the classic solution and the solution pair of (2-adapt), we did the following simulation 100000 times. We sampled a cost scenario from $[c, c + d]$ by setting the cost of each edge e with 95% probability to c_e and with 5% probability to $c_e + d_e$. After we have sampled the cost scenario, we computed the cost of the classic solution and the cost of the cheapest solution of the solution pair. The corresponding distribution is shown in Figure 5. The comparison reveals that the worst case performance guarantee can be improved considerably if the problem setup allows for a plan B.

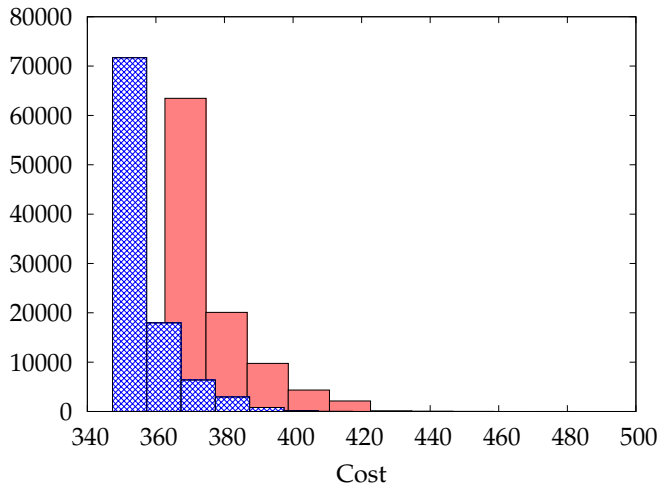


Figure 5: Simulated cost distribution. The distribution of the classic robust solution is shown in red. Choosing always the cheapest of the solution pair of (2-adapt) yields the blue (patterned) distribution.

6 Conclusions and Future Research

In this paper, we presented an extension of the classic minmax robustness concept from [4]. The concept is applicable for situations in which decisions can be selected after the uncertain scenario has been revealed. The idea is to find a pair of solutions from which the best can be chosen after the realization of the uncertainty set. We show that the concept can be solved in polynomial time for the unconstrained, the selection, the minimum spanning tree, and the shortest path problem on series-parallel graphs. For the shortest path problem on general graphs, we prove that the resulting problem is NP-complete. We present a reformulation of the problem which decomposes it into polynomially many subproblems. We show that an approximate algorithm for the subproblem can be extended to an approximation algorithm for the original problem.

The goal of future research is to analyze the application of this concept to other combinatorial problems. Also NP-complete problems, e.g., the knapsack problem, should be considered. For these problems, the goal is to find good approximation algorithms for the coordination problems (P_{Coord}), since the approximation guarantee transfers to the original problem.

We have already shown that the corresponding variant of the minimum spanning tree problem is polynomially solvable. However, the polynomial algorithm relies on the ellipsoid method. Hence, it is an interesting task to find a combinatorial algorithm for this problem which runs in strongly polynomial time.

References

[1] H. Aissi, C. Bazgan, and D. Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

[2] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[3] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[4] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1):49–71, 2003.

[5] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[6] C. Buchheim and J. Kurtz. Min–max–min robust combinatorial optimization. *Mathematical Programming*, pages 1–23, 2016. to appear.

[7] A. Chassein and M. Goerigk. A new bound for the midpoint solution in minmax regret optimization with an application to the robust shortest path problem. *European Journal of Operational Research*, 244(3):739–747, 2015.

[8] A. Chassein and M. Goerigk. On the recoverable robust traveling salesman problem. *Optimization Letters*, 10(7):1479–1492, 2016.

[9] A. Chassein and M. Goerigk. Minmax regret combinatorial optimization problems with ellipsoidal uncertainty sets. *European Journal of Operational Research*, 258(1):58–69, 2017.

[10] D. Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41–55, 1992.

[11] M. Fischetti and M. Monaci. Light robustness. In *Robust and Online Large-Scale Optimization*, chapter 2. Springer, 2009.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[13] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[14] A. Kasperski, A. Kurpisz, and P. Zieliński. *Recoverable Robust Combinatorial Optimization Problems*, pages 147–153. Springer International Publishing, 2014.

[15] A. Kasperski and P. Zieliński. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5):177 – 180, 2006.

[16] A. Kasperski and P. Zieliński. Robust discrete optimization under discrete and interval uncertainty: A survey. In *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, pages 113–143. Springer, 2016.

[17] L. C. Lau, R. Ravi, and M. Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.

[18] C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In *Robust and Online Large-Scale Optimization*, chapter 1. Springer, 2009.

[19] B. J. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations research*, 41(2):338–350, 1993.