Having a Plan B for Robust Optimization

André Chassein[1]

[1]Fachbereich Mathematik, Technische Universität Kaiserslautern, Germany

Abstract We continue in this paper the study of k-adaptable robust solutions for combinatorial optimization problems with bounded uncertainty sets. In this concept not a single solution needs to be chosen to hedge against the uncertainty. Instead one is allowed to choose a set of k different solutions from which one can be chosen after the uncertain scenario has been revealed. We first show how the problem can be decomposed into polynomially many subproblems if k is fixed. In the remaining part of the paper we consider the special case where $k = 2$, i.e., one is allowed to choose two different solutions to hedge against the uncertainty. We decompose this problem into so called coordination problems. The study of these coordination problems turns out to be interesting on its own. We prove positive results for the unconstrained combinatorial optimization problem, the matroid maximization problem, the selection problem, and the shortest path problem on series parallel graphs. The shortest path problem on general graphs turns out to be NP-complete. Further, we present for minimization problems how to transform approximation algorithms for the coordination problem to approximation algorithms for the original problem. We study the knapsack problem to show that this relation does not hold for maximization problems in general. We present a PTAS for the corresponding coordination problem and prove that the 2-adaptable knapsack problem is not at all approximable.

Keywords Robust Optimization; k-Adaptability; Combinatorial Optimization; Bounded Uncertainty; 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, 9, 18], recoverable robustness [10, 17, 20], adjustable robustness [2], or light robustness [13].

In [6] the concept of k-adaptability is introduced. In this concept a solution consists of two components. A first stage component which needs to be decided before a uncertain scenario realizes and a second stage component which can be implemented after the uncertain parameters are known. To reduce the complexity of the problem the authors restrict the second stage solution to stem from a finite set of possible solution candidates. Hence, an optimal solution for the k-adaptability problem consists of a single first stage solution and a list of k second stage solutions, from which the best may be chosen after the uncertain scenario has been realized. However, the resulting problem is still very challenging and has attained considerable attention in the community (see for example [15, 16]). A further simplification is to remove the first stage decision and only focus on the set of k second stage solutions, as done in [8]. We call this variant in the following (k-adapt). For problem (k-adapt) it is shown that it 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 their linear relaxation if k is large enough.

We continue in this paper the study of problem (k-adapt) by showing that it can be reformulated to polynomially many subproblems if k is fixed. Next, we focus on the restricted version of the problem where $k = 2$, i.e. problem (2-adapt). This study may give a first hint about the theoretical and practical complexity of the general problem (k-adapt). Especially, since different positive results are proven which allow for efficient algorithms to solve the problems. Besides the study of adaptability problems, the subproblems in which problem (2-adapt) can be decomposed turns out to have a compelling structure which may find also applications in other areas like game theory. Hence, the study of these problems is interesting on its own.

1.1 Structure of this Paper

In the first part of the paper, we focus on the general problem (k-adapt) and show in Section 2 a general reformulation which converts this problem to polynomially many subproblems if k is fixed. In the remaining part of the paper, we focus on the special case where $k = 2$, i.e. on problem (2-adapt). It turns out that the resulting subproblems for this case obey an interesting combinatorial structure. We call these problem coordination problems (P_{Coord}) and introduce them in Section 3. We analyze in Section 4 the coordination problems for the unconstrained combinatorial, the matroid maximization, the selection, and the shortest path problem. We prove that all of these problems except the shortest path problem can be solved efficiently, which allows for an efficient solution of the corresponding (2-adapt) problem versions. For the shortest path problem, we show that the (2-adapt) problem is strongly NP-complete and present efficient algorithms if the underlying graph is restricted to be series-parallel. In the first part of Section 5, we show for minimization problems how an approximation algorithm for the coordination

problem can be used to approximate problem (2-adapt). In the second part, we use the knapsack problem to show that this relationship does not hold for maximization problems in general. First, we present a PTAS for the knapsack coordination problem and, second, we show that the (2-adapt) knapsack problem is not at all approximable. We conclude the paper in Section 6 with an application of the (2-adapt) concept to the minimum spanning tree problem. We summarize the main results of the paper and post future research questions in Section 7.

1.2 The Concept of k-adaptability

We begin by presenting a formal definition of the concepts studied in the paper.

$$\min_{x^1,\dots,x^k\in\mathcal{X}} \max_{\tilde{c}\in\mathcal{U}} \min\left(\tilde{c}^T x^1,\dots,\tilde{c}^T x^k\right) \tag{k-adapt}$$

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

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

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 restriction to the case where $k=2$ is called problem (2-adapt) and defined as follows.

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

2 General Reformulation

We assume in the following that the number of scenarios k is fixed. We show how to transform problem (k-adapt) to polynomially many subproblems of the form

$$\min_{x^1,\dots,x^k\in\mathcal{X}} \sum_{i=1}^{n} f_i(x_i^1,\dots,x_i^k)$$

where f_i maps from $\mathbb{B}^k$ to $\mathbb{R}^+$.

In a first step, we derive a compact nonlinear reformulation of the problem. We begin by formulating the adversary problem $\max_{\tilde{c}\in\mathcal{U}}\min_{j\in[k]}\tilde{c}^T x^j$ as a linear program. We introduce a new variable z to replace $\min_{j\in[k]}\tilde{c}^T x^j$ with the constraints $z\le\tilde{c}^T x^j$ for $j\in[k]$. Further, we introduce variables $\delta_i\in[0,1]$ for $i\in[n]$ 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.

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

The corresponding dual of the problem is given by

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

where $\Delta = \left\{\alpha \ge 0 \mid \sum_{j=1}^{k} \alpha_j = 1\right\}$. We use $X := (x^1, \ldots, x^k)$ to denote all k solutions. The optimal value of the linear program Eval(X) is equal to the objective value of solution X. Using this reformulation of the adversary problem, we get the following nonlinear reformulation of (k-adapt).

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

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

$$\min_{X,\alpha} \; \sum_{j=1}^{k} \alpha_j c^T x^j + \left\| \begin{pmatrix} \sum_{j=1}^{k} \alpha_j d_i x_i^j \\ \vdots \\ \sum_{j=1}^{k} \alpha_j d_i x_i^j \end{pmatrix} \right\|^{(\Gamma)} \qquad \text{(NL}'\text{)}$$

$$\text{s.t. } \alpha \in \Delta$$
$$X \in \mathcal{X}^k$$

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 $\gamma_i = \max(0, \sum_{j=1}^{k} \alpha_j d_i x_i^j - \theta)$. Further, if we fix X and α, (NL) reduces to

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

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

$$h_X(\alpha) := \sum_{j=1}^{k} \alpha_j c^T x^j + \left\| \begin{pmatrix} \sum_{j=1}^{k} \alpha_j d_1 x_1^j \\ \vdots \\ \sum_{j=1}^{k} \alpha_j d_n x_n^j \end{pmatrix} \right\|^{(\Gamma)}$$

Further, we define $I = \{(h, l) \mid h < l, h \in [n], l \in [n]\}$ and

$$\mathcal{A}_s^X = \left\{ \alpha \in \Delta \middle| \sum_{j=1}^{k} \alpha_j d_h x_h^j s_{(h,l)} \leq \sum_{j=1}^{k} \alpha_j d_l x_l^j s_{(h,l)} \quad \forall (h, l) \in I \right\}$$

where $s \in \{-1, 1\}^{|I|}$. For $s_{(h,l)} = 1$ the corresponding constraint is

$$\sum_{j=1}^{k} \alpha_j d_h x_h^j \leq \sum_{j=1}^{k} \alpha_j d_l x_l^j$$

and for $s_{(h,l)} = -1$ the corresponding constraint is

$$\sum_{j=1}^{k} \alpha_j d_h x_h^j \geq \sum_{j=1}^{k} \alpha_j d_l x_l^j$$

We call these constraints in the following *dividing constraints*. Observe that $\Delta = \bigcup_{s\in\{-1,1\}^{|I|}} \mathcal{A}_s^X$. Further, $\mathcal{A}_s^X$ is a bounded polytope, its vertices are denoted by $\mathcal{V}(\mathcal{A}_s^X)$ (recall that a bounded polytope is pointed). Note that $h_X(\alpha)$ is linear on $\mathcal{A}_s^X$. Therefore, the minimum of $h_X(\alpha)$ is attained at $\mathcal{P}(X) := \bigcup_{s\in\{-1,1\}^{|I|}} \mathcal{V}(\mathcal{A}_s^X)$. Note that each vertex of $\mathcal{V}(\mathcal{A}_s^X)$ can be found by selecting a subset of the inequality constraints defining $\mathcal{A}_s^X$ of cardinality $k-1$ and replacing them with equality constraints. Denote by $I_1 = \{j_1, \ldots, j_t\}$ a subset of the sign constraints ($\alpha_j \geq 0$) and by $I_2 = \{(h_1, l_1), \ldots, (h_u, l_u)\} \subset I$ a subset of the dividing constraints, where $t + u = k - 1$. The corresponding linear equation system looks as follows

$$\begin{aligned} \alpha_{j_r} &= 0 && \forall r \in [t] \\ \sum_{j=1}^{k} \alpha_j d_{h_r} x_{h_r}^j &= \sum_{j=1}^{k} \alpha_j d_{l_r} x_{l_r}^j && \forall r \in [u] \\ \sum_{j=1}^{k} \alpha_j &= 1 \end{aligned}$$

Note that the resulting linear equation system is independent from the index vector s. Hence, all vertices in $\mathcal{P}(X)$ are found as a solution of one of these linear equation systems. Consider the set $\mathcal{A} := \bigcup_{X\in\mathcal{X}^k} \mathcal{P}(X)$. To obtain a point in $\mathcal{A}$ we first have to choose a subset of the sign and dividing constraints of cardinality $k-1$. For this choice we have $\binom{\binom{n}{2}+k}{k-1}$ many options. Next, we have to choose a solution $X \in \mathcal{X}^k$. Assume that the subset of constraints is fixed. The sign constraints are independent of X, for each dividing constraint there are $(2^k)^2$ different possibilities which can be defined by a solution X. Since we chose at most $k-1$ dividing constraints we have at most $(2^k)^{(2k-2)}$ different equation systems for a fixed subset of constraints. Overall, we can bound the cardinality of $\mathcal{A}$ by $\binom{\binom{n}{2}+k}{k-1}(2^k)^{(2k-2)}$, i.e. $|\mathcal{A}| \leq \binom{\binom{n}{2}+k}{k-1} 4^{k^2} \in O(n^{2k-2})$. We obtain that

$$\min_{\alpha\in\Delta, X\in\mathcal{X}^k} h_X(\alpha) = \min_{X\in\mathcal{X}^k} \min_{\alpha\in\Delta} h_X(\alpha)$$

$$
\begin{aligned}
&= \min_{X \in \mathcal{X}^k} \min_{\alpha \in \mathcal{P}(X)} h_X(\alpha) \\
&= \min_{X \in \mathcal{X}^k} \min_{\alpha \in \mathcal{A}} h_X(\alpha) \\
&= \min_{\alpha \in \mathcal{A}} \min_{X \in \mathcal{X}^k} h_X(\alpha)
\end{aligned}
$$

In conclusion, we can remove α as a variable from the optimization problem since we could enumerate over the discrete candidate set $\mathcal{A}$ in polynomial time. In the following we assume that α is fixed. Consider the problem

$$\min_{X \in \mathcal{X}^k} h_X(\alpha)$$

We introduce again variable θ to obtain the following formulation for the problem under consideration.

$$
\begin{aligned}
\min_{X,\theta} \ & \sum_{j=1}^{k} \alpha_j c^T x^j + \Gamma\theta + \sum_{i=1}^{n} \max\left(0, \sum_{j=1}^{k} \alpha_j d_i x_i^j - \theta\right) \\
\text{s.t. } & 0 \leq \theta \\
& X \in \mathcal{X}^k
\end{aligned}
$$

We define $g_{X,\alpha}(\theta) := \sum_{j=1}^{k} \alpha_j c^T x^j + \sum_{i=1}^{n} \max\left(0, \sum_{j=1}^{k} \alpha_j d_i x_i^j - \theta\right)$.

Note that for fixed X, the function $g_{X,\alpha}$ is piecewise linear. Further, the set of all breakpoints of $g_{X,\alpha}$ is contained in the set

$$\Theta_{X,\alpha} = \{0\} \cup \bigcup_{i=1}^{n} \left\{ \sum_{j=1}^{k} \alpha_j d_i x_i^j \right\}.$$

Again, we union over all solutions X to obtain $\Theta(\alpha) := \bigcup_{X \in \mathcal{X}^k} \Theta_{X,\alpha}$. To obtain a point of $\Theta(\alpha)$ we first have to chose an index $i \in [n]$ and then to select a solution $X \in \mathcal{X}^k$. After the index i is fixed there are at most 2^k different possible breakpoints since the breakpoint is uniquely defined by k binary values. We conclude that $|\Theta(\alpha)| \leq n2^k + 1 \in O(n)$. We obtain that

$$
\begin{aligned}
\min_{\theta \geq 0, X \in \mathcal{X}^k} \Gamma\theta + g_{X,\alpha}(\theta) &= \min_{X \in \mathcal{X}^k} \min_{\theta \geq 0} \Gamma\theta + g_{X,\alpha}(\theta) \\
&= \min_{X \in \mathcal{X}^k} \min_{\theta \in \Theta_{X,\alpha}} \Gamma\theta + g_{X,\alpha}(\theta) \\
&= \min_{X \in \mathcal{X}^k} \min_{\theta \in \Theta(\alpha)} \Gamma\theta + g_{X,\alpha}(\theta) \\
&= \min_{\theta \in \Theta(\alpha)} \min_{X \in \mathcal{X}^k} \Gamma\theta + g_{X,\alpha}(\theta)
\end{aligned}
$$

In conclusion, we can remove θ as a variable from the optimization problem since we can enumerate over all candidates in $\Theta(\alpha)$ in linear time. After α and θ are fixed the following optimization problem remains

$$\min_{X\in\mathcal{X}^k} \sum_{i=1}^{n}\sum_{j=1}^{k} \alpha_j c_i x_i^j + \max\left(0, \sum_{j=1}^{k} \alpha_j d_i x_i^j - \theta\right) \qquad (P_{sub})$$

Hence, by setting

$$f_i(x_i^1,\ldots,x_i^k) := \sum_{j=1}^{k} \alpha_j c_i x_i^j + \max\left(0, \sum_{j=1}^{k} \alpha_j d_i x_i^j - \theta\right)$$

we have transformed the subproblem to the form posted at the beginning.

Theorem 2.2. *Problem (k-adapt) can be solved by solving at most $O(n^{2k-1})$ problems of the form*

$$\min_{X\in\mathcal{X}^k} \sum_{i=1}^{n} f_i(x_i^1,\ldots,x_i^k)$$

Proof. For each $\alpha \in \mathcal{A}$ we have to consider $O(n)$ many values of $\theta \in \Theta(\alpha)$. Since there are $O(n^{2k-2})$ many values for α which needs to be consider we end up with a total of $O(n^{2k-1})$ many subproblems. □

3 The Coordination Problem

In this section, we consider the special case where $k = 2$. We show that the subproblem (P_{sub}) obeys an interesting combinatorial structure which allows use to define polynomial-time combinatorial algorithms for this problem. For the case where $k = 2$ problem (P_{sub}) simplifies to

$$\min_{x,y\in\mathcal{X}} \sum_{i=1}^{n} \alpha c_i x_i + (1-\alpha) c_i y_i + \max\left(0, \alpha d_i x_i + (1-\alpha) d_i y_i - \theta\right) \qquad (P(\alpha,\theta))$$

Due to symmetry we can assume that $\alpha \leq 0.5$. We introduce an additional set of variables $z \in \mathbb{B}^n$. We use z_i to indicate that both x_i and y_i are equal to one simultaneously. This allows us to reformulate the problem as the following IP. We call the resulting problem a coordination problem (P_{Coord}).

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

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

Table 1: Values for the objective function f_i in problem (P_{Coord}).

In the following, we discuss the correct choice of the cost vectors a, b and p. Recall that the objective function of problem (P_{sub}) is separable, i.e. can be written as $\sum_{i=1}^{n} f_i(x_i, y_i)$. We list the values of f_i for a fixed index i in Table 1.

The following choice for a, b and p ensure that problem (P_{sub}) is indeed equivalent to problem (P_{Coord}).

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

Note that $a \leq b$ and $p \geq 0$. He we can interpret the coordination problem (P_{Coord}) in the following way. We have to find two feasible solutions x and y to the original problem and the cost of each single solution is given by the cost vector a and b independently. However, if an element i is part of both solutions an additional penalty term p_i needs to be paid. The goal is to minimize the sum of the cost of both solutions as well as the sum of all penalty terms. This observation is the motivation to call these problems coordination problems, since both solutions x and y need to coordinate each other to avoid penalties and to improve their cost.

4 Complexity Results for Problem (2-adapt)

In this section, we first show that problem (P_{Coord}) can be solved for different classic combinatorial optimization problems efficiently. Recall that the solution of problem (2-adapt) requires the solution of $O(n^3)$ many problem of type (P_{Coord}) (Theorem 2.2). Hence, we can use these positive results to prove that problem (2-adapt) is polynomial solvable for these problems. We end the section, with a negative result in which we prove that the (2-adapt) shortest path problem is strongly NP-complete for general graphs, which implies also the NP-completeness for the corresponding coordination problem. However, if we restrict the graph class to series-parallel graphs, problem (P_{Coord}) can be solved in linear time.

4.1 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 [11]).

Lemma 4.1. *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 Theorem 2.2 and Lemma 4.1, we obtain the next theorem.

Theorem 4.2. *Problem (2-adapt) of the unconstrained combinatorial problem can be solved in $O(n^4)$.*

4.2 The Matroid Maximization Problem

The underlying problem in this section is the problem of matroid maximization. A finite matroid is a pair $(E, \mathcal{I})$, where E is a finite set (the ground set) and $\mathcal{I}$ is a family of subsets of E (called the independent sets) with the following properties:

1. $\emptyset \in \mathcal{I}$
2. For $A \subset B \subset E$ it holds that $B \in \mathcal{I} \Rightarrow A \in \mathcal{I}$
3. For $A, B \in \mathcal{I}$ with $|A| < |B|$ it holds that $\exists e \in B \backslash A : A \cup \{e\} \in \mathcal{I}$

The *rank function* $r : 2^E \to \mathbb{N}$ of a matroid is defined as

$$r(B) = \max\{|A| : A \in \mathcal{I}, A \subset B\}.$$

Given a profit vector $p' \in \mathbb{Z}^{|E|}$, the matroid maximization problem is defined as

$$\max_{A \in \mathcal{I}} p'(A)$$

where $p'(A) := \sum_{e \in A} p'_e$. It is known that the matroid maximization problem can be formulated by the following linear programming problem

$$\begin{aligned} \max \ & \sum_{e \in E} p'_e x_e && && \text{(LP-MM)} \\ \text{s.t.} \ & \sum_{e \in S} x_e \leq r(S) && \forall S \subset E \end{aligned}$$

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

We call the constraints defined for each subset $S \subset V$ the *rank* constraints. Using the ellipsoid method combined with an efficient separation oracle allows us to solve the linear programming problem efficiently. Observe that a point $x \geq 0$ lies in the feasible set of (LP-MM) if and only if $\min_{S \subset E} r(S) - x(S) \geq 0$. Hence the separation oracle can be obtained by solving this minimization problem. Note that function $h(S) := r(S) - x(S)$ is a submodular function.

$$\begin{aligned} h(S_1) + h(S_2) &= r(S_1) + r(S_2) + x(S_1) + x(S_2) \\ &\geq r(S_1 \cap S_2) + r(S_1 \cup S_2) + x(S_1 \cap S_1) + x(S_1 \cup S_2) \\ &= h(S_1 \cap S_2) + h(S_1 \cup S_2) \end{aligned}$$

In [21] it is shown how a submodular function can be minimized in strongly polynomial time, if the function can be evaluated in strongly polynomial time. If we can decide efficiently if $S \in \mathcal{I}$ for all $S \subset E$, wen can also compute $h(S)$ efficiently. Hence, the only assumption we post in the following on the matroid $(E, \mathcal{I})$ is to test efficiently if a set $S \in \mathcal{I}$.

For the equivalence of matroid maximization and the linear program (MM) it is crucial that the feasible set of the linear program is an integral polyhedron. We start by showing a structural lemma of the feasible set of (LP-MM). Given a solution x of (LP-MM). We say an element e is *fractional* (with respect to solution x) if $x_e \in (0, 1)$. Denote by $\delta(e) \in \mathbb{B}^m$ the incidence vector of element e. We say two different fractional elements e_1 and e_2 are *compatible* with each other if it 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-MM). We denote this definition of x^+ and x^- also as *shifting* solution x on elements e_1 and e_2 with shifting parameter ϵ. Denote by $C_x(e)$ the set of all compatible elements of element e with respect to solution x. We say an fractional element is *lonely* if it exists an $\epsilon > 0$ such that $x^+ = x + \epsilon\delta(e)$ and $x^- = x - \epsilon\delta(e)$ are both feasible for (LP-MM).

Lemma 4.3. *Let x be a solution of (LP-MM). Then, each fractional solution e is either lonely or the set of compatible elements $C_x(e)$ is not empty.*

The proof is given in the appendix.

Remark 4.4. *Note that Lemma 4.3 can be used to prove that each vertex of the feasible set of (LP-MM) is integral.*

It is important to note that the set of tight sets $\mathcal{F}$ (see the proof of Lemma 4.3) is the same for x, x^+, and x^-. Since the definition of compatible elements depends purely on $\mathcal{F}$, we conclude that, if two elements 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 4.5. *Let x be a solution of (LP-MM) and $(e'_1, e''_1), \ldots (e'_k, e''_k)$ a list of compatible pairs. Set $E' = \{e'_1, \ldots, e'_k\}$ and $E'' = \{e''_1, \ldots, e''_k\}$ and define $x^+ := x + \epsilon\delta(E') - \epsilon\delta(E'')$ and $x^- := x - \epsilon\delta(E') + \epsilon\delta(E'')$, where $\delta(M)$ denotes the incidence vector of set M. For sufficiently small ϵ, x^+ and x^- are both feasible for (LP-MM).*

The proof of Lemma 4.3 also reveals that the set of compatible elements guarantees the following *transitivity* property.

Observation 4.6. *Let e', e'', and e''' be pairwise different elements. Let e' be compatible with e'' and let e'' be compatible with e''', then e' is also compatible with e'''.*

Consider the coordination matroid maximization problem.

$$\max_{A\in\mathcal{I},B\in\mathcal{I}} a(A) + b(B) + p(A,B)$$

where $a(A) = \sum_{e\in A} a_e$, $b(B) = \sum_{e\in B} b_e$, and $p(A,B) = \sum_{e\in A\cap B} p_e$, where $a, b \in \mathbb{N}^n$ and $p \in \mathbb{Z}^n_{\leq 0}$. It can be formulated by the following integer programming formulation.

$$\begin{aligned}
\max\ & a^T x + b^T y + p^T z && && \text{(Co-MM-I)}\\
\text{s.t. }\ & x_e + y_e \leq z_e + 1 && \forall e \in E\\
& \sum_{e\in S} x_e \leq r(S) && \forall S \subset E\\
& \sum_{e\in S} y_e \leq r(S) && \forall S \subset E\\
& x_e \in \{0,1\} && \forall e \in E\\
& y_e \in \{0,1\} && \forall e \in E\\
& z_e \in \{0,1\} && \forall e \in E
\end{aligned}$$

Consider the linear relaxation of (Co-MM-I).

$$\begin{aligned}
\max\ & a^T x + b^T y + p^T z && && \text{(Co-MM)}\\
\text{s.t. }\ & x_e + y_e \leq z_e + 1 && \forall e \in E\\
& \sum_{e\in S} x_e \leq r(S) && \forall S \subset E\\
& \sum_{e\in S} y_e \leq r(S) && \forall S \subset E\\
& x_e \geq 0 && \forall e \in E\\
& y_e \geq 0 && \forall e \in E\\
& z_e \geq 0 && \forall e \in E
\end{aligned}$$

The constraints $z_e \leq 1\ \forall e \in E$ can be removed, since p is assumed to be non-positive. Note that (CO-MM) can be solved efficiently using the ellipsoid method combined with an efficient separation oracle for the rank constraints.

Lemma 4.7. *The vertices of the feasible set of (Co-MM) are integral.*

The proof is given in the appendix.

Lemma 4.7 shows that we can relax the integrality constraints of (Co-MM-I) without changing the problem. Hence, it suffices to solve (Co-MM) which can be done efficiently. We summarize the findings of this section in the next theorem.

Theorem 4.8. *The coordination matroid maximization problem can be solved in polynomial time.*

Transferring these results to the original problem (2-adapt) we obtain the following theorem.

Theorem 4.9. *Problem (2-adapt) of the matroid maximization problem can be solved in polynomial time.*

Remark 4.10. *Note that the matroid maximization problem is a generalization of the well known minimum spanning tree problem. Hence, we can transfer these results directly to the minimum spanning tree problem.*

Corollary 4.11. *The problem versions (2-adapt) and (P_{Coord}) of the minimum spanning tree problem can be solved in polynomial time.*

4.3 The Selection Problem

The task of the selection problem is to select 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 [19]). The corresponding problem (P_{Coord}) can be represented by the following IP formulation.

$$
\begin{aligned}
\min\ & a^T x + b^T y + p^T z & & \\
\text{s.t. }\ & x_i + y_i \leq z_i + 1 & & \forall i \in [n] \\
& \sum_{i=1}^{n} x_i = P & & \\
& \sum_{i=1}^{n} y_i = P & & \\
& x_i \in \{0,1\} & & \forall i \in [n] \\
& y_i \in \{0,1\} & & \forall i \in [n] \\
& z_i \in \{0,1\} & & \forall i \in [n]
\end{aligned}
$$

The linear relaxation is given by

$$
\begin{aligned}
\min\ & a^T x + b^T y + p^T z & & \\
\text{s.t. }\ & x_i + y_i \leq z_i + 1 & & \forall i \in [n] \\
& \sum_{i=1}^{n} x_i = P & &
\end{aligned}
$$

$$\sum_{i=1}^{n} y_i = P$$
$$x_i \in [0,1] \qquad \forall i \in [n]$$
$$y_i \in [0,1] \qquad \forall i \in [n]$$
$$z_i \in [0,1] \qquad \forall i \in [n]$$

Note that the selection problem is a special case of the matroid maximization problem. Recall that we have shown in Section 4.2 that the IP formulation of the corresponding coordination problem is integral, which means that the linear relaxation is equivalent to the original problem. Note that we can remove the constraints $z_i \leq 1$ since we have that $p_i \geq 0$ for all $i \in [n]$. The dual of the linear relaxation is given by

$$\begin{aligned} \max\ & P(v_1+v_2) - \sum_{i=1}^{n} u_i^1 + u_i^2 + \lambda_i && && \text{(D)} \\ \text{s.t.}\ & -u_i^1 - \lambda_i + v_1 \leq a_i && \forall i \in [n] \\ & -u_i^2 - \lambda_i + v_2 \leq b_i && \forall i \in [n] \\ & \lambda_i \leq p_i && \forall i \in [n] \\ & u^1, u^2, \lambda \geq 0 \end{aligned}$$

Observe that in an optimal solution $u_i^1 = \max(0, v_1 - a_i - \lambda_i)$ and $u_i^2 = \max(0, v_2 - b_i - \lambda_i)$ $\forall i \in [n]$. Hence, we can remove u^1 and u^2 and obtain the following problem.

$$\begin{aligned} \max\ & P(v_1+v_2) + \sum_{i=1}^{n} \min(0, a_i - v_1 + \lambda_i) + \min(0, b_i - v_2 + \lambda_i) - \lambda_i \\ \text{s.t.}\ & 0 \leq \lambda_i \leq p_i \quad \forall i \in [n] \end{aligned}$$

Lemma 4.12. *The optimal value of the maximization problem* $\max_{0 \leq \lambda_i \leq p_i} \min(0, a_i - v_1 + \lambda_i) + \min(0, b_i - v_2 + \lambda_i) - \lambda_i$ *is given by* $\min(0, a_i - v_1, b_i - v_2, a_i + b_i + p_i - v_1 - v_2)$.

Proof. Introducing two additional variables t_1 and t_2, the maximization problem can be written as the following linear program

$$\begin{aligned} \max\ & -t_1 - t_2 - \lambda_i \\ \text{s.t.}\ & -t_1 - \lambda_i \leq a_i - v_1 \\ & -t_2 - \lambda_i \leq b_i - v_2 \\ & \lambda_i \leq p_i \\ & t_1, t_2, \lambda_i \geq 0 \end{aligned}$$

The dual of this linear program is given by

$$\min\ (a_i - v_1)\alpha + (b_i - v_2)\beta + p_i\gamma$$

$$\begin{aligned} \text{s.t. } & \alpha + \beta - \gamma \leq 1 \\ & \alpha \leq 1 \\ & \beta \leq 1 \\ & \alpha, \beta, \gamma \geq 0 \end{aligned}$$

In an optimal solution one can set $\gamma = \max(0, \alpha + \beta - 1)$. Resulting in the following two dimensional optimization problem.

$$\begin{aligned} \min\ & (a_i - v_1)\alpha + (b_i - v_2)\beta + p_i \max(0, \alpha + \beta - 1) \\ \text{s.t. } & 0 \leq \alpha \leq 1 \\ & 0 \leq \beta \leq 1 \end{aligned}$$

Note that the function $L(\alpha, \beta) = (a_i - v_1)\alpha + (b_i - v_2)\beta + p_i \max(0, \alpha + \beta - 1)$ must attain its minimum at one of the four vertices of $[0, 1]^2$. Hence, the problem simplifies to $\min(L(0,0), L(1,0), L(0,1), L(1,1))$ which is equivalent to $\min(0, a_i - v_1, b_i - v_2, a_i + b_i + p_i - v_1 - v_2)$. □

Using Lemma 4.12 we can simplify the problem to a two dimensional optimization problem with variables v_1 and v_2.

$$\max\ P(v_1 + v_2) + \sum_{i=1}^{n} \min(0, a_i - v_1, b_i - v_2, a_i + b_i + p_i - v_1 - v_2) \qquad (2D)$$

Consider the functions $f_1(t) = \max_{v_2} P(t + v_2) + \sum_{i=1}^{n} \min(0, a_i - t, b_i - v_2, a_i + b_i + p_i - t - v_2)$ and $f_2(t) = \max_{v_1} P(v_1 + t) + \sum_{i=1}^{n} \min(0, a_i - v_1, b_i - t, a_i + b_i + p_i - v_1 - t)$. Note that a function $f(x) = \min_{y \in C} h(x, y)$ is convex if h is jointly convex in (x, y) and C is a convex (see [7]). Hence, we can conclude that f_1 and f_2 are concave.

Lemma 4.13. *The functions f_1 and f_2 can be computed in linear time.*

Proof. Consider function f_1 (f_2 is analogous). To compute $f_1(t)$ we have to solve the maximization problem

$$\max_{v_2}\ P(t + v_2) + \sum_{i=1}^{n} \min(0, a_i - t, b_i - v_2, a_i + b_i + p_i - t - v_2)$$

Define $m_i = \min(0, a_i - t)$ and $h_i = \min(b_i, a_i + b_i + p_i - t)$. Note that the maximization problem is equivalent to

$$\max_{v_2}\ P(t + v_2) + \sum_{i=1}^{n} \min(m_i, h_i - v_2)$$

Hence, we are maximizing a piecewise linear function with breakpoints at $h_i - m_i$. Note that the set of all breakpoints can be computed in linear time. The maximum of this piecewise linear function is at the P^{th} smallest breakpoint which can be found in linear time. After the P^{th} smallest breakpoint is found, one can compute the optimal value of the maximization problem in linear time. □

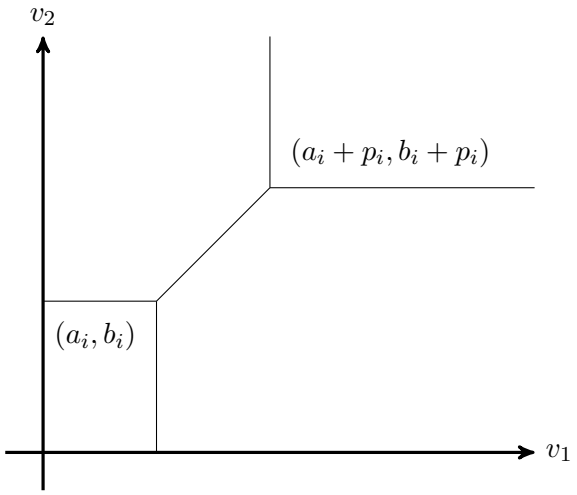


Figure 1: The set of breakpoints of the piecewise linear function $\min(0, a_i - v_1, b_i - v_2, a_i + b_i + p_i - v_1 - v_2)$ is given by five line segments.

Consider the piecewise linear function $\min(0, a_i - v_1, b_i - v_2, a_i + b_i + p_i - v_1 - v_2)$. The breakpoints form a set as shown in Figure 1. The lines intersect at the points (a_i, b_i) and $(a_i + p_i, b_i + p_i)$.

If multiple of these functions are added the set of breakpoints is obtained by combining the set of breakpoints of each individual function, see Figure 2.

Note that the maximum of the piecewise linear function $P(v_1 + v_2) + \sum_{i=1}^{n} \min(0, a_i - v_1, b_i - v_2, a_i + b_i + p_i - v_1 - v_2)$ can be found at one of the intersection points shown in Figure 2. Denote by $\mathcal{V}$ the set of all intersection points. Observe that for each $v \in \mathcal{V}$ either $v_1 \in \mathcal{A} := \{a_i : i \in [n]\} \cup \{a_i + p_i : i \in [n]\}$ or $v_2 \in \mathcal{B} := \{b_i : i \in [n]\} \cup \{b_i + p_i : i \in [n]\}$. Using this observation we can reduce our problem to

$$\max\left(\max_{t \in \mathcal{A}} f_1(t), \max_{t \in \mathcal{B}} f_2(t)\right)$$

A naive approach is to compute $f_1(t)$ for all $t \in \mathcal{A}$ and $f_2(t)$ for all $t \in \mathcal{B}$. This naive approach has a time complexity of $O(n^2)$, since $|\mathcal{A}|$ and $|\mathcal{B}|$ are in $O(n)$ and computing f_1 and f_2 can be done in $O(n)$ time (see Lemma 4.13). But, since f_1 and f_2 are concave, it is not necessary to consider each value in $\mathcal{A}$ and $\mathcal{B}$. Using golden section search over $\mathcal{A}$ and $\mathcal{B}$, it is sufficient to consider only $O(\log(n))$ many values. This allows us to find the optimal solution (v_1^*, v_2^*) of problem (2D) in $O(n \log(n))$ time. The optimal values λ^* can be found by solving $\max_{0 \le \lambda_i \le p_i} \min(0, a_i - v_1^* + \lambda_i) + \min(0, b_i - v_2^* + \lambda_i) - \lambda_i$ which can be done in constant time for fixed i. Hence, λ^* can be computed in $O(n)$ time. Setting $u_i^{1*} = \max(0, v_1^* - a_i - \lambda_i^*)$ and $u_i^{2*} = \max(0, v_2^* - b_i - \lambda_i^*)$, we obtain the optimal solution $(v_1^*, v_2^*, \lambda^*, u^{1*}, u^{2*})$ of (D).

Lemma 4.14. *Given the optimal solution of (D) one can compute an optimal solution of (P_{coord}) in $O(n)$.*

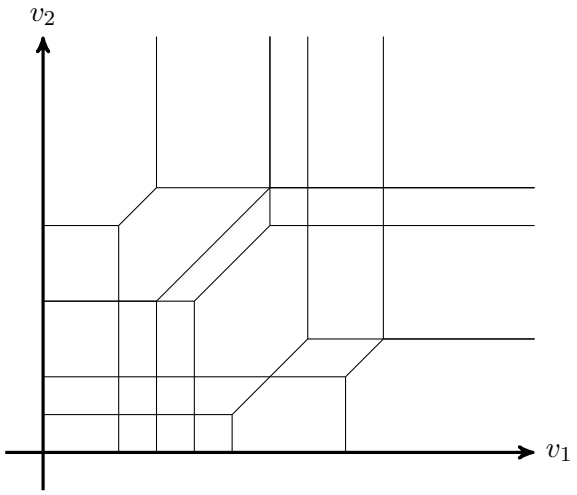


Figure 2: To obtain the set of breakpoints of the piecewise linear function $\sum_{i=1}^{n} \min(0, a_i - v_1, b_i - v_2, a_i + b_i + p_i - v_1 - v_2)$ one needs to combine the set of breakpoints of each individual function.

The proof is given in the appendix.

Due to Lemma 4.14, the overall running time of the algorithm is bounded by $O(n \log(n))$. Combining this result with Theorem 2.2 we obtain the following theorem.

Theorem 4.15. *Problem (2-adapt) of the selection problem can be solved in* $O(n^4 \log(n))$.

4.4 The Shortest Path Problem

Unfortunately, it turns out that 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 4.16. *Problem (2-adapt) is strongly NP-complete for the shortest path problem, even if* $\Gamma = 1$.

Proof. We give a reduction from the 2-disjoint path problem, which is known to be strongly NP-complete (see [14]). 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 a $s_1 - t_1$ path P and a $s_2 - t_2$ 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),

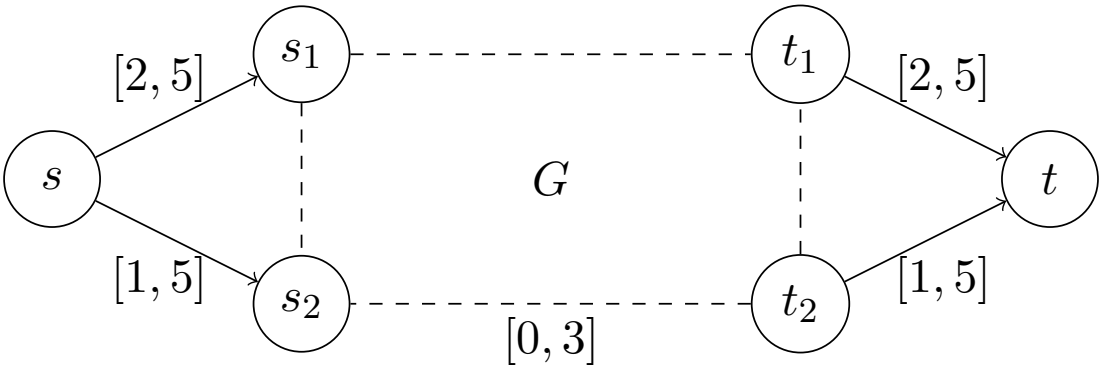


Figure 3: We construct graph G' by adding nodes s and t to G and by connecting them appropriately to s_1, s_2, t_1 and t_2. The intervals on the edges define the corresponding cost intervals. The cost interval of all edges from G are set to $[0, 3]$.

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

Corollary 4.18. *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 4.16. 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 4.18 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 class 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 4.19. *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 [12]). 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

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

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. □

The combination of Theorem 2.2 and Lemma 4.19 leads to the following theorem.

Theorem 4.20. *Problem (2-adapt) of the shortest path problem on series-parallel graphs can be solved in $O(m^4)$.*

5 Approximating Problem (2-adapt)

In the first part of this section, we show for minimization problems that approximation algorithms for (P_{Coord}) can be used to approximate (2-adapt). In the second part, we investigate approximation properties for the knapsack problem, which is a classic maximization problem. Interestingly, it turns out that it exists a PTAS for the (P_{Coord}) knapsack problem but the (2-adapt) knapsack problem is not at all approximable. Hence, it is essential for the proof of the first part that the underlying problem is a minimization problem.

5.1 Approximating (P_{Coord}) to approximate Problem (2-adapt)

Assume we have given an β-approximation algorithm for a minimization 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 $P(\alpha, \theta)$ (see Section 3). Denote by $(x_{(\alpha,\theta)}, y_{(\alpha,\theta)})$ the approximate solution of $P(\alpha, \theta)$.

Algorithm 1 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 $(x_{(\alpha^*,\theta^*)}, y_{(\alpha^*,\theta^*)})$

Theorem 5.1. *Algorithm 1 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 1. From Section 2, we know that there exists $\alpha^* \in \mathcal{A}$ and $\theta^* \in \Theta(\alpha^*)$ and a solution (x^*, y^*) such that

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

where $f_i(x_i, y_i, \alpha, \theta) = c_i x_i \alpha + c_i y_i (1-\alpha) + \max(0, d_i x_i \alpha + d_i y_i (1-\alpha) - \theta)$. Denote by (x', y') the β-approximate solution computed for $P(\alpha^*, \theta^*)$ by Algorithm 1 and by $(\tilde{x}, \tilde{y})$ the solution returned by Algorithm 1. 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 problem $P(\alpha^*, \theta^*)$ it holds that

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

Recall that the objective function of problem $P(\alpha, \theta)$ is equal to $\sum_{i=1}^{n} f_i(x, y, \alpha, \theta)$. Lastly, we have to observe that

$$\begin{aligned}
&Eval((x', y')) = \\
&\min_{\substack{0\leq\alpha\leq 1\\ 0\leq\theta}} c^T x' \alpha + c^T y'(1-\alpha) + \Gamma\theta + \sum_{i=1}^{n} \max(0, d_i x_i' \alpha + d_i y_i'(1-\alpha) - \theta) \leq \\
&c^T x' \alpha^* + c^T y'(1-\alpha^*) + \Gamma\theta^* + \sum_{i=1}^{n} \max(0, d_i x_i' \alpha^* + d_i y_i'(1-\alpha^*) - \theta^*) = \\
&\Gamma\theta^* + \sum_{i=1}^{n} f_i(x_i', y_i', \alpha^*, \theta^*)
\end{aligned}$$

Using $\beta \geq 1$ 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 5.2 is an immediate consequence of Theorem 5.1.

Corollary 5.2. *If the underlying problem is a minimization problem, an FPTAS (PTAS) for (P_{coord}) can be transferred to an FPTAS (PTAS) for problem (2-adapt).*

5.2 The Knapsack Problem

We start this chapter with an detailed analysis of (P_{Coord}) for the knapsack problem. The feasible set of the knapsack problem is specified by a strictly positive weight vector $w \in \mathbb{N}^+$ and a capacity $W \in \mathbb{N}$, the feasibility set is $\mathcal{X} = \{x \in \mathbb{B}^n : \sum_{i=1}^{n} w_i x_i \leq W\}$. Note that we can assume without loss of generality that $w_i \leq W$ for all i, since each item with $w_i > W$ can be removed from the instance. The integer programming formulation of (P_{Coord}) is given by

$$\begin{aligned}
\max\ & a^T x + b^T y + p^T z && && \text{(K)} \\
\text{s.t. } & x_i + y_i \leq z_i + 1 && \forall i \in [n] \\
& \sum_{i=1}^{n} w_i x_i \leq W \\
& \sum_{i=1}^{n} w_i y_i \leq W \\
& x_i \in \{0, 1\} && \forall i \in [n] \\
& y_i \in \{0, 1\} && \forall i \in [n] \\
& z_i \in \{0, 1\} && \forall i \in [n]
\end{aligned}$$

Note that the knapsack problem is a maximization problem with positive reward vectors a and b. Therefore the penalty vector p for sharing one item in both solutions is negative.

We know already that the knapsack problem is an NP-complete problem. Hence, problem (K) can not be expected to be solved exactly in polynomial time. Therefore, it is interesting to search for exact algorithms with pseudopolynomial running time or for approximation algorithms with polynomial running time. We begin the analysis of the problem with an exact algorithm with pseudopolynomial running time.

Theorem 5.3. *Problem (P_{coord}) of the knapsack problem can be solved in time $O(nW^2)$.*

Proof. We use dynamic programming to compute the table $T(k, W_1, W_2)$ for $i \in [n], W_1 \in [W]$ and $W_2 \in [W]$. The entry in $T(k, W_1, W_2)$ denotes the maximum profit of the restricted problem.

$$
\begin{aligned}
\max \ & \sum_{i=1}^{k} a_i x_i + b_i y_i + p_i z_i & \\
\text{s.t. } & x_i + y_i \leq z_i + 1 & \forall i \in [k] \\
& \sum_{i=1}^{k} w_i x_i \leq W_1 & \\
& \sum_{i=1}^{k} w_i y_i \leq W_2 & \\
& x_i \in \{0, 1\} & \forall i \in [k] \\
& y_i \in \{0, 1\} & \forall i \in [k] \\
& z_i \in \{0, 1\} & \forall i \in [k]
\end{aligned}
$$

Note that the optimal objective value of the original problem is given by $T(n, W, W)$. To obtain the claimed running time of $O(nW^2)$ we have to prove that dynamic programming can be used to compute each value in T in constant time. The values in T are computed with the following scheme:

$$
T(k+1, W_1, W_2) = \max \left\{ \begin{array}{l} T(k, W_1, W_2), \\ T(k, W_1 - w_i, W_2) + a_i, \\ T(k, W_1, W_2 - w_i) + b_i, \\ T(k, W_1 - w_i, W_2 - w_i) + a_i + b_i + p_i \end{array} \right\}
$$

The initial and boundary values of T are given by

$$
T(k, W_1, W_2) = \begin{cases} -\infty, & \text{if } W_1 < 0 \text{ or } W_2 < 0, \\ 0, & \text{if } k = 0, W_1 \geq 0, \text{ and } W_2 \geq 0 \end{cases}
$$

Using induction, it is straightforward to show that the values of $T(k, W_1, W_2)$ denote the maximum profits of the restricted problems. □

The goal of the next steps is to define a PTAS for Problem (K). First, we consider the linear relaxation of the problem:

$$\begin{aligned} \max\ & a^T x + b^T y + p^T z && && \text{(Rel-K)} \\ \text{s.t. } & x_i + y_i \leq z_i + 1 && \forall i \in [n] \\ & \sum_{i=1}^{n} w_i x_i \leq W \\ & \sum_{i=1}^{n} w_i y_i \leq W \\ & x_i \in [0,1] && \forall i \in [n] \\ & y_i \in [0,1] && \forall i \in [n] \\ & z_i \in [0,1] && \forall i \in [n] \end{aligned}$$

Since (Rel-K) is only a relaxation of (K) an optimal solution might contain fractional values. Let (x, y, z) be a solution of (Rel-K) we say index i is *fractional* if $x_i \in (0,1)$ or $y_i \in (0,1)$. The next lemma shows that each vertex of the feasible set of (Rel-K) hast at most two fractional indices.

Lemma 5.4. *Each vertex of (Rel-K) has at most two fractional indices.*

The proof is given in the appendix.
The following algorithm makes use of the observation of Lemma 5.4.

Algorithm 2 $(Rel-3)$

1: Compute an optimal vertex (x', y', z') of problem $(Rel-K)$, which has at most two fractional indices i and j.
2: Define (x^1, y^1, z^1) by all non fractional indices of (x', y', z'), i.e. $x^1_k = x_k$, $y^1_k = y_k$, and $z^1_k = z_k$ for all $k \notin \{i, j\}$ and $x^1_i = x^1_j = y^1_i = y^1_j = z^1_i = z^1_j = 0$ for i and j.
3: Define (x^2, y^2, z^2) by $x^2_k = y^2_k = z^2_k = 0$ for all $k \neq i$ and $(x^2_i, y^2_i) = \operatorname{argmax}_{x\in\{0,1\}, y\in\{0,1\}} a_i x + b_i y + p_i \max(0, x + y - 1)$ and $z^2_i = \max(0, x^2_i + y^2_i - 1)$.
4: Define (x^3, y^3, z^3) by $x^3_k = y^3_k = z^3_k = 0$ for all $k \neq j$ and $(x^3_j, x^3_j) = \operatorname{argmax}_{x\in\{0,1\}, y\in\{0,1\}} a_j x + b_j y + p_j \max(0, x + y - 1)$ and $z^3_j = \max(0, x^3_j + y^3_j - 1)$.
5: Let $(\tilde{x}, \tilde{y}, \tilde{z})$ be the best solution of (x^1, y^1, z^1), (x^2, y^2, z^2), and (x^3, y^3, z^3).
6: **return** $(\tilde{x}, \tilde{y}, \tilde{z})$.

Theorem 5.5. *Algorithm* $(Rel-3)$ *is a* $\frac{1}{3}$ *approximation algorithm for problem* (K).

Proof. Consider Algorithm (Rel-3) described as Algorithm 2. Denote by f^* the optimal objective value of problem (K), by f' the optimal value of the relaxed problem and by $\tilde{f}$ the value of the solution returned by Algorithm (Rel-3). Further, we set f_l equal to the objective value of solution (x^l, y^l, z^l) for $l = 1, 2, 3$. The claimed approximation ratio follows, since

$$f^* \leq f' \leq f_1 + f_2 + f_3 \leq 3\tilde{f}.$$

The first and third inequality are straightforward, since f' is the optimal value of the relaxed problem and $f_l \leq \tilde{f}$. The second inequality holds since

$$\begin{aligned}
f' &= a^T x' + b^T y' + p^T z' \\
&= \textstyle\sum_{k \notin \{i,j\}} a_k x_k^1 + b_k y_k^1 + p_k z_k^1 + a_i^T x_i' + b_i^T y_i' + p_i^T z_i' + a_j^T x_j' + b_j^T y_j' + p_j^T z_j' \\
&= f_1 \quad + \quad a_i^T x_i' + b_i^T y_i' + p_i^T z_i' + a_j^T x_j' + b_j^T y_j' + p_j^T z_j' \\
&\leq f_1 \quad + \quad \max_{x \in [0,1], y \in [0,1]} a_i^T x + b_i^T y + p_i^T \max(0, x + y - 1) \\
&\qquad + \quad \max_{x \in [0,1], y \in [0,1]} a_j^T x + b_j^T y + p_j^T \max(0, x + y - 1) \\
&= f_1 \quad + \quad \max_{x \in \{0,1\}, y \in \{0,1\}} a_i^T x + b_i^T y + p_i^T \max(0, x + y - 1) \\
&\qquad + \quad \max_{x \in \{0,1\}, y \in \{0,1\}} a_j^T x + b_j^T y + p_j^T \max(0, x + y - 1) \\
&= f_1 + f_2 + f_3
\end{aligned}$$

□

The idea of the PTAS is to guess the set $L \subset [n]$ of ℓ items of the optimal solution which yield the most reward. For a solution (x, y, z) the reward of item i is defined by $r_i = a_i x_i + b_i y_i + p_i z_i$. The choice of ℓ depends on the desired approximation guarantee ϵ. After $L \subset [n]$ has been chosen a restricted problem instance needs to be solved.

Assume that (x, y, z) is the solution for which the values of x_i, y_i and z_i are guessed for all $i \in L$. As a first step, all items in L are removed from the instance by introducing the constraints $x_i = 0, y_i = 0$, and $z_i = 0$ for all $i \in L$. In the second step, we update the weights for the weight constraints to $W_1 = W - w^T x$ and $W_2 = W - w^T y$ and remove all items which lead to an immediate violation of these weight constraints by introducing the constraints $x_i = 0$ if $w_i > W_1$ and $y_i = 0$ if $w_i > W_2$ for all $i \notin L$. In the last step, all possibilities are removed, which could lead to a higher reward as it is done by the items in L. Denote by $B = \min_{i \in L} r_i$ the smallest reward of all items in L. We introduce the constraints $x_i = 0$ if $a_i > B$, $y_i = 0$ if $b_i > B$, and $z_i = 0$ if $a_i + b_i + p_i > B$ for all $i \notin L$. We denote by $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$ the so defined instance, where I_x, I_y, and I_z is the index set of all variables which are fixed to 0. The restricted instance $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$ can be formulated by the following integer program.

$$\begin{aligned}
\max\ & a^T x + b^T y + p^T z && (\mathcal{I}(W_1, W_2, I_x, I_y, I_z)) \\
\text{s.t. } & x_i + y_i \leq z_i + 1 && \forall i \in [n] \\
& \sum_{i=1}^{n} w_i x_i \leq W_1 \\
& \sum_{i=1}^{n} w_i y_i \leq W_2 \\
& x_i \in \{0, 1\} && \forall i \in [n] \\
& y_i \in \{0, 1\} && \forall i \in [n] \\
& z_i \in \{0, 1\} && \forall i \in [n] \\
& x_i = 0 && \forall i \in I_x
\end{aligned}$$

$$y_i = 0 \qquad \forall i \in I_y$$
$$z_i = 0 \qquad \forall i \in I_z$$

Note that the result of Lemma 5.4 for (Rel-K) can be transferred to $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$.

Lemma 5.6. *Each vertex of the linear relaxation of $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$ has at most two fractional indices.*

Proof. The proof is identical to the proof of Lemma 5.4. □

Lemma 5.6 allows us to modify algorithm $(Rel-3)$ to obtain an $\frac{1}{3}$ approximation algorithm for $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$. Denote by $(Rel'-3)$ the modified version of $(Rel-3)$.

Algorithm 3 A PTAS for problem (K).

Chose $\epsilon > 0$.
Set $\ell = \min(\lceil \frac{2}{\epsilon} \rceil - 3, n)$.
Set $\hat{f} = 0$.
for all subsets $L \subset [n]$ with $|L| \leq \ell$ **do**
  Define $T = \left\{ (x,y,z) : \begin{array}{ll} x_i + y_i \geq 1,\ z_i = \max(0, x_i + y_i - 1) & \forall i \in L \\ x_i = y_i = z_i = 0 & \forall i \notin L \end{array} \right\}$.
  for all $(x,y,z) \in T$ **do**
    Set $B = \min_{i \in L} a_i x_i + b_i y_i + p_i z_i$.
    if $w^T x \leq W$ and $w^T y \leq W$ **then**
      Define $W_1 = W - w^T x$ and $W_2 = W - w^T y$.
      Define $I_x = L \cup \{i \notin L : a_i > B\} \cup \{i \notin L : w_i > W_1\}$.
      Define $I_y = L \cup \{i \notin L : b_i > B\} \cup \{i \notin L : w_i > W_2\}$.
      Define $I_z = L \cup \{i \notin L : a_i + b_i + p_i > B\}$.
      Use $(Rel' - 3)$ to find an approx. sol. of $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$.
      Denote by $(\tilde{x}, \tilde{y}, \tilde{z})$ the solution of $(Rel' - 3)$.
      Set $\tilde{f} = a^T \tilde{x} + b^T \tilde{y} + p^T \tilde{z}$.
      Set $f_L = a^T x + b^T y + p^T z$.
      if $f_L + \tilde{f} > \hat{f}$ **then**
        Set $\hat{f} = f_L + \tilde{f}$.
        Set $\hat{x}_i = x_i\ \forall i \in L$ and $\hat{x}_i = \tilde{x}_i\ \forall i \notin L$.
        Set $\hat{y}_i = y_i\ \forall i \in L$ and $\hat{y}_i = \tilde{x}_i\ \forall i \notin L$.
        Set $\hat{z}_i = z_i\ \forall i \in L$ and $\hat{z}_i = \tilde{x}_i\ \forall i \notin L$.
      end if
    end if
  end for
end for
return $(\hat{x}, \hat{y}, \hat{z})$.

Theorem 5.7. *Algorithm 3 defines a PTAS for problem (K).*

Proof. Consider the Algorithm 3. We start with the run time analysis. To enumerate over all subset $L \subset [n]$ with $|L| \leq \ell$ requires $O(n^\ell)$ time. Note that the set T consists of at most 3^ℓ elements since for each index $i \in L$ there are three possible choices for $(x_i, y_i, z_i) \in \{(1,0,0),(0,1,0),(1,1,1)\}$. Hence, the inner loop of both for loops is called $O((3n)^\ell)$ times. The running time of the inner loop is mainly defined by the running time T' of $(Rel'-3)$. Note that T' is polynomial in the input size, since the main part of $(Rel'-3)$ is the solution of a linear program. Since $\ell = \min(\lceil \frac{2}{\epsilon} \rceil - 3, n)$, the complete running time of the algorithm is given by $O((3n)^{\lceil \frac{2}{\epsilon} \rceil} T')$, which is polynomial in the input size for fixed ϵ.

Next, we prove the claimed approximation ratio. Denote by (x^*, y^*, z^*) the optimal solution of problem (K) and by f^* the optimal objective value. If the optimal solution consists of less than ℓ items, i.e. $|\{i : x_i^* + y_i^* \geq 1\}| < \ell$, Algorithm 3 will find this solution during the enumeration procedure. Hence, we can assume in the following that the optimal solution consists of more than ℓ items. Let L^* be the set of the ℓ items i of the optimal solution with the highest rewards $r_i^* = a_i x_i^* + b_i y_i^* + p_i z_i^*$. During the enumeration procedure we will find set L^* and the solution (x, y, z) which is equal to the optimal solution for all items $i \in L^*$, i.e. $x_i = x_i^*$, $y_i = y_i^*$, and $z_i = z_i^*$ for all $i \in L^*$. Note that the remaining items of the optimal solution not in L^* can be found by solving $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$. Denote the optimal objective value of this problem by $f_{\mathbb{B}}^{\mathcal{I}}$. Note that $f^* = \sum_{i \in L^*} r_i^* + f_{\mathbb{B}}^{\mathcal{I}}$. Denote by $f^{\mathcal{I}}$ the optimal value of the linear relaxation of $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$ and by $\tilde{f}$ the value of the approximate solution found by Algorithm $(Rel'-3)$ applied to $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$. Hence, we obtain for the value $\hat{f}$ of the solution returned by Algorithm 3 that $\hat{f} \geq \sum_{i \in L^*} r_i^* + \tilde{f}$. Recall that $\frac{1}{3} f_{\mathbb{B}}^{\mathcal{I}} \leq \tilde{f}$. In the following, we distinguish two cases.

Case 1: $\sum_{i \in L^*} r_i^* \geq \frac{\ell \epsilon}{2} f^*$
Combining the observations from above, we obtain

$$
\begin{aligned}
\hat{f} &\geq \sum_{i \in L^*} r_i^* + \tilde{f} \\
&\geq \sum_{i \in L^*} r_i^* + \frac{1}{3} f_{\mathbb{B}}^{\mathcal{I}} \\
&= \frac{2}{3} \sum_{i \in L^*} r_i^* + \frac{1}{3} \left(\sum_{i \in L^*} r_i^* + f_{\mathbb{B}}^{\mathcal{I}} \right) \\
&\geq \frac{2}{3} \frac{\ell \epsilon}{2} f^* + \frac{1}{3} f^* \\
&\geq \frac{(\frac{2}{\epsilon} - 3)\epsilon}{3} f^* + \frac{1}{3} f^* \\
&\geq \frac{(2 - 3\epsilon)}{3} f^* + \frac{1}{3} f^* \\
&\geq (1 - \epsilon) f^*
\end{aligned}
$$

Case 2: $\sum_{i \in L^*} r_i^* < \frac{\ell \epsilon}{2} f^*$

In this case, we get that $B = \min_{i \in L^*} r_i^* < \frac{\epsilon}{2} f^*$. Hence, the reward of a single item in the restricted instance $\mathcal{I}(W_1, W_2, I_x, I_y, I_z)$ is bounded by $\frac{\epsilon}{2} f^*$. Recall that $(Rel' - 3)$ chooses the best of three solutions, the first solution is defined by all integral values and the second and third solution consists of a single item corresponding to a fractional index. Hence, we conclude that

$$f_{\mathbb{B}}^{\mathcal{I}} \leq f^{\mathcal{I}} \leq f' + 2\frac{\epsilon}{2} f^* = f' + \epsilon f^*.$$

Which suffices to verify the approximation guarantee

$$(1 - \epsilon) f^* = \sum_{i \in L^*} r_i^* + f_{\mathbb{B}}^{\mathcal{I}} - \epsilon f^* \leq \sum_{i \in L^*} r_i^* + f' + \epsilon f^* - \epsilon f^* \leq \hat{f}.$$

□

Remark 5.8. *The PTAS is not well defined for $\epsilon > \frac{2}{3}$. In this case we use algorithm $(Rel - 3)$ to obtain an $\frac{1}{3}$ approximation for problem (K).*

The following negative approximation result proves that the PTAS is best possible for problem (K).

Theorem 5.9. *There exists no FPTAS for problem (K), unless $\mathsf{P} = \mathsf{NP}$.*

Proof. We prove this statement, by showing that an FPTAS for problem (K) could be used to solve the partition problem (PART) in polynomial time. Note that PART is known to be NP-complete.

An instance of PART consists of a set of natural numbers $\{s_1, \ldots, s_n\}$. The goal is to decide if there is a partition I_1, I_2 of the index set $[n]$ such that $\sum_{i \in I_1} s_i = \sum_{i \in I_2} s_i$. Consider the following instance of problem (K)

$$a_i = 1, b_i = 1, p_i = -1, w_i = s_i \qquad \forall i \in [n]$$

The capacity W is set to $\frac{1}{2} \sum_{i=1}^{n} s_i$. We claim that PART is a yes-instance if and only if the optimal value of the corresponding instance of (K) has objective value n. First, assume PART is a yes-instance. This means that it exists a partition I_1, I_2 such that $\sum_{i \in I_1} s_i = \sum_{i \in I_2} s_i$. We define the following solution (x', y', z') for problem (K): $x'_i = 1, y'_i = 0 \; \forall i \in I_1$, $x'_i = 0, y'_i = 1 \; \forall i \in I_2$ and $z' = 0$. It is easy to see that (x', y', z') is feasible for (K) and has objective value of n. Contrary, assume that an optimal solution (x^*, y^*, z^*) of (K) has objective value of at least n. Assume that $x_i^* y_i^* = 1$ in an optimal solution for some item i. In this case, the contribution to the objective function for this item i is given by $a_i + b_i + p_i = 1 + 1 - 1 = 1$. Note that setting either x_i^* or y_i^* to 0 decreases $w^T x^*$ and $w^T y^*$ and does not decrease the objective function. Hence, we can assume without loss of generality that $x_i^* y_i^* = 0$ for each $i \in [n]$. Further, we conclude that $z^* = 0$. Since the objective value of (x^*, y^*, z^*) is at least n we have that $I_x = \{i : x_i^* = 1\}$ and $I_y = \{i : y_i^* = 1\}$ forms a partition of $[n]$. Further, since (x^*, y^*, z^*) is a feasible solution we have that $\sum_{i \in I_x} s_i \leq \frac{1}{2} \sum_{i \in [n]} s_i$ and $\sum_{i \in I_y} s_i \leq \frac{1}{2} \sum_{i \in [n]} s_i$.

Summing both inequalities reveals that both inequalities must be tight, which proves that I_x and I_y solves problem PART.

Note that the optimal objective value of the constructed instance of (K) lies in $\{0, 1, \ldots, n\}$. Hence, a $\left(1 - \frac{1}{n+1}\right)$ approximation algorithm must find the optimal solution of the problem. Recall that an FPTAS for problem (K), means that we can find for each $\epsilon > 0$ in polynomial time with respect to the encoding length L of the problem and $\frac{1}{\epsilon}$ an $(1 - \epsilon)$ approximate solution. Therefore, by setting $\epsilon = \frac{1}{n+1}$, we could use the FPTAS to find in polynomial time an optimal solution of the problem and decide PART. □

Since knapsack is a maximization problem we cannot apply Corollary 5.2. In fact, it turns out that the (2-adapt) knapsack problem cannot be approximated at all as shown in the next theorem.

Theorem 5.10. *The (2-adapt) knapsack problem cannot be approximated at all if* $\mathsf{P} \neq \mathsf{NP}$.

Proof. We use a gap introducing reduction from PART. Given a list of natural numbers $s_1, \ldots, s_n$. We define the following (2-adapt) knapsack instance: We introduce n items with weights s_i and unit profits. The capacity of the knapsack is set to $\frac{1}{2}\sum_{i=1}^{n} s_i$. The uncertainty set is defined as $\mathcal{U} = \{\tilde{p} \in \mathbb{R}^n : \tilde{p}_i \in [0, 1], \sum_{i=1}^{n} \tilde{p}_i \geq 1\}$. Note that for maximization problems the profit values are assumed to stem from $[p_i - d_i, p_i]$, in contrast to minimization problems where the cost values are assumed to stem from $[c_i, c_i + d_i]$. Hence, the defined uncertainty set is in fact a bounded uncertainty set with budget $\Gamma = n - 1$.

Assume that the PART instance is a yes-instance, i.e., it exists a partition of $[n]$ into I_1 and I_2 such that $\sum_{i \in I_1} s_i = \sum_{i \in I_2} s_i$. Consider the following (2-adapt) knapsack solution (x, y): $x_i = 1, y_i = 0\ \forall i \in I_1$ and $x_i = 0, y_i = 1\ \forall i \in I_2$. It is easy to see that (x, y) is in fact a solution of the (2-adapt) knapsack problem. To compute the objective value of (x, y) we have to solve the following adversary problem

$$\min_{d_1 \leq |I_1|, d_2 \leq |I_2|, d_1 + d_2 \leq n-1} \max\{|I_1| - d_1, |I_2| - d_2\}$$

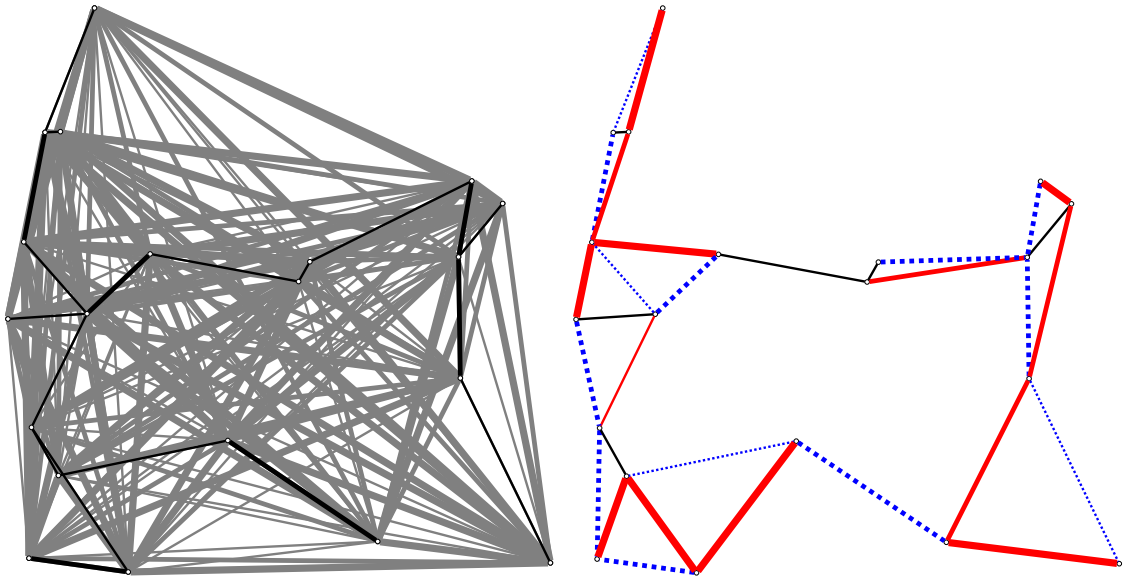
where d_1/d_2 is the amount of uncertainty budget which is used to reduce the profit of solution x/y. The optimal solution of this problem is $d_1 = |I_1| - 0.5$ and $d_2 = |I_2| - 0.5$ which results in an objective value of 0.5 for solution (x, y).

Next, assume that the PART instance is a no-instance. We claim that for each solution (x, y) of the (2-adapt) knapsack problem it must exist an item j which is neither part of x nor of y, i.e., $x_j = y_j = 0$. For the sake of contradiction assume that each item is part of either x of y. In this case, we observe that $\sum_{i=1}^{n} s_i \leq \sum_{i:x_i=1} s_i + \sum_{i:y_i=1} s_i \leq \sum_{i=1}^{n} s_i$. We conclude that x and y form a partition such that $\sum_{i:x_i=1} s_i = \sum_{i:y_i=1} s_i$ which contradicts the fact that the actual PART instance is a no-instance. Hence, we know that each solution (x, y) consists of at most $n - 1$ different items. Note that in the adversary problem these $n - 1$ different items can be set to have 0 profit. Therefore, the solution value of each (x, y) solution is 0.

The gap between 0.5 for yes-instances and 0 for no-instances leads to the conclusion that each approximation algorithm for (2-adapt) knapsack could be used to decide PART. □

6 Case Study

In this case study, we demonstrate the application of (2-adapt) 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 the performance



(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.

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

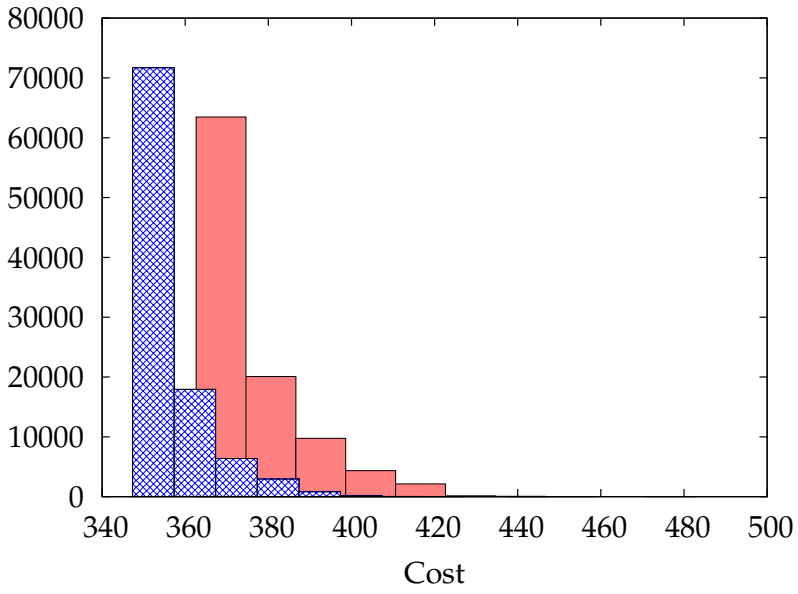


Figure 5: Simulated cost distribution. The distribution of the classic robust solution is shown in red. Choosing always the cheapest of the solution pair of (2-adapt) yields the blue (patterned) distribution.

improved considerably if the problem setup allows for a Plan B.

7 Conclusions and Future Research

In the first part of the paper we studied the general concept of k-adaptability. We proved how to transform, each problem (k-adapt) to polynomially many subproblems (for fixed k). This transformation allows a new approach to solve the general (k-adapt) problem. The main focus of this paper was the case where $k = 2$, i.e. problem (2-adapt). We proved that the underlying subproblems, called coordination problems obey an interesting theoretical structure. We analyzed these coordination problems for different classic combinatorial optimization problems and derived positive and negative complexity results depending on the underlying problem. Further, we showed that approximation algorithms for the coordination problem transfer to approximation algorithms for the (2-adapt) problem for minimization problems. For maximization problems, we disprove this relation by presenting a PTAS for the knapsack coordination problem and proving inapproximablity for the (2-adapt) knapsack problem. We conclude the paper with a case study of the (2-adapt) minimum spanning tree problem.

The following aspects are directly connected to the work presented in this paper and

seem promising for future research. The problem definition of the coordination problem (P_{Coord}) is rather general and may be interesting also in other areas such as game theory. Hence, studying this problem variant for further optimization problems seems valuable. We have shown that (P_{Coord}) can be solved in polynomial time for the matroid maximization problem. However, this method is only weakly polynomial since it relies on the ellipsoid method. Hence, the existence of a strongly polynomial time algorithm for this problem is an open question. Finally, more work should be done to study the more general problem (k-adapt).

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] Dimitris Bertsimas and Constantine Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

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

[9] 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.

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

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

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

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

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

[15] Grani A Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

[16] Grani A Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage distributionally robust binary programming. *Operations Research Letters*, 44(1):6–11, 2016.

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

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

[19] 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.

[20] 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.

[21] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346 – 355, 2000.

Appendix

Proof of Lemma 4.3:

Proof. Denote by $\mathcal{F}$ the set of all tight rank constraints for solution x, i.e., $\sum_{e\in F} x_e = r(F)$ for all $F \in \mathcal{F}$. We call $F \in \mathcal{F}$ a *tight set*. These are the critical constraints for the definition of lonely and compatible elements since all non-tight rank constraints stay fulfilled if we choose ϵ small enough.

Consider a fractional element $\hat{e}$. Assume that $\hat{e}$ is not contained in any tight constraint. It is obvious that $x^+ = x + \epsilon\delta(\hat{e})$ and $x^- = x - \epsilon\delta(\hat{e})$ are both feasible for (LP-MM) for small enough $\epsilon > 0$. In this case, element $\hat{e}$ is lonely.

We assume in the following that $\hat{e}$ is contained in at least one tight constraint. The goal is to show that the set of compatible elements $C_x(\hat{e})$ is not empty. First, we show that the set $\mathcal{F}$ fulfills the following property which is essential for the rest of the proof.

Claim 7.1. *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 $x(S) = \sum_{e \in S} x_e$ the sum over all elements of M. For any two sets S_1 and S_2, we have that $x(S_1) + x(S_2) = x(S_1 \cup S_2) + x(S_1 \cap S_2)$. For $F_1, F_2 \in \mathcal{F}$ and $F_1 \cap F_2 \neq \emptyset$, we have that

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

The first inequality holds since x is feasible and $F_1 \cap F_2 \neq \emptyset$. The second inequality holds, since the rank function r is submodular. 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.

Assume that element $\hat{e}$ is contained in such an inclusionwise minimal tight set F^*. We claim that all other fractional elements in this set are compatible to element $\hat{e}$, $C_x(\hat{e}) = \{e | e \in F^* \backslash \{\hat{e}\}, x_e \in (0,1)\}$. If $C_x(\hat{e}) = \emptyset$, we have that

$$\sum_{e \in F^*} x_e = \underbrace{\sum_{e \in F^* \backslash \{\hat{e}\}} x_e}_{\in \mathbb{N}} + \underbrace{x_{\hat{e}}}_{\in \mathbb{Q}} < \underbrace{r(F^*)}_{\in \mathbb{N}}.$$

This contradicts the fact that F^* is a tight set, wherefore we conclude that $C_x(\hat{e}) \neq \emptyset$.

Let e' be an arbitrary element in $C_x(\hat{e})$, we define $x^+ := x + \epsilon\delta(\hat{e}) - \epsilon\delta(e')$ and $x^- := x - \epsilon\delta(\hat{e}) + \epsilon\delta(e')$ for a sufficiently small $\epsilon > 0$ such that the sign constraints and all non-tight rank constraints remain fulfilled for x^+ and x^-. Further, we claim that x^+ and x^- are feasible for all tight constraints. Assume to have a tight set $F' \in \mathcal{F}$ with $\hat{e} \in F'$ and $e' \notin F$ or $\hat{e} \notin F'$ and $e' \in F$. Hence, $|F' \cap F^*| < |F^*|$ and, due to Claim 7.1, $F' \cap F^*$ is a tight set. Since this contradicts the fact that F^* is an inclusionwise minimal tight set, we conclude that each tight set F contains either $\hat{e}$ and e' or none of both. In both cases set F remains tight since

$$\sum_{e \in F} x_e^+ = \sum_{e \in F} x_e^- = \sum_{e \in F} x_e = r(F).$$

Next, we consider the case that $\hat{e}$ is not contained in an inclusion-wise minimal tight set. Consider the family $\mathcal{F}(\hat{e}) := \{F : F \in \mathcal{F}, \hat{e} \in F\}$. It must hold that $\hat{e}$ is contained in an inclusion-wise minimal tight set F^* with respect to family $\mathcal{F}(\hat{e})$. Inclusion-wise minimal with respect to a family means that no strict subset of this set is contained in the family. Consider the set $\mathcal{F}'$ of tight sets which are strict subsets of F^*, i.e.

$\mathcal{F}' = \{F : F \in \mathcal{F}, F \subsetneq F^*\}$. Observe that a set $F \in \mathcal{F}'$ with $\hat{e} \in F$ contradicts the fact that F^* is inclusion-wise minimal with respect to family $\mathcal{F}(\hat{e})$. Hence, we conclude that $\hat{e} \notin F$ for all $F \in \mathcal{F}'$. We define the set $F' = \bigcup_{F \in \mathcal{F}'} F$ and $R = F^* \backslash F'$. We claim that each element in the set $C_x(\hat{e}) = \{e : x_e \in (0,1), e \in R \backslash \{\hat{e}\}\}$ is compatible with $\hat{e}$. To show that $C_x(\hat{e}) \neq \emptyset$, we have to prove that $\sum_{e \in F'} x_e \in \mathbb{N}$.

Claim 7.2. $\sum_{e \in F'} x_e \in \mathbb{N}$.

Proof. We say a set M is *integral* if $\sum_{e \in M} x_e \in \mathbb{N}$. Assume that $\mathcal{F}' = \{F_1, \ldots, F_n\}$. We use a binary vector $b \in \mathbb{B}^n$ to denote all interesting subsets of F'. For all $b \in \mathbb{B}^n$, we denote by

$$S(b) = \{e : e \in F_i \ \forall i \text{ with } b_i = 1 \text{ and } e \notin F_i \ \forall i \text{ with } b_i = 0\}.$$

We proof by induction over the number of 0 entries of b that $S(b)$ is integral $\forall b \in \mathbb{B}$. Since $F' = \bigcup_{b \in \mathbb{B}^n} S(b)$ and $S(b_1) \cap S(b_2) = \emptyset \ \forall b_1 \neq b_2$, it follows that F' is integral if $S(b)$ is integral $\forall b \in \mathbb{B}$.

We denote by $\mathbf{1}_n, \mathbf{0}_n \in \mathbb{B}^n$ the all one (all zero) vector of length n. To start the induction consider the set $S(\mathbf{1}) = \bigcap_{i=1}^{n} F_i$. This set is either empty (in which it is by definition also integral) or due to Claim 7.1, we know that this set is itself a tight set and, therefore, $\sum_{e \in S(\mathbf{1})} x_e = r(S(\mathbf{1})) \in \mathbb{N}$.

Next, consider with out loss of generality the vector $b' = \begin{pmatrix} \mathbf{1}_{n-k-1} \\ \mathbf{0}_{k+1} \end{pmatrix}$. Observe that

$$\bigcap_{i=1}^{n-k-1} F_i = S(b') \cup \bigcup_{\tilde{b} \in \mathbb{B}^{k+1} \backslash \{0\}} S\left(\begin{pmatrix} \mathbf{1}_{n-k-1} \\ \tilde{b} \end{pmatrix}\right)$$

By induction it follows, that $S\left(\begin{pmatrix} \mathbf{1}_{n-k-1} \\ \tilde{b} \end{pmatrix}\right)$ is integral for all $\tilde{b} \in \mathbb{B}^{k+1} \backslash \{0\}$. Further, $\bigcap_{i=1}^{n-k-1} F_i$ is a tight set (or empty) and, hence, integral. Therefore, we conclude that $S(b')$ is also integral. This completes the step of the induction and, therefore, the proof of the claim. □

Using Claim 7.2 we arrive at a contradiction if $C_x(\hat{e}) = \emptyset$, since

$$\begin{aligned} \sum_{e \in F^*} x_e &= \sum_{e \in F'} x_e + \sum_{e \in R} x_e \\ &= \underbrace{\sum_{e \in F'} x_e}_{\in \mathbb{N}} + \underbrace{\sum_{e \in R \backslash \{\hat{e}\}} x_e}_{\in \mathbb{N}} + \underbrace{x_{\hat{e}}}_{\in \mathbb{Q}} < \underbrace{r(F^*)}_{\in \mathbb{N}} \end{aligned}$$

which contradicts that F^* is a tight set. Hence, we conclude that $C_x(\hat{e}) \neq \emptyset$.

Let e' be an arbitrary element in $C_x(\hat{e})$, we define $x^+ := x + \epsilon\delta(\hat{e}) - \epsilon\delta(e')$ and $x^- := x - \epsilon\delta(\hat{e}) + \epsilon\delta(e')$ for a sufficiently small $\epsilon > 0$ such that the sign constraints and all non-tight rank constraints remain fulfilled for x^+ and x^-. Further, we claim that x^+

and x^- are feasible for all tight constraints. We consider in the following two cases for an arbitrary tight set $F \in \mathcal{F}$.

Assume to have a tight set $F \in \mathcal{F}$ with $\hat{e} \in F$ and $e' \notin F$. In this case, we have that $|F \cap F^*| < |F^*|$ and, due to Claim 7.1, $F \cap F^*$ is a tight set. This contradicts the fact that F^* is an inclusionwise minimal tight set with respect to family $\mathcal{F}(\hat{e})$.

Assume to have a tight set $F \in \mathcal{F}$ with $\hat{e} \notin F$ and $e' \in F$. Due to Claim 7.1, $F \cap F^*$ is a tight set and $F \cap F^* \in \mathcal{F}'$. Hence, $e' \in F'$, which contradicts the fact that $e' \in R$.

Since both cases lead to contradictions, we conclude that each tight set F contains either $\hat{e}$ and e' or none of both. In both cases set F remains tight for x^+ and x^- since

$$\sum_{e \in F} x_e^+ = \sum_{e \in F} x_e^- = \sum_{e \in F} x_e = r(F).$$

This concludes the proof of the lemma, since we have shown for each fractional element e that it is either lonely or has a non-empty set of compatible elements.

□

Proof of Lemma 4.7:

Proof. Assume that (x, y, z) is a fractional solution of (Co-MM). The idea of the proof is to find two other feasible solutions (x^+, y^+, z^+) and (x^-, y^-, z^-) of (Co-MM) 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 (Co-MM) breaks into two feasible sets of (LP-MM) 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 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 element e (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 rank constraints for x and y.

Consider an element $e \in I_{x,z}$ (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 element $e \in I_x^0$ (for I_y^0 it works analogously). For this element, 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 element 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 element $e \in I_{\text{conflict}}$. 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 elements $e \in I_{\text{conflict}}$ are tight.

We summarize: We can change x_e for all elements 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 rank constraints into account. Using Lemma 4.3, 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 element $e \in I_{x,z}$, we also increase or decrease z_e, analogously for y_e and elements $e \in I_{y,z}$. 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 elements, 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.

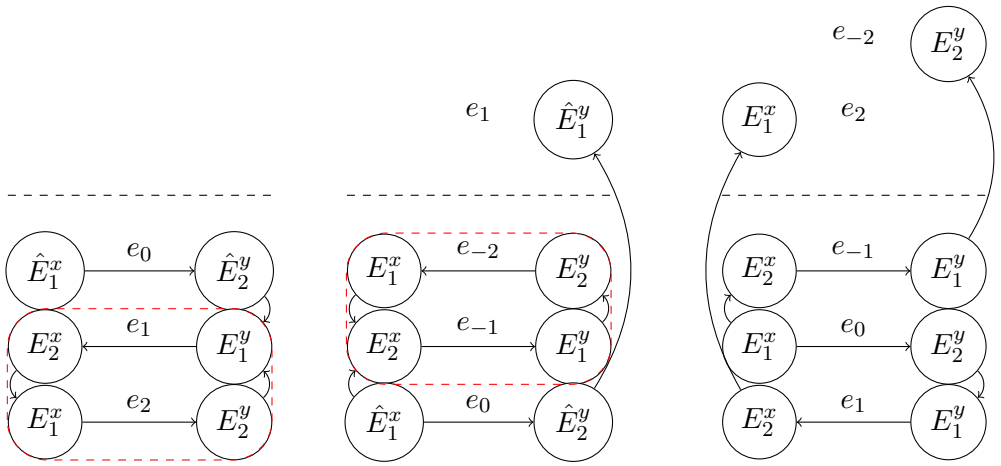
We begin with the easy case where $I_{\text{conflict}} = \emptyset$. In this case, we select a fractional element e from I_{agree}. Due to Lemma 4.3, e is either lonely or we find another element $e' \in C_x(e)$ which is compatible to e with respect to the solution x (Observe that if (x, y, z) is feasible for (Co-MM), then x and y are also feasible for (LP-MM)). If e is lonely, we define $x^+ := x + \epsilon\delta(e)$ and $x^- := x - \epsilon\delta(e)$. Otherwise, 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 (Co-MM) 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 elements. Instead, we will define a procedure that defines four sets of elements 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 contain only lonely elements and pairs of compatible elements with respect to solution x and that E_1^y and E_2^y contain only lonely elements and pairs of compatible elements with respect to solution y. This ensures feasibility of the rank 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:

$$x^+ := x + \epsilon\delta(E_1^x) - \epsilon\delta(E_2^x)$$

$$x^- := x - \epsilon\delta(E_1^x) + \epsilon\delta(E_2^x)$$
$$y^+ := y + \epsilon\delta(E_1^y) - \epsilon\delta(E_2^y)$$
$$y^- := y - \epsilon\delta(E_1^y) + \epsilon\delta(E_2^y)$$

Again, z^+ and z^- have to be chosen accordingly to x^+, x^- and y^+, y^-. Recall Observation 4.5, which is necessary to show that (x^+, y^+, z^+) and (x^-, y^-, z^-) are feasible for (Co-MM). 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 6: 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 element. 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 elements 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 n 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 element $e_0 \in I_{\text{conflict}}$ and add this element to $\hat{E}_1^x$. Element 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 element e.

We begin the y-step with starting element $e_0 \in \hat{E}_1^x$. Since $e_0 \in I_{\text{conflict}}$, we add e_0 to $\hat{E}_2^y$. If e_0 is lonely with respect to solution y, we terminate the forward execution and continue with the backward execution. Otherwise, we choose an element $e_1 \in C_y(e_0)$ 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 element.

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 4.6) that there exists another element $e_1' \in \hat{E}_1^y$ 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 element e_1.

Since $e_1 \in I_{\text{conflict}}$, we add e_1 to $\hat{E}_2^x$. If e_1 is lonely with respect to solution x, we terminate the forward execution and continue with the backward execution. Otherwise, we choose an element $e_2 \in C_x(e_1)$ 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 element.

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 4.6) that there exists another element $e_2' \in \hat{E}_1^x$ 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 n iterations of y- and x-steps, we have either found a cycle, in this case, we can skip

		Step	Starting Element	Selected Element	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 6.

the backward execution, or no cycle has been found. In the latter case, we start the backward execution with starting element 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, Cases y1 and x1 needs to be adapted and the cases that a lonely element is found.

In Cases y1 and x1 or if a lonely element is found, 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 removing all elements 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 contain only lonely elements or pairs of compatible elements. Further, the forward and backward execution of the process ensure that we have for all elements $e \in I_{\text{conflict}}$ 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 6 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.

□

Proof of Lemma 4.14:

Proof. Given an optimal solution of the dual $(v_1^*, v_2^*, u^{1*}, u^{2*}, \lambda^*)$ we get an optimal solu-

tion (x^*, y^*, z^*) of (P_{coord}) by finding a feasible solution which fulfills the complementary slackness conditions. In our case the complementary slackness conditions are for all $i \in [n]$

$$\begin{aligned} u_i^{1*} > 0 &\Rightarrow x_i^* = 1 \\ u_i^{2*} > 0 &\Rightarrow y_i^* = 1 \\ \lambda_i^* > 0 &\Rightarrow x_i^* + y_i^* = z_i^* + 1 \\ -u_i^{1*} - \lambda_i^* - v_1^* < a_i &\Rightarrow x_i^* = 0 \\ -u_i^{2*} - \lambda_i^* - v_2^* < b_i &\Rightarrow y_i^* = 0 \\ \lambda_i^* < p_i &\Rightarrow z_i^* = 0 \end{aligned}$$

The complementary slackness conditions separate the index set into different subsets

$$\begin{aligned} I_{0,0} &:= \{i : x_i^* = 0, y_i^* = 0\} \\ I_{1,0} &:= \{i : x_i^* = 1, y_i^* = 0\} \\ I_{0,1} &:= \{i : x_i^* = 0, y_i^* = 1\} \\ I_{1,1} &:= \{i : x_i^* = 1, y_i^* = 1\} \\ I_{[0,1],0} &:= \{i : x_i^* \in [0,1], y_i^* = 0\} \\ I_{[0,1],1} &:= \{i : x_i^* \in [0,1], y_i^* = 1\} \\ I_{0,[0,1]} &:= \{i : x_i^* = 0, y_i^* \in [0,1]\} \\ I_{1,[0,1]} &:= \{i : x_i^* = 1, y_i^* \in [0,1]\} \\ I^{\leq}_{[0,1],[0,1]} &:= \{i : x_i^* \in [0,1], y_i^* \in [0,1], x_i^* + y_i^* \leq 1\} \\ I^{=}_{[0,1],[0,1]} &:= \{i : x_i^* \in [0,1], y_i^* \in [0,1], x_i^* + y_i^* = 1\} \\ I^{\geq}_{[0,1],[0,1]} &:= \{i : x_i^* \in [0,1], y_i^* \in [0,1], x_i^* + y_i^* \geq 1\} \end{aligned}$$

For each set I_a^b we introduce two variables X_a^b and Y_a^b which denote the number of x and y variables which are set to 1 in the corresponding set. Recall that we can assume to have an optimal solution which is integral. Note that the values of $X_{0,0}, X_{1,0}, X_{0,1}, X_{1,1}, Y_{0,0}, Y_{1,0}, Y_{0,1}$, and $Y_{1,1}$, are already given. This transformation allows us to formulate the problem of finding a feasible solution which fulfills the complementary slackness conditions as a linear program with a constant number of constraints and variables.

$$\begin{aligned} X_{[0,1],0} + X_{[0,1],1} + X^{\leq}_{[0,1],[0,1]} + X^{=}_{[0,1],[0,1]} + X^{\geq}_{[0,1],[0,1]} &= P - |I_{1,0}| - |I_{1,1}| - |I_{1,[0,1]}| \\ Y_{0,[0,1]} + Y_{1,[0,1]} + Y^{\leq}_{[0,1],[0,1]} + Y^{=}_{[0,1],[0,1]} + Y^{\geq}_{[0,1],[0,1]} &= P - |I_{0,1}| - |I_{1,1}| - |I_{[0,1],1}| \\ X^{\leq}_{[0,1],[0,1]} + Y^{\leq}_{[0,1],[0,1]} &\leq |I^{\leq}_{[0,1],[0,1]}| \end{aligned}$$

$$
\begin{aligned}
X^{=}_{[0,1],[0,1]} + Y^{=}_{[0,1],[0,1]} &= |I^{=}_{[0,1],[0,1]}| \\
X^{\geq}_{[0,1],[0,1]} + Y^{\geq}_{[0,1],[0,1]} &\geq |I^{\geq}_{[0,1],[0,1]}| \\
X^b_a &\leq |I^b_a| && \forall (a,b) \\
Y^b_a &\leq |I^b_a| && \forall (a,b) \\
X^b_a &\geq 0 && \forall (a,b) \\
Y^b_a &\geq 0 && \forall (a,b)
\end{aligned}
$$

Since the number of constraints and variables is fixed the problem can be solved in constant time. Note that an optimal solution of this problem does not specify a unique solution of (P_{Coord}). However, we can transform in linear time each solution of this problem to a solution of (P_{Coord}). The so found solution must be optimal since the complementary slackness conditions are fulfilled.

□

Proof of Lemma 5.4:

Proof. Let $s = (x, y)$ be an vertex of (Rel-K) with at least three fractional indices. To simplify the presentation we omit the choice of the z variable, since z_i is either equal to $\max(0, x_i + y_i - 1)$ or to 1 for each vertex. We denote the i^{th} unit vector by e_i. We call a fractional index i *tight* if $x_i + y_i = 1$. We distinguish three cases. In each case we can show how s can be represented as a convex combination of two other distinct feasible points, which contradicts the assumption of s being a vertex.

Case 1: At least two indices are tight.
Let i, j be two tight indices. We define $\delta = \frac{1}{w_i} e_i - \frac{1}{w_j} e_j$. Let $\epsilon > 0$ be small enough such that $s_1 = (x + \epsilon\delta, y - \epsilon\delta)$ and $s_2 = (x - \epsilon\delta, y + \epsilon\delta)$ are both still fractional. Without loss of generality consider s_1 (s_2 is analogous). Since $w^T(x + \epsilon\delta) = w^T x + \epsilon w_i \frac{1}{w_i} - \epsilon w_j \frac{1}{w_j} = w^T x \leq W$, $w^T(y - \epsilon\delta) = w^T y - \epsilon w_i \frac{1}{w_i} + \epsilon w_j \frac{1}{w_j} = w^T y \leq W$, $x_i + \epsilon \frac{1}{w_i} + y_i - \epsilon \frac{1}{w_i} = x_i + y_i = 1$, and $x_j + \epsilon \frac{1}{w_j} + y_j - \epsilon \frac{1}{w_j} = x_j + y_j = 1$, s_1 is a feasible solution. The fact that $s = \frac{1}{2}(s_1 + s_2)$ contradicts the fact that s is a vertex.
Case 2: Exactly one index is tight.
Since we assume to have at least three fractional indices it follows that at least two fractional indices are not tight. Let i and j be these two indices. Denote by k the fractional, tight index. We distinguish two subcases:

Case 2.1: $(x_i \in (0,1), x_j \in (0,1))$ or $(y_i \in (0,1), y_j \in (0,1))$
Consider the case that $x_i \in (0,1), x_j \in (0,1)$ (the case that $y_i \in (0,1), y_j \in (0,1)$ is analogous). Let $\epsilon > 0$ be small enough such that $s_1 = (x + \frac{\epsilon}{w_i} e_i - \frac{\epsilon}{w_j} e_j, y)$ and $s_2 = (x - \frac{\epsilon}{w_i} e_i + \frac{\epsilon}{w_j} e_j, y)$ are both still fractional and i and j become no tight indices. Since $w^T(x + \frac{\epsilon}{w_i} e_i - \frac{\epsilon}{w_j} e_j) = w^T x + w_i \frac{\epsilon}{w_i} - w_j \frac{\epsilon}{w_j} = w^T x \leq W$, s_1 is still a feasible solution (s_2 is analogous). Again, we obtain that $s = \frac{1}{2}(s_1 + s_2)$ which contradicts the fact that s is a vertex.

Case 2.2: $(x_i \in (0,1), y_j \in (0,1))$ or $(y_i \in (0,1), x_j \in (0,1))$
Consider the case that $x_i \in (0,1), y_j \in (0,1)$ (the case that $y_i \in (0,1), x_j \in (0,1)$ is analogous). Define $\delta_{ik} = \frac{1}{w_i}e_i - \frac{1}{w_k}e_k$ and $\delta_{jk} = -\frac{1}{w_j}e_j + \frac{1}{w_k}e_k$. Consider solution $s_1 = (x + \epsilon\delta_{ik}, y + \epsilon\delta_{jk})$ and $s_2 = (x - \epsilon\delta_{ik}, y - \epsilon\delta_{jk})$, where $\epsilon > 0$ is chosen small enough such that all fractional indices stay fractional and no index becomes tight. Consider solution s_1 (s_2 is analogous). We have that $w^T(x+\epsilon\delta_{ik}) = w^T x + \epsilon\frac{w_i}{w_i} - \epsilon\frac{w_k}{w_k} = w^T x \leq W$ and $w^T(y+\epsilon\delta_{jk}) = w^T y - \epsilon\frac{w_j}{w_j} + \epsilon\frac{w_k}{w_k} = w^T y \leq W$. Hence, s_1 is feasible (s_2 is analogous). Again, we obtain that $s = \frac{1}{2}(s_1 + s_2)$ which contradicts the fact that s is a vertex.
Case 3: No index is tight.
Since at least three indices are fractional, we must have that either $x_i \in (0,1)$ and $x_j \in (0,1)$ or $y_i \in (0,1)$ and $y_j \in (0,1)$ for two indices i and j. In this case, we argue as in Case 2.1.

All cases lead to contradictions, hence, we can conclude that each vertex has at most 2 fractional indices. □