A Lagrangean Decomposition Approach for Robust Combinatorial Optimization*

Frank Baumann, Christoph Buchheim, and Anna Ilyina

Fakultät für Mathematik, Technische Universität Dortmund, Germany
{frank.baumann, christoph.buchheim, anna.ilyina}@tu-dortmund.de

Abstract. We address robust versions of combinatorial optimization problems, specializing on the discrete scenario case and the uncorrelated ellipsoidal uncertainty case. We present a branch and bound-algorithm for the min-max variant of these problems which uses lower bounds obtained from Lagrangean decomposition, allowing to separate the uncertainty aspect in the objective function from the combinatorial structure of the feasible set. For addressing the unrestricted binary subproblems arising in the uncorrelated ellipsoidal uncertainty case and in the two-scenario case, we devise fast combinatorial algorithms, where in the latter case we resort to solving a relaxation of the problem. In an extensive experimental evaluation we apply this approach to the robust versions of knapsack, shortest path, and minimum spanning tree problems. The results show that our approach clearly outperforms other solution methods in the uncorrelated ellipsoidal uncertainty case, as well as in the two-scenario case when the underlying problem is not given by a compact linear description.

1 Introduction

Most optimization problems arising in practice contain uncertain data. Possible sources of uncertainty are measurement errors or problem parameters that materialize in the future and cannot be predicted precisely. In the robust optimization approach, one searches for a solution that is feasible under all probable scenarios and optimizes the worst case over all these scenarios.

When dealing with uncertain combinatorial optimization problems, one often assumes that the set of feasible solutions is certain, so that the uncertainty affects only the cost coefficients in the objective function. In this paper we assume that a set $\mathcal{U}$ of potential objective functions is given. We thus consider problems of the form

$$\begin{aligned} &\min \max_{c \in \mathcal{U}} c^\top x \\ &\text{s.t. } x \in X \ , \end{aligned} \tag{R}$$

* The first author has been supported by the German Research Foundation (DFG) under grant BU 2313/2. The third author has been supported by the German Federal Ministry of Economics and Technology within the 6th Energy Research Programme

with $X \subseteq \{0,1\}^n$ defining the combinatorial structure of the underlying problem. Reasonable choices of the scenario set $\mathcal{U}$ depend on the application at hand, but also on the (theoretical and practical) tractability of the resulting problems. In the literature, three types of uncertainties have been discussed intensively: the case of a finite number of scenarios $\mathcal{U} = \{c_1, \ldots, c_k\}$, the case of interval uncertainties where each coefficient may vary independently within a certain range, and the case of ellipsoidal uncertainty, where the set $\mathcal{U}$ is an ellipsoid. The latter choice is motivated by the fact that, when assuming that the objective function coefficients are jointly normally distributed, the confidence regions form ellipsoids. Under the additional assumption that the distributions are independent, we obtain axis-parallel ellipsoids. We will refer to the latter case as uncorrelated ellipsoidal uncertainty.

The objective of this paper is to present a new generic approach to problems of type (R). Even if this approach can in principle be applied to a wide set of applications, we restrict ourselves to problems where linear optimization over X can be performed efficiently, and to the special cases of scenario-based and uncorrelated ellipsoidal uncertainty. Our motivation is two-fold: on the one hand, the min-max versions of classical combinatorial optimization problems that are easy in the deterministic variant can become *NP*-hard already for two scenarios and thus are challenging from a theoretical point of view [10]. On the other hand such problems are highly relevant in practice: in contrast to interval uncertainty, which is more commonly used and easier to deal with, the discrete scenario approach implicitly allows to model correlations in the objective function coefficients, while using ellipsoidal uncertainties can avoid too pessimistic solutions. In fact, the worst case scenario in Problem (R) always corresponds to an extreme point of $\mathcal{U}$, so that in the interval case all coefficients are at extreme points of their feasible ranges, which is very unlikely in practice. In the ellipsoidal case, this is explicitly excluded.

Discrete Scenario Case Robust combinatorial optimization as defined above has been studied extensively in the literature. For the discrete scenario case, Aissi et al. [1] develop fully polynomial-time approximation schemes (FPTAS) for the robust versions of the *minimum spanning tree problem* and the *shortest path problem* with a fixed number of scenarios. Both problems are *NP*-hard [17, 10]. In an in-depth study by Yu [16], the discrete scenario *knapsack problem* is shown to inherit the theoretical complexity of its deterministic variant, again assuming a fixed number of scenarios. For the same case, the author proposes a pseudo-polynomial algorithm as well as an exact algorithm based on surrogate relaxation. The only classical combinatorial optimization problem that is known to be tractable in the discrete scenario min-max-variant is the *minimum cut problem* with a fixed number of scenarios, as shown by Armon and Zwick [3]. However, the corresponding robust *minimum s-t-cut problem* is again *NP*-hard [3]. When allowing a finite but unbounded number of scenarios, most robust problems become strongly *NP*-hard. For a survey of the complexity of combinatorial optimization problems with discrete scenarios, see Aissi et al. [2].

Ellipsoidal Uncertainty Case Compared with the discrete scenario case, much less is known about ellipsoidal uncertainty. As one can easily reduce a two-scenario problem to a problem with ellipsoidal uncertainty, for any underlying set X, the latter class of problems is (weakly) *NP*-hard in general for most combinatorial optimization problems. In fact, as we will show later, this problem is strongly *NP*-hard even in the unconstrained case $X = \{0, 1\}$. However, the situation changes when assuming uncorrelated ellipsoidal uncertainty. Depending on the underlying combinatorial structure efficient algorithms may exist, as shown by Nikolova [12], who also proposes general-purpose approximation schemes. Moreover, she notes that the *minimum spanning tree problem with uncorrelated ellipsoidal uncertainty* can be solved in polynomial time. Otherwise the standard approach has been to address the problem by general mixed-integer quadratic programming methods. Atamtürk and Narayanan [4] propose an SOCP-based branch and bound-algorithm for the robust *knapsack problem* with uncorrelated ellipsoidal uncertainty that additionally exploits the submodularity of the objective function. As this approach only concerns the objective function, it can in principle be applied for any underlying set X.

Our Contribution In this paper, we develop a new exact approach for min-max versions of combinatorial optimization problems. We propose a branch and bound-algorithm using lower bounds obtained from Lagrangean decomposition, allowing to separate the uncertainty aspect in the objective function from the combinatorial structure of the feasible set. In particular, we present algorithms to solve Problem (R) for the unconstrained case $X = \{0, 1\}^n$, for the two classes of uncertainty sets mentioned above. An extended abstract of this paper will appear in the proceedings of ISCO 2014 [6]; the idea of using Lagrangean decomposition to solve mean-risk optimization problems was first discussed in [5]. In the current paper, besides giving the proofs omitted in [6], we extend our algorithm to the two-scenario case and evaluate it experimentally for a much larger number of different applications. Finally, we added a comparison to our own implementation of a polynomial time algorithm for the *minimum spanning tree problem with uncorrelated ellipsoidal uncertainty*.

This paper is organized as follows. In Section 2 we present the Lagrangean decomposition approach for robust combinatorial optimization problems and discuss two different types of uncertainty. For both types, we study the corresponding nonlinear unconstrained binary optimization problems arising in the decomposition. We devise algorithms to solve these problems that can deal with fixed variables, which allows us to embed the decomposition approach into a branch and bound-algorithm to compute provably optimal solutions. In Section 3 we apply our approach to six different problem types: the robust shortest path, knapsack and minimum spanning tree problems, all with two scenarios and with uncorrelated ellipsoidal uncertainty. Extensive experimental studies show that our algorithm outperforms the approaches described in the literature so far.

2 A Lagrangean Decomposition Approach

Lagrangean decomposition can be considered a special case of Lagrangean relaxation, applied to a set of artificial constraints. Its aim is to decompose a problem into auxiliary problems that can be easily computed. We use Lagrangean decomposition to separate the nonlinear objective function from the combinatorial constraints. Starting from the problem

$$\begin{array}{l} \min\ f(x) \\ \text{s.t. } x \in X \subseteq \{0,1\}^n\,, \end{array} \tag{P}$$

we introduce new variables $y \in \mathbb{R}^n$ along with artificial linking constraints and express the original set of combinatorial constraints in the new variables:

$$\begin{array}{l} \min\ f(x) \\ \text{s.t. } x = y \\ \qquad x \in \{0,1\}^n \\ \qquad y \in X\,. \end{array}$$

Lagrangean relaxation of the linking equations yields

$$\begin{array}{l} \min\ f(x) + \lambda^\top (y - x) \\ \text{s.t. } x \in \{0,1\}^n \\ \qquad y \in X\,, \end{array}$$

where $\lambda \in \mathbb{R}^n$ is the vector of Lagrangean multipliers. Since the original objective function and the set of constraints are now independent of each other, the problem decomposes into

$$\begin{array}{l} \min\ f(x) - \lambda^\top x \\ \text{s.t. } x \in \{0,1\}^n \end{array} \quad + \quad \begin{array}{l} \min\ \lambda^\top y \\ \text{s.t. } y \in X\,. \end{array} \tag{$LD(\lambda)$}$$

The two minimization problems in ($LD(\lambda)$) can be solved independently. The left-hand side problem is an unconstrained nonlinear minimization problem over binary variables, whereas the right-hand side problem is a linear instance of the underlying combinatorial problem. For any $\lambda \in \mathbb{R}^n$, ($LD(\lambda)$) is a relaxation of the original problem (P) and yields a lower bound on its optimal value. The best possible bound is obtained by computing the Lagrangean dual

$$\max_{\lambda \in \mathbb{R}^n}\ LD(\lambda)\,, \tag{1}$$

for example with a subgradient algorithm. In each iteration the two subproblems of ($LD(\bar{\lambda})$) have to be solved for a given $\bar{\lambda}$. Moreover, it is easy to verify that a supergradient of LD in a given point $\bar{\lambda}$ can be computed as $\bar{y} - \bar{x}$, where $\bar{x}$ and $\bar{y}$ are optimal solutions of the two minimization problems in ($LD(\bar{\lambda})$).

Note that

$$LD(\lambda) \quad = \quad \begin{cases} \min\ z - \lambda^\top x \\ \text{s.t. } (z,x) \in \operatorname{conv}(F) \end{cases} \quad + \quad \begin{array}{l} \min\ \lambda^\top y \\ \text{s.t. } y \in \operatorname{conv}(X) \end{array}$$

where $F := \{(z,x) \mid x \in \{0,1\}^n,\ z \geq f(x)\}$. By general results on Lagrangean relaxation we obtain

Lemma 1.

$$\max_{\lambda \in \mathbb{R}^n} LD(\lambda) \quad = \quad \begin{cases} \min z \\ s.t.\ (z,x) \in \operatorname{conv}(F) \\ \qquad x \in \operatorname{conv}(X)\ . \end{cases}$$

This observation will be generalized by Lemma 2 below. Note that

$$\begin{array}{l} \min z \\ \text{s.t. } (z,x) \in \operatorname{conv}(F) \\ \qquad x \in \operatorname{conv}(X) \end{array} \quad \geq \quad \begin{array}{l} \min f(x) \\ \text{s.t. } x \in \operatorname{conv}(\{0,1\}^n) \\ \qquad x \in \operatorname{conv}(X) \end{array} \quad = \quad \begin{array}{l} \min f(x) \\ \text{s.t. } x \in \operatorname{conv}(X)\ , \end{array}$$

and that the inequality is strict in general if f is nonlinear. This is due to the fact that the objective function f is minimized over $\{0,1\}^n$ in the left-hand side problem of ($LD(\lambda)$) instead of over $[0,1]^n$. In other words, the bounds we obtain are potentially stronger than those obtained from convexifying the feasible set in Problem (P).

Instead of solving the decomposition ($LD(\lambda)$) exactly, it is possible to solve relaxations of the subproblems. This might be advantageous when the subproblems are computationally hard or no exact solution algorithm is known. Even if this approach decreases the quality of the resulting lower bounds and hence increases the number of nodes in the branch and bound-algorithm, this effect might be compensated by the shorter time required to compute the bounds.

When embedding the computation of the Lagrangean dual into a branch and bound-algorithm in order to obtain exact solutions, the problem solved in the root node of the branch and bound-tree is (1), but in deeper levels of the tree variable fixings have to be respected. This means that the algorithms for both the (formerly) unconstrained nonlinear subproblem and the linear combinatorial subproblem have to be adapted to handle fixed variables. In most cases this is straight-forward. E.g., Dantzig's Algorithm for the *fractional knapsack problem* [7] can be easily extended by a preprocessing step such that in the later stages only free variables have to be considered. For other applications, like the *robust shortest path problem*, adapting combinatorial algorithms is more complicated. Note however that, whenever the combinatorial subproblem is solved by linear programming, fixed variables can be easily incorporated.

Within a branch and bound-scheme, our approach can be improved significantly by reoptimization: in order to compute the Lagrangean dual (1) quickly, a good starting guess for the multipliers λ is crucial. The choice of the initial multipliers in the root node should depend on the objective function, i.e. on the type of uncertainty set considered. In the remaining nodes of the branch and bound-tree, we use the optimal multipliers of the parent node to warmstart the subgradient algorithm.

An important advantage of the Lagrangean decomposition approach is that we get a primal heuristic for free: each time we solve ($LD(\lambda)$) we obtain a feasible

solution $y \in X$ for Problem (R). In particular, we can use $f(y)$ as an upper bound in our algorithm.

2.1 General Case of Robust Optimization

In robust optimization, the function f is defined as the worst case solution quality of a vector $x \in X$ over all scenarios c in a given set $\mathcal{U}$. More formally, we consider objective functions of the form

$$f(x) := \max_{c \in \mathcal{U}} c^\top x$$

where $\mathcal{U} \subset \mathbb{R}^n$ is a compact set.

In many applications, linear optimization over the feasible set X is only possible (or at least easier) if the objective function satisfies certain additional conditions such as, e.g., nonnegativity or triangle inequalities. In the following, we argue that our approach also works in this case. More precisely, we show that any homogeneous inequality that is valid for all scenarios $c \in \mathcal{U}$ can be assumed to be valid also for each vector λ defining the objective function of the combinatorial problem appearing in the right-hand side of ($LD(\lambda)$). To this end, we show

Theorem 1. *Let* $\operatorname{cone}(\mathcal{U})$ *denote the closed convex cone generated by* $\mathcal{U}$ *and let* $\operatorname{cone}(\mathcal{U})^*$ *be its dual cone. Then*

$$\min \max_{c \in \mathcal{U}} c^\top x \quad \text{s.t. } x \in X \quad = \quad \begin{array}{ll} \min & \max_{c \in \mathcal{U}} c^\top x \\ \text{s.t.} & x \in \{0,1\}^n \\ & y \in X \\ & x - y \in \operatorname{cone}(\mathcal{U})^* \, . \end{array}$$

Proof. By definition, we have $x - y \in \operatorname{cone}(\mathcal{U})^*$ if and only if $d^\top (x - y) \geq 0$ for all $d \in \mathcal{U}$. For every feasible solution (x, y) of the problem

$$\begin{array}{ll} \min & \max_{c \in \mathcal{U}} c^\top x \\ \text{s.t.} & x \in \{0,1\}^n \\ & y \in X \\ & x - y \in \operatorname{cone}(\mathcal{U})^* \end{array}$$

we thus have $d^\top x \geq d^\top y$ for all $d \in \mathcal{U}$ and hence $\max_{c \in \mathcal{U}} c^\top x \geq \max_{c \in \mathcal{U}} c^\top y$. Since we minimize the former expression, we may even assume equality if (x, y) is an optimal solution, hence we may replace $\max_{c \in \mathcal{U}} c^\top x$ by $\max_{c \in \mathcal{U}} c^\top y$, obtaining an equivalent problem

$$\begin{array}{ll} \min & \max_{c \in \mathcal{U}} c^\top y \\ \text{s.t.} & x \in \{0,1\}^n \\ & y \in X \\ & x - y \in \operatorname{cone}(\mathcal{U})^* \, . \end{array}$$

As $X \subseteq \{0,1\}^n$, the latter problem can be solved by first choosing $y \in X$ that minimizes $\max_{c\in\mathcal{U}} c^\top y$ and then setting $x := y$. This shows that the problem is equivalent to

$$\begin{aligned} \min \max_{c\in\mathcal{U}} & c^\top x \\ \text{s.t. } & x \in X \ . \end{aligned}$$

□

We can thus apply the Lagrangean relaxation approach directly to the problem

$$\begin{aligned} \min \max_{c\in\mathcal{U}} & c^\top x \\ \text{s.t.} \quad & x \in \{0,1\}^n \\ & y \in X \\ & x - y \in \text{cone}(\mathcal{U})^* \ , \end{aligned}$$

meaning that the dual multipliers have to be chosen from $\text{cone}(\mathcal{U})$. It remains to investigate whether this restriction on λ yields the same bound as (1).

Lemma 2. *Let $\mathcal{C} \subseteq \mathbb{R}^n$ be any closed convex cone and assume that the problem*

$$\begin{aligned} \min \ & z \\ \textit{s.t. } & (z,x) \in \text{conv}(F) \\ & y \in \text{conv}(X) \\ & x - y \in \mathcal{C}^* \end{aligned} \tag{2}$$

satisfies Slater's condition. Then (2) agrees with $\max_{\lambda\in\mathcal{C}} LD(\lambda)$.

Proof. By construction, $\max_{\lambda\in\mathcal{C}} LD(\lambda)$ is the partial Lagrangean dual of (2) with respect to the last constraint. As this problem is convex, strong duality follows from Slater's condition. □

Note that Lemma 2 implies Lemma 1 when applied to the cone $\mathcal{C} = \mathbb{R}^n$ for which $\mathcal{C}^* = \{0\}$. The following result finally shows that, under mild conditions, we may assume $\lambda \in \text{cone}(\mathcal{U})$ without weakening our Lagrangean bound.

Theorem 2. *Assume that (2) satisfies Slater's condition for $\mathcal{C} = \text{cone}(\mathcal{U})$. Then*

$$\max_{\lambda\in\text{cone}(\mathcal{U})} LD(\lambda) \quad = \quad \max_{\lambda\in\mathbb{R}^n} LD(\lambda) \ .$$

Proof. Analogously to the proof of Theorem 1, one can show that

$$\begin{aligned} \min \ & z \\ \text{s.t. } & (z,x) \in \text{conv}(F) \\ & y \in \text{conv}(X) \\ & x - y \in \text{cone}(\mathcal{U})^* \end{aligned} \quad = \quad \begin{aligned} \min \ & z \\ \text{s.t. } & (z,x) \in \text{conv}(F) \\ & y \in \text{conv}(X) \\ & x - y = 0 \ . \end{aligned}$$

Applying Lemma 2 to both sides, the result follows. □

In the case of a polyhedral cone, we obtain

Corollary 1. *Let $A \in \mathbb{R}^{m\times n}$ such that $Ac \leq 0$ for all $c \in \mathcal{U}$. Then*

$$\max_{A\lambda\leq 0} \; LD(\lambda) \quad = \quad \max_{\lambda\in\mathbb{R}^n} \; LD(\lambda)\,.$$

Proof. If $\mathcal{C}$ is described by finitely many linear inequalities, problem (2) satisfies Slater's condition. □

In particular, this shows that any finite number of conditions on the objective function of one of the following types can be carried over from $\mathcal{U}$ to λ:

- nonnegativity or nonpositivity of a given objective function coefficient;
- the triangle inequality on a given triple of coefficients;
- if variables correspond to edges of a graph, the nonnegativity of the total cost of a given cycle.

If cone($\mathcal{U}$) is not polyhedral, the assumption of Theorem 2 might still be satisfied. E.g., assume that the condition $y \in X$ fixes no variable y_i, i.e. that $y \in X$ does not imply $y_i = 0$ nor $y_i = 1$ for any i. Choose a vector $y_0 \in \text{relint}(\text{conv}(X))$. By our assumption, we have $y_0 \in (0,1)^n$. Now let $z \in \text{relint}(\mathcal{C}^*)$ and $\varepsilon > 0$ with $x_0 := y_0 + \varepsilon z \in (0,1)^n$. Then $x_0 - y_0 \in \text{relint}(\mathcal{C}^*)$, so that Slater's condition is satisfied and strong duality holds. In particular, eliminating all fixed variables from the problem formulation, we can always apply Theorem 2.

The results of the previous section are valid for general uncertainty sets $\mathcal{U}$. Since the effectiveness of the decomposition approach depends largely on the efficient computation of the two subproblems in ($LD(\lambda)$), we now study the nonlinear unconstrained subproblem in more detail.

2.2 Discrete Scenario Case

We first consider finite sets of scenarios $\mathcal{U} = \{c_1, \ldots, c_k\}$ for a given $k \in \mathbb{N}$. In this case the unconstrained robust optimization problem can be written as

$$\begin{array}{l} \min \max\{c_1^\top x, \ldots, c_k^\top x\} \\ \text{s.t. } x \in \{0,1\}^n\,. \end{array} \tag{3}$$

Complexity of the Unconstrained Problem

We first prove that already the unconstrained problem with two scenarios,

$$\min_{x\in\{0,1\}^n} \max\{c_1^\top x, c_2^\top x\} \tag{4}$$

with $c_1, c_2 \in \mathbb{R}^n$, is *NP*-hard by reducing the subset sum problem to it. The latter is known to be *NP*-complete [8], it is defined as follows:

Subset sum problem. Given numbers $s_1, \ldots, s_n \in \mathbb{Z}$ and $S \in \mathbb{Z}$, is there a subset $I \subseteq \{1, \ldots, n\}$ such that the sum $\sum_{i\in I} s_i$ is exactly S?

Theorem 1 *The unconstrained binary two-scenario optimization problem* (4) *is* NP*-hard.*

Proof. We describe a polynomial reduction of the *subset sum problem* to (4). Given a subset sum instance $(s, S) \in \mathbb{Z}^{n+1}$ of size n, we may assume that $S \geq 0$ and construct an instance of (4) of dimension $n+1$ by setting $c_1 = (s, -2S)$ and $c_2 = (-s, 0)$. We then have

$$\begin{aligned}
\min_{z \in \{0,1\}^{n+1}} \max\{c_1^\top z, c_2^\top z\} &= \min_{\substack{x \in \{0,1\}^n \\ y \in \{0,1\}}} \max\{s^\top x - 2Sy, -s^\top x\} \\
&= \min_{x \in \{0,1\}^n} \max\{s^\top x - 2S, -s^\top x\} \quad (\text{as } S \geq 0) \\
&= -S + \min_{x \in \{0,1\}^n} \max\{s^\top x - S, -s^\top x + S\} \\
&= -S + \min_{x \in \{0,1\}^n} |s^\top x - S| \,.
\end{aligned}$$

It follows that there exists a set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} s_i = S$ if and only if the optimal value of the constructed instance of (4) is $-S$. □

An Exact Algorithm for the Unconstrained Two-Scenario Problem

In the following we devise a pseudo-polynomial algorithm for Problem (4) using a reduction to a pair of *knapsack problems*. For two functions $f, g \colon \{0,1\}^n \longrightarrow \mathbb{R}$ the function $h(x) = \max\{f(x), g(x)\}$ can be written as

$$h(x) = \begin{cases} f(x) \text{ if } f(x) \geq g(x) \\ g(x) \text{ otherwise.} \end{cases}$$

For Problem (4), slightly generalized to allow for constant terms in the two objective functions, we thus obtain

$$\begin{aligned}
&\min_{x \in \{0,1\}^n} \max\{a^\top x + a_0, b^\top x + b_0\} \\
&\qquad = \min \left\{ \begin{array}{c} \min a^\top x_1 + a_0 \\ \text{s.t. } a^\top x_1 + a_0 \geq b^\top x_1 + b_0 \\ x_1 \in \{0,1\}^n \end{array} \;,\; \begin{array}{c} \min b^\top x_2 + b_0 \\ \text{s.t. } b^\top x_2 + b_0 \geq a^\top x_2 + a_0 \\ x_2 \in \{0,1\}^n \end{array} \right\} \\
&\qquad = -\max \left\{ \begin{array}{c} \max -a^\top x_1 - a_0 \\ \text{s.t. } (b-a)^\top x_1 \leq a_0 - b_0 \\ x_1 \in \{0,1\}^n \end{array} \;,\; \begin{array}{c} \max -b^\top x_2 - b_0 \\ \text{s.t. } (a-b)^\top x_2 \leq b_0 - a_0 \\ x_2 \in \{0,1\}^n \end{array} \right\}
\end{aligned}$$

Both subproblems are integer linear optimization problems with a single constraint. Replacing variables with negative weight by their complement and then eliminating variables with negative objective function coefficient, these subproblems can be reduced to *knapsack problems*. If the original coefficients in Problem (4) are integer, the same holds for both weights and profits of the resulting knapsack problems. Moreover, as the reduction is polynomial, the well-known

dynamic programming approach for the knapsack problem yields a pseudo-polynomial algorithm for Problem (4).

However, in our Lagrangean decomposition approach the coefficients a and b include the Lagrangean multipliers λ, which are not integer in general. Instead of solving the two knapsack subproblems exactly, we thus propose to either solve their continuous relaxations using Dantzig's algorithm [7] or to use the FPTAS proposed by Ibarra and Kim [9] for solving them approximately. Both approaches yield weaker dual bounds in general, but these can be computed quickly. In our experiments we follow the former approach.

When the above formulation of the unconstrained two-scenario problem is embedded into a branch and bound-algorithm, the algorithm used to solve the two *knapsack problems* (or their continuous relaxations) must handle fixed variables caused by branching decisions. This can be easily achieved by adding a preprocessing step to the algorithm, which just leads to a dimension reduction and additional constant terms in the objective functions. The resulting problems are thus again *knapsack problems* and can be solved by the original algorithm.

In this paper, we restrict ourselves to the two scenario case. One can show that Problem (3) admits a pseudo-polynomial algorithm as long as the number k of scenarios is considered a constant, e.g., by adapting the pseudo-polynomial algorithm for the k-scenario knapsack problem devised by Yu [16]. However, our approach to compute a dual bound quickly by solving the continuous relaxation of two knapsack problems, using Dantzig's algorithm, cannot easily be generalized: we would obtain k problems having $k-1$ general constraints each, which cannot be turned into knapsack constraints simultaneously. Each of the corresponding relaxations could be solved efficiently as a linear program, but we are not aware of any fast combinatorial algorithm.

Using the results of Section 2.1, the computation of the optimal Lagrangean multipliers λ can be improved significantly. Indeed, by Theorem 2, it suffices to optimize λ over $\operatorname{cone}(\mathcal{U})$. As $\mathcal{U} = \{c_1, \dots, c_k\}$, we can write $\lambda = \sum_{i=1}^{k} \mu_i c_i$ with $\mu \geq 0$ and optimize over $\mu \in \mathbb{R}^k$ instead of $\lambda \in \mathbb{R}^n$. The dimension of the dual problem (1) now reduces to the number of scenarios k, which is usually much smaller than the number of variables n.

A Mixed-Integer Linear Formulation

As an alternative to the Lagrangean decomposition approach, the nonlinear model (3) for the discrete scenario case can be easily linearized by introducing a single additional variable and k constraints:

$$\begin{aligned} &\min\ y \\ &\text{s.t. } y \geq c_i^\top x \quad \text{for all } i \in \{1, \dots, k\} \\ &\qquad x \in X\,. \end{aligned} \tag{5}$$

If a polyhedral description of the set X is known, model (5) can be solved as a mixed-integer linear program by, e.g., a branch and cut-algorithm. Linearization is thus an alternative to our Lagrangean decomposition approach in the discrete

scenario case, see our experimental evaluation in Section 3. However, many combinatorial optimization problems require a separation procedure when modeled as linear programs, while at the same time fast combinatorial algorithms exist. This is the case, e.g., for the *minimum spanning tree problem*, where an exponential number of cycle elimination constraints is needed in the standard integer programming model.

2.3 Uncorrelated Ellipsoidal Uncertainty Case

We next consider the case of ellipsoidal uncertainty. The set $\mathcal{U}$ of all possible scenarios then has the form of an ellipsoid in $\mathbb{R}^n$,

$$\mathcal{U} = \left\{ c \in \mathbb{R}^n \middle| \, (c - c_0)^\top A^{-1} (c - c_0) \leq 1 \right\},$$

with $c_0 \in \mathbb{R}^n$ denoting the center of the ellipsoid and $A \in \mathbb{R}^{n \times n}$ being a positive definite symmetric matrix. In this case, different from the two-scenario case, the objective function

$$f(x) = \max_{c \in \mathcal{U}} c^\top x$$

of (P) can be replaced by a closed formula: for a given $x \in \mathbb{R}^n$, the value $f(x)$ is obtained by a linear maximization over an ellipsoid, yielding

$$f(x) = c_0^\top x + \sqrt{x^\top A x} \, .$$

Thereby the unconstrained min-max problem arising in the left-hand part of problem ($LD(\lambda)$) in the ellipsoidal uncertainty case reads

$$\min_{x \in \{0,1\}^n} (c_0 - \lambda)^\top x + \sqrt{x^\top A x} \, . \tag{6}$$

Here, c_0 and A can be interpreted as the expected values and the covariance matrix of a set of random variables.

In the following, we restrict ourselves to the case of uncorrelated random variables. In this case, the ellipsoid $\mathcal{U}$ is axis-parallel or, equivalently, the matrix A is diagonal. Exploiting the binarity of x we can simplify Problem (6) to

$$\min_{x \in \{0,1\}^n} (c_0 - \lambda)^\top x + \sqrt{a^\top x} \, , \tag{7}$$

where $A = \mathrm{Diag}(a)$ for some positive vector $a \in \mathbb{R}^n$.

An Efficient Algorithm for the Unconstrained Problem

Problem (7) is an unconstrained variant of the so-called *mean-risk optimization problem*. It can be solved to optimality in polynomial time since the objective function is submodular [4]. As minimization algorithms for general submodular functions are too slow to be applied in practice, we aim at a faster algorithm exploiting the special structure of Problem (7).

To this end, consider two solutions of (7) which differ in exactly one variable i. The difference between the corresponding objective values is

$$\Delta_i f(J) = (c_0 - \lambda)_i + \sqrt{\sum_{j \in J} a_j + a_i} - \sqrt{\sum_{j \in J} a_j}, \tag{8}$$

with J denoting the set of variables which are 1 in both solutions. The value (8) is also known as the discrete derivative of variable i [15], it describes the contribution of setting variable i to 1, which clearly depends on the set J or, more precisely, on the quantity $\sum_{j \in J} a_j$. We hence define for each variable i its *contribution function* by

$$C_i(z) = (c_0 - \lambda)_i + \sqrt{z + a_i} - \sqrt{z}\ ,$$

see Figure 1 for an illustration. The functions C_i are strictly decreasing and therefore have at most one root each. The root r_i of C_i indicates the value which $\sum_{j \in J} a_j$ must reach such that setting variable i to 1 becomes profitable. Note that setting a variable to 1 never has a negative effect on the contributions of other variables, due to submodularity of the objective function of (7).

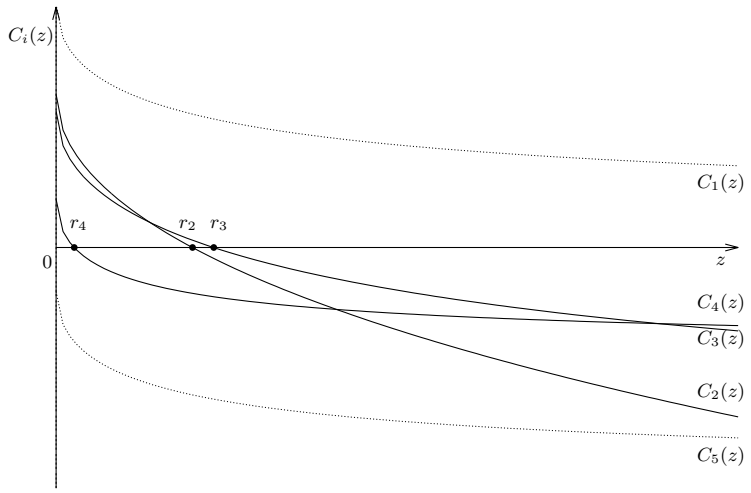

Fig. 1. An illustration of contribution functions; the dashed functions have no roots.

Our basic idea for the construction of an optimal solution of (7) is that, due to the definition of r_i, a variable i cannot be 1 in the optimal solution while another variable having a smaller root is 0. This leads to an obvious sorting algorithm. However, in a first step we have to eliminate variables i for which C_i has no root, using the following lemma.

Lemma 1 *There exists an optimal solution x^* of Problem (7) with the following properties:*

(i) if $(c_0 - \lambda)_i \geq 0$, then $x_i^* = 0$.
(ii) if $(c_0 - \lambda)_i \leq -\sqrt{a_i}$, then $x_i^* = 1$.

Proof. The condition in (i) implies that the function C_i is positive everywhere, as $a_i > 0$. This implies that any solution with $x_i = 1$ can be improved by setting $x_i = 0$. The condition in (ii) implies that C_i is non-positive everywhere, as

$$(c_0 - \lambda)_i + \sqrt{z + a_i} - \sqrt{z} \leq (c_0 - \lambda)_i + \sqrt{a_i} \leq 0$$

by the concavity of the square root function. The contribution of variable i to the value of an arbitrary solution is therefore non-positive, so that it may be fixed to 1 without loss of generality. □

The two cases of Lemma 1 are illustrated by the dashed function graphs in Figure 1. The functions C_i for the remaining variables, i.e. of those with $0 > (c_0 - \lambda)_i > -\sqrt{a_i}$, have exactly one positive root each. The entire algorithm is stated in Algorithm 1.

Theorem 3. *Algorithm 1 solves Problem (7) in time $O(n \log n)$.*

Proof. The algorithm runs in linear time except for the sorting of variables which takes $O(n \log n)$ time. It thus remains to prove the correctness of Algorithm 1. By Lemma 1 we may assume $k = n$, then it suffices to show that (after sorting) every optimal solution x^* satisfies $x_1^* \geq x_2^* \geq \cdots \geq x_n^*$.

Assume on contrary that x^* is an optimal solution with $x_j^* = 0$ and $x_{j+1}^* = 1$ for some $j < n$. Consider the two solutions x^0 and x^1 defined by

$$x_i^0 = \begin{cases} 0 & \text{for } i = j+1 \\ x_i^* & \text{otherwise,} \end{cases} \qquad x_i^1 = \begin{cases} 1 & \text{for } i = j \\ x_i^* & \text{otherwise.} \end{cases}$$

By optimality of x^* we have

$$0 \geq f(x^*) - f(x^0) = C_{j+1}\big(\textstyle\sum_{i \in I} a_i\big)$$

for $I = \{i \in \{1, ..., n\} \setminus \{j+1\} \mid x_i^* = 1\}$ and hence by definition of r_{j+1} and r_j

$$\sum_{i \in I} a_i \geq r_{j+1} \geq r_j \,. \tag{9}$$

Then, using the concavity of the square-root function we have

$$\begin{aligned} f(x^1) - f(x^*) &= (c_0 - \lambda)_j + \sqrt{\sum_{i \in I} a_i + a_{j+1} + a_j} - \sqrt{\sum_{i \in I} a_i + a_{j+1}} \\ &< (c_0 - \lambda)_j + \sqrt{\sum_{i \in I} a_i + a_j} - \sqrt{\sum_{i \in I} a_i} \\ &\overset{(9)}{\leq} (c_0 - \lambda)_j + \sqrt{r_j + a_j} - \sqrt{r_j} = 0 \end{aligned}$$

which contradicts the optimality of x^*. □

Algorithm 1

Input: $c_0, \lambda, a \in \mathbb{R}^n$
Output: vector $x^* \in \{0,1\}^n$ minimizing $(c_0 - \lambda)^\top x + \sqrt{a^\top x}$

$k = 0$, $A = 0$, $C = 0$
for $i \in \{1, ..., n\}$ **do**
  if $(c_0 - \lambda)_i \geq 0$ **then**
    $x_i^* = 0$
  else if $(c_0 - \lambda)_i \leq -\sqrt{a_i}$ **then**
    $x_i^* = 1$
    $A = A + a_i$, $C = C + (c_0 - \lambda)_i$
  else
    $r_i \leftarrow \left(\frac{a_i - (c_0 - \lambda)_i^2}{2(c_0 - \lambda)_i}\right)^2$
    $k \leftarrow k + 1$
  end if
end for

sort the variables such that $x_1, \ldots, x_k$ are unfixed with $r_1 \leq r_2 \leq \cdots \leq r_k$

$f^* = C + \sqrt{A}$, $i^* = 0$
for $i \in \{1, ..., k\}$ **do**
  $A = A + a_i$, $C = C + (c_0 - \lambda)_i$
  if $C + \sqrt{A} < f^*$ **then**
    $f^* = C + \sqrt{A}$, $i^* = i$
  end if
end for

for $i \in \{1, ..., i^*\}$ **do**
  $x_i^* = 1$
end for
for $i \in \{i^* + 1, ..., k\}$ **do**
  $x_i^* = 0$
end for

return x^*

Note that a similar algorithm for Problem (7) using a different sorting rule has been devised by Shen et al. [14].

The fixings arising in the branch and bound-scheme for the min-max problem do not affect this strategy. The roots can be computed using the same formula, because an additional constant in the square root term does not affect the order of the roots. It merely causes a shift of the roots to the left by the same amount for each variable. The correctness of the algorithm in the presence of fixings is easily verified.

To conclude this Section, we show that the unconstrained binary problem (6) for general A is *NP*-hard in the strong sense, so that we cannot even expect to find a pseudo-polynomial algorithm to solve it in general.

Theorem 4. *Problem* (6) *is strongly* NP*-hard.*

Proof. We use the well known fact that binary quadratic programming is strongly *NP*-hard, as can be shown by its equivalence to the maximum cut problem. It thus suffices to describe a polynomial reduction from a problem of the form

$$\min_{x\in\{0,1\}^n} \frac{1}{2}x^\top Qx + L^\top x \;, \tag{10}$$

where $Q \in \mathbb{Z}^{n\times n}$ is any symmetric matrix and $L \in \mathbb{Z}^n$, to Problem (6) with data of polynomial size. First, compute

$$\lambda := \min_{i=1,\dots,n} \left(|Q_{ii}| - \sum_{j\neq i} |Q_{ij}| \right) \in \mathbb{Z} \,.$$

Setting $\bar{Q} := Q - 2(\lambda-1)I \in \mathbb{Z}^{n\times n}$ and $\bar{L} := L + (\lambda-1)e \in \mathbb{Z}^n$, we have

$$\frac{1}{2}x^\top Qx + L^\top x = \frac{1}{2}x^\top \bar{Q}x + \bar{L}^\top x \text{ for all } x \in \{0,1\}^n \,.$$

By construction, the matrix $\bar{Q} - I$ is diagonally dominant, so that $\bar{Q} - I \succeq 0$ and $\bar{Q} \succ 0$. Next, define $c := \bar{L}^\top \bar{L} + 1 \in \mathbb{Z}$. Then the matrix

$$A := \begin{pmatrix} \bar{Q} & \bar{L} \\ \bar{L}^\top & c \end{pmatrix}$$

is integer and each entry has polynomial size in the entries of Q and L. The Schur complement shows that A is positive definite, as

$$c - \bar{L}^\top \bar{Q}^{-1}\bar{L} = 1 + \bar{L}^\top (I - \bar{Q}^{-1})\bar{L} \geq 1$$

due to $I - \bar{Q}^{-1} \succeq 0$. Now (10) agrees with

$$\begin{aligned} -\tfrac{1}{2}c + \min\ & \tfrac{1}{2}y^\top Ay \\ \text{s.t.}\quad & y \in \{0,1\}^{n+1} \\ & y_{n+1} = 1 \,, \end{aligned}$$

which can be reduced to solving

$$\begin{aligned} \min\ & \sqrt{y^\top A y} \\ \text{s.t.}\ & y \in \{0,1\}^{n+1} \\ & y_{n+1} = 1\,. \end{aligned}$$

Setting $M := \sum_{ij} A_{ij} + 1 \in \mathbb{Z}$, the latter can be rewritten as

$$\begin{aligned} M + \min\ & (-M) y_{n+1} + \sqrt{y^\top A y} \\ \text{s.t.}\ & y \in \{0,1\}^{n+1}\,, \end{aligned}$$

since $y^\top A y \in \{0, \dots, M-1\}$ and hence $0 \le \sqrt{y^\top A y} \le \sqrt{M-1} \le M-1$ for all $y \in \{0,1\}^{n+1}$. The latter problem is of the form (6). □

A Mixed-Integer SOCP Formulation

Due to the non-linearity of the set $\mathcal{U}$, there is no straight-forward linear mixed-integer formulation of Problem (R) in the ellipsoidal uncertainty case. However, it is well known that the problem can be modeled as a mixed-integer second-order cone program (SOCP), even in the correlated case: the objective function in (6) can be modeled by an SOCP constraint, while the feasible set X has to be modeled by a polyhedral description of $\mathrm{conv}(X)$ as in the discrete scenario case. In practice, mixed-integer SOCPs are much harder to solve than mixed-integer linear programs, making this approach much less competitive than the linearization approach in the discrete scenario case. We could also observe this in our experimental evaluation presented in the following section.

3 Applications

The Lagrangean decomposition approach presented in Section 2 is applicable to a wide range of robust combinatorial optimization problems. In the following we present numerical results for the *robust knapsack problem*, the *robust shortest path problem*, and the *robust spanning tree problem*. We compare the performance of the decomposition algorithm with the standard mixed-integer approaches: for the two-scenario problems, we solve the mixed-integer linear program discussed in Section 2.2; in case of uncorrelated ellipsoidal uncertainty, we consider a mixed-integer SOCP model as explained in Section 2.3. In both cases, we use CPLEX 12.6 to solve the resulting programs.

The left subproblems of the decomposition ($LD(\lambda)$), i.e. the unconstrained nonlinear binary minimization problems, were solved with the algorithms we discussed in Sections 2.2 (using Dantzig's algorithm to solve the relaxations of the two resulting knapsack problems) and 2.3. In the two-scenario case the initial multipliers were component-wise chosen as the smaller of the two scenarios, in the ellipsoidal uncertainty case as the center of the ellipsoid.

All experiments were carried out on a machine running SUSE Linux on an Intel Xeon E5 CPU at 2.60 GHz. All running times are stated in CPU-seconds; the time limit for each instance was one CPU-hour.

3.1 The Two-Scenario Knapsack Problem

In the *knapsack problem*, the task is to choose a subset from a set of objects. Each object is characterized by a profit and a weight. The subset has to be selected such that the sum of the profits of the chosen elements is maximum and the sum of the weights does not exceed a given threshold. In the two-scenario variant of the *knapsack problem* each element is associated with one profit for each scenario. The aim here is to maximize the minimum profit the two scenarios yield. The corresponding IP model is

$$\begin{aligned} \max\ & \min\{c_1^\top x, c_2^\top x\} \\ \text{s.t. } & w^\top x \leq b \\ & x \in \{0,1\}^n\ , \end{aligned} \tag{11}$$

where c_1 and c_2 are the profit vectors for the two scenarios, w is the weight vector and b the weight threshold. The solution of the above model (11) can be equivalently computed as

$$\begin{aligned} -\min\ & \max\{-c_1^\top x, -c_2^\top x\} \\ \text{s.t. } & w^\top x \leq b \\ & x \in \{0,1\}^n\ . \end{aligned} \tag{12}$$

The two-scenario *knapsack problem* was shown to be *NP*-hard by Yu [16], who also proposed a pseudo-polynomial algorithm. The decomposition algorithm can be directly applied to formulation (12). The combinatorial subproblem is a linear *knapsack problem*. The binary constraint on the y-variables in this subproblem can be relaxed to $0 \leq y \leq 1$, which is equivalent to solving a *fractional knapsack problem* instead of a *binary knapsack problem*. The obvious advantage of solving the relaxed problem is that the Lagrangean dual of (12) can be computed in polynomial time, while solving the binary knapsack problem would take pseudo-polynomial time. The disadvantage is that the lower bound on the objective value of (12) obtained from the Lagrangean dual is weaker in general. In our experiments we solved the *fractional knapsack problem* with an adapted variant of Dantzig's algorithm that can deal with fixed variables.

We used a set of randomly generated instances. The two scenarios were determined by first generating a nominal scenario with profits chosen randomly and uniformly distributed from the interval $[0, 100]$. The two scenarios were then generated by, for each coefficient, randomly choosing a deviation of up to $dev\%$ in both directions from the nominal value, with $dev \in \{1, 5, 10, 50, 100\}$. The (certain) weights w used in the constraint were chosen randomly and uniformly distributed from $[0, 100]$, while the threshold b was determined as $\frac{1}{2}\sum_{i=1}^{n} w_i$. This choice of the right-hand side was proposed in [4] to exclude trivial instances. Instance sizes range from 600 to 1400 elements, and 10 instances of each size and type were created.

The left part of Table 1 shows our results for the *two-scenario knapsack problem*. The bold numbers indicate the instance sizes. The columns show the maximum allowed deviation from the nominal scenario in percent, the number of instances that could be solved within one hour, the average number of

subproblems in the branch and bound-tree, the average number of times the subproblems in the decomposition ($LD(\lambda)$) were solved and the average running time in seconds.

All but two instances could be solved within the time limit of one hour. Instances with highly similar scenarios seem to be easiest to solve by our approach. For smaller n the instances with the largest deviation are hardest. With growing instance size this shifts towards lower deviations. Note that the number of iterations in the determination of the Lagrangean bound is relatively small compared to the number of nodes in the branch and bound-tree: in most cases, Problem ($LD(\lambda)$) has to be solved between two and five times per node.

Although it can solve very large instances to optimality, the decomposition approach for the *two-scenario knapsack problem* is not competitive with the linearization approach discussed in Section 2.2. Using the CPLEX 12.6 MIP solver, all instances could be solved to optimality in less than one second each. This is due to the fact that the linearized model is very compact in the knapsack case, it contains only three non-trivial constraints.

3.2 The Two-Scenario Shortest Path Problem

The *shortest path problem* asks for a path from a given node s to a given node t in a directed network $G = (V, A)$, such that the sum of the arc costs is minimal. In the two-scenario variant of the *shortest path problem* each arc is associated with two cost values, one for each scenario. This problem is known to be *NP*-hard, but solvable in pseudo-polynomial time [17]. Using the well-known flow conservation constraints, it can be modeled as an IP:

$$
\begin{aligned}
&\min\ y \\
&\text{s.t.} && y \geq c_1^\top x \\
& && y \geq c_2^\top x \\
& \sum_{e \in \delta^-(v)} x_e - \sum_{e \in \delta^+(v)} x_e = 0 && \text{for all } v \in V \setminus \{s,t\} \\
& \sum_{e \in \delta^-(s)} x_e - \sum_{e \in \delta^+(s)} x_e = 1 \\
& \sum_{e \in \delta^-(t)} x_e - \sum_{e \in \delta^+(t)} x_e = -1 \\
& && x \in \{0,1\}^A\ ,
\end{aligned}
\tag{13}
$$

where c_1 and c_2 are the cost vectors of the scenarios, s is the start node of the path, t the end node and $\delta^-(v)$ and $\delta^+(v)$ denote the ingoing and outgoing arcs of a node v.

Table 1. Results of the decomposition approach for the *two-scenario knapsack problem* and the *two-scenario shortest path problem* on $n \times n$ grid graphs.

two-scenario knapsack					two-scenario shortest path				
vars					**n**	**vars**			
dev	#s	subs	iter	time/s	dev	#s	subs	iter	time/s
600					**60**	**7080**			
1	10	5775.0	14701.4	5.44	1	10	1848.8	8321.8	30.39
5	10	4010.2	8544.8	3.18	5	10	62.0	346.9	1.33
10	10	5903.6	16208.0	5.88	10	10	1904.2	10330.2	39.49
50	10	14982.2	41181.5	15.12	50	10	1067.8	4425.9	18.07
100	10	45748.8	108384.8	40.19	100	10	7920.0	31094.3	145.58
800					**80**	**12640**			
1	10	5038.0	11913.1	5.87	1	10	10932.0	56841.7	399.18
5	10	12091.0	47233.1	23.37	5	10	286.8	1618.0	11.32
10	10	11872.0	40213.1	19.84	10	10	212.6	935.5	6.67
50	10	40848.8	108000.0	52.71	50	10	3270.8	11552.4	88.66
100	10	61176.0	145068.7	71.50	100	10	13442.2	54445.5	475.74
1000					**100**	**19800**			
1	10	8489.6	21929.7	13.13	1	10	11679.8	63458.0	712.28
5	10	38521.0	212050.8	131.49	5	10	2445.6	12540.4	138.89
10	10	15910.2	47550.7	28.67	10	10	4088.6	17532.3	199.17
50	10	157801.0	404639.4	240.22	50	10	7226.8	30919.1	395.54
100	10	115055.4	272395.1	164.54	100	10	22090.6	87446.8	1206.41
1200					**120**	**28560**			
1	10	11373.8	37921.0	28.04	1	10	19669.6	97995.1	1614.62
5	10	90018.0	544235.5	424.55	5	10	5874.8	32004.4	533.20
10	10	84065.8	389386.9	296.30	10	10	3927.2	19637.8	336.77
50	10	216866.8	592290.9	448.06	50	10	11992.2	45560.2	838.04
100	10	215516.6	515710.7	384.57	100	8	18436.5	69108.5	1388.15
1400					**140**	**38920**			
1	10	13110.0	39264.7	33.87	1	3	16400.3	74812.7	1591.10
5	10	88028.2	478130.7	437.00	5	8	2257.0	11419.2	267.74
10	8	76964.5	244141.1	209.65	10	7	1202.4	5596.6	135.97
50	10	188407.6	467830.8	415.90	50	9	7168.6	27253.2	724.82
100	10	355767.6	867242.6	761.47	100	5	22711.0	75398.8	2055.50

In our experiments, we solved the right-hand side of the decomposed problem ($LD(\lambda)$), which is a linear *shortest path problem* with fixed variables, with the network simplex optimizer of CPLEX 12.6. As the problem is successively solved for different objective functions within the subgradient algorithm, we can exploit the reoptimization feature. As before we tested the performance of the decomposition approach against solving the standard linearization of (13) with the CPLEX 12.6 MIP solver.

All tests were done on directed grid graphs which have the following form: $n \times n$ nodes are arranged on a grid. Each node is linked by an arc to the node to the right and to the node below. The start node s is the node in the upper left corner of the grid, the end node is in the lower right corner. In these graphs the total number of arcs is $2n(n-1)$ and each path consists of $2(n-1)$ arcs. One can show easily that the *two-scenario shortest path problem* remains *NP*-hard on such instances.

The two sets of arc weights were generated as for the *two-scenario knapsack problem*. First a nominal scenario was generated by randomly choosing weights in the interval $[0, 100]$, then the two scenarios were determined by randomly deviating up to $dev\%$ in both directions from the nominal weight, with $dev \in \{1, 5, 10, 50, 100\}$.

The right part of Table 1 shows the experimental results for the *two-scenario shortest path problem*. For grid sizes of up to 100×100 all instances were solved within the time limit of one hour. Instances with very small and very large deviations are hardest to solve, more subproblems need to be enumerated in the branch and bound-tree then. However, compared with the knapsack instances, the number of subproblems is significantly smaller even if the number of variables is much larger. This suggests that the dual bounds in the shortest path application are stronger.

As for the *two-scenario knapsack problem*, solving the linearized problem with a MIP solver is faster than the decomposition approach. In our experiments the longest running time over all instances was less than twenty seconds.

3.3 The Two-Scenario Minimum Spanning Tree Problem

The *minimum spanning tree problem* asks for a minimum-weight cycle-free connected subgraph of an undirected graph G that spans all nodes of G. While for the deterministic minimum spanning tree problem several efficient algorithms are known [13, 11], its discrete scenario variant is (weakly) *NP*-hard, even if the number of scenarios is limited to two [1]. We solved the right-hand side of the decomposed problem ($LD(\lambda)$) with Kruskal's algorithm [11]. A pre-processing step was added to handle fixed variables.

In contrast to the path and knapsack polytopes, the spanning tree polytope does not admit a compact characterization by linear inequalities. Therefore, to evaluate the performance of the decomposition approach for the two-scenario spanning tree we implemented a separation routine for the cut-based IP-formulation of the spanning tree polytope and solved the following IP-model

with the CPLEX 12.6 MIP solver:

$$
\begin{aligned}
&\min\ y \\
&\text{s.t.} \qquad y \geq c_1^\top x \\
&\qquad\quad y \geq c_2^\top x \\
&\qquad \sum_{e \in E} x_e = |V| - 1 \\
&\qquad \sum_{e \in \delta(S)} x_e \geq 1 \qquad \text{for all } S \subset V, S \neq \emptyset \\
&\qquad\quad x \in \{0,1\}^E \\
&\qquad\quad y \in \mathbb{R}\,,
\end{aligned}
\tag{14}
$$

where V denotes the set of nodes of the undirected graph and E the set of edges.

Instances were generated by assigning two sets of edge weights to complete graphs with $|V| \in \{100, 120, 140, 160, 180, 200\}$ as described in Section 3.1. Table 2 shows that using the Lagrangean decomposition approach we were able to solve all instances with up to 200 nodes (19900 edges) well within the time limit of 1h. In comparison, the CPLEX MIP-optimizer already failed for some instances with 50 nodes (1225 edges). The bounds obtained by solving the LP-relaxation of (14) are generally weak such that the large number of subproblems that need to be enumerated in the branch and bound-algorithm leads to long running times even for small instances.

As before, it can be observed that the running time of the decomposition approach not only depends on the size of the instance but also on the maximum deviation allowed between the two objective functions. For graph sizes up to 160 nodes, instances with 100% allowed deviation are significantly more difficult to solve than those with more similar objective functions. The average number of iterations in the subgradient algorithm needed to compute the Lagrangean dual again is low (roughly the same order of magnitude as the number of variables) and the number of subproblems grows moderately with the instance size.

3.4 The Knapsack Problem With Ellipsoidal Uncertainty

In portfolio theory an important concept is to not only consider the expected return when choosing a set of investments but also take into account the risk associated with investments. Such *mean-risk optimization problems* can be modeled using stochastic objective functions. Potential investment decisions are represented by independent random variables that have an associated expected value as well as a variance. The expected value stands for the expected return of the investments, and the variance models the uncertainty inherent in the investment, i.e. the risk that the real return deviates from the expected. The case of continuous variables is well studied whereas the case of discrete variables has received relatively little attention yet.

We concentrate on the *risk-averse capital budgeting problem* with binary variables [4]. In this variant of the mean-risk optimization problem a set of

Table 2. Results for the *two-scenario minimum spanning tree problem* on complete graphs, decomposition approach. The number of edges is $m = \frac{n(n-1)}{2}$.

n	**vars**				**n**	**vars**			
dev	#s	subs	iter	time/s	dev	#s	subs	iter	time/s
100	**4950**				**160**	**12720**			
1	10	34.0	195.1	0.39	1	10	774.8	3259.5	17.50
5	10	91.4	432.8	0.88	5	10	1699.6	10465.1	61.12
10	10	122.4	379.0	0.75	10	10	2103.6	11437.3	62.02
50	10	939.0	2966.1	5.79	50	10	7095.2	22768.5	119.92
100	10	2143.0	7416.1	16.08	100	10	7530.4	24414.3	139.24
120	**7140**				**180**	**16110**			
1	10	736.6	1993.7	5.70	1	10	1056.0	5021.7	34.78
5	10	1825.2	8327.8	23.50	5	10	11076.0	51408.8	347.98
10	10	1689.8	7964.4	22.04	10	10	7803.0	29529.3	204.74
50	10	1449.8	5333.1	14.56	50	10	6920.8	25641.0	178.14
100	10	3182.6	10496.6	32.00	100	10	7223.2	26357.5	193.92
140	**9730**				**200**	**19900**			
1	10	2099.6	10275.6	41.79	1	10	4106.8	30118.4	271.90
5	10	3266.0	15146.4	59.51	5	10	8118.4	44224.1	382.32
10	10	2300.8	8228.8	32.05	10	10	6619.2	25176.7	217.72
50	10	2375.6	7081.8	28.60	50	10	8040.4	29103.9	253.26
100	10	6785.6	24800.1	106.11	100	10	8253.6	25018.0	235.54

possible investments characterized by their costs, expected return values given by a vector c_0, variances represented by a vector a and a number ε are given as input. The number $\varepsilon > 0$ characterizes the level of risk the investor is willing to take. Investment decisions are binary. The only constraint, as in Section 3.1, is a limit on the available budget. The choice of investments guarantees that with probability $1 - \varepsilon$ the portfolio will return at least a profit of the objective value.

The corresponding nonlinear IP-model is

$$\begin{aligned} \max\ & c_0^\top x - \sqrt{\frac{1-\varepsilon}{\varepsilon} a^\top x} \\ \text{s.t. } & w^\top x \le b \\ & x \in \{0,1\}^n\ , \end{aligned} \tag{15}$$

which can easily be converted into a minimization problem of the form considered in Section 2.3. In this case the underlying combinatorial optimization problem is a *knapsack problem* as in Section 3.1.

We generated our test instances as proposed in [4]: the ellipsoid center c_0 was produced by randomly choosing weights in the interval $[0, 100]$, then the variances a were determined as squares of randomly chosen numbers in the interval between 0 and the ellipsoid center. Finally, the constraints were generated as described in Section 3.1. We produced 10 instances of each size between 1000 and 6000 and solved each instance for the values of ε given in Table 3.

Table 3. Results for the *knapsack problem with ellipsoidal uncertainty*, decomposition approach

vars / ε	#s	subs	iter	time/s	vars / ε	#s	subs	iter	time/s
1000					**4000**				
0.10	10	11561.6	25059.9	8.88	0.10	10	77751.4	173648.2	260.22
0.05	10	11901.0	25946.6	9.04	0.05	10	74034.2	167908.2	247.60
0.03	10	18080.6	40439.2	14.13	0.03	10	134695.0	311877.6	462.83
0.02	10	17116.4	38528.3	13.74	0.02	10	134252.4	303120.6	451.51
0.01	10	18227.4	40036.7	14.09	0.01	10	158270.2	370843.7	558.05
2000					**5000**				
0.10	10	38648.8	86407.2	62.46	0.10	10	105903.4	239099.0	459.70
0.05	10	43502.8	99245.1	71.46	0.05	9	229644.3	541620.1	1040.87
0.03	10	83618.2	188819.6	134.42	0.03	8	210731.2	478897.1	926.99
0.02	10	54335.8	120607.0	88.38	0.02	10	211222.2	508668.6	981.95
0.01	10	45562.8	103159.3	73.70	0.01	9	288255.7	668599.0	1302.85
3000					**6000**				
0.10	10	70660.2	158974.0	181.48	0.10	10	214092.4	494438.2	1172.40
0.05	10	65901.0	147838.1	167.27	0.05	8	303761.0	701917.5	1663.84
0.03	10	71944.2	164729.6	185.35	0.03	9	294969.7	690688.8	1610.91
0.02	10	195983.4	446662.6	502.76	0.02	6	271478.7	621214.5	1445.17
0.01	10	122424.0	281471.0	315.72	0.01	9	258226.6	611664.8	1491.80

Also here we compared the performance of the decomposition approach with the performance of the SOCP solver of CPLEX. However, we do not state the results of the SOCP solver because it was not competitive: within the time limit of 1 hour it could not even completely solve all instances with only 40 variables.

Atamtürk and Narayanan [4] present an approach to strengthen the second-order cone program by adding cutting planes to the relaxation in each node of the enumeration tree. The cutting planes are derived from the submodularity of the objective function of Problem (15). Their results show that the additional cutting planes significantly improve the dual bounds and lead to a much lower number of subproblems and faster solution times. Still, their approach is not competitive with the Lagrangean decomposition approach presented here: as reported in [4], it takes more than 800 seconds on average for solving the instances with $n = 100$ and $\varepsilon = 0.01$.

In the decomposition approach the dependence of the number of subproblems or the running times on the balance between the linear and the nonlinear parts of the objective function, i.e. the scaling factor ε, is nearly absent, which was not the case for the SOCP solver. If we take a closer look at the ratio between the number of calls to the combinatorial algorithms for the two parts of the decomposition and the number of subproblems, we see that only few calls per subproblem are necessary. For all n this ratio is less than three. Additionally, the algorithms applied to solve the subproblems are very fast in theory and in

practice. In combination with strong primal bounds this leads to very moderate overall running times.

In summary we could show that the decomposition algorithm is well suited for the risk-averse capital budgeting problem. It dramatically outperforms both standard SOCP solvers and more problem-specific approaches found in the literature.

3.5 The Shortest Path Problem With Ellipsoidal Uncertainty

In the uncorrelated ellipsoidal uncertainty variant of the *shortest path problem*, each arc is again associated with an expected value and a variance. The uncertain part of the objective function is weighted with a parameter $\Omega \in \{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{5}, \frac{1}{10}\}$, the resulting problem

$$f(x) = c_0^\top x + \Omega\sqrt{a^\top x}$$

falls into the class of problems considered in Section 2.3. The factor Ω leads to a scaling of the ellipsoid $\mathcal{U}$ by Ω^{-2}. Note that here the scaling factor is $\Omega \leq 1$, while for the *knapsack problem* with ellipsoidal uncertainty we had $\Omega = \frac{1-\varepsilon}{\varepsilon} > 1$ resulting from $0 < \varepsilon < \frac{1}{2}$.

We solved the combinatorial subproblem of the decomposition ($LD(\lambda)$) as explained in Section 3.2; for the left subproblem Algorithm 1 is directly applicable. As in Section 3.2 all tests were done on directed grid graphs with the difference in sizes and in the objective function. We generated the objective function for our instances as described in Section 3.4. We produced 10 instances of each size and type. Table 4 shows our results compared to the SOCP solver of CPLEX. For the latter, we report the number of solved instances, the average numbers of branch and bound nodes and of simplex iterations, and the running time in CPU-seconds.

Our approach could solve all but 2 instances within the time limit of 1 hour while the CPLEX solver reached its limit at 500×500 grids. Instances with smaller ellipsoid volume turned out to be easier to solve by both methods, which can be seen in all performance parameters. The number of subproblems is not substantially different in both approaches, but while CPLEX needed many simplex iterations to solve the SOCPs in the nodes of the branch and bound-tree, the number of iterations in the decomposition approach remains very small. Overall, our approach was faster than CPLEX by a factor of 10 to 100.

3.6 The Minimum Spanning Tree Problem with Ellipsoidal Uncertainty

In contrast to the two applications considered in the previous sections, it is known that the *minimum spanning tree problem* can be solved in polynomial time even under ellipsoidal uncertainty. In the following we compare our decomposition approach with the SOCP-approach and the polynomial-time algorithm suggested by [12]. The latter is based on the computation of all so-called *breakpoints*, i.e., of all points $\mu \in (0, 1)$ where the optimal spanning tree with objective

Table 4. Results for the *shortest path problem with ellipsoidal uncertainty* on $n \times n$ grid graphs with $n \in \{100, 200, 300, 400, 500\}$.

			decomposition approach				CPLEX SOCP			
n	**vars**	$\frac{1}{\Omega}$	#s	subs	iter	time/s	#s	subs	iter	time/s
100	**19800**	10	10	4.0	17.4	0.20	10	1.1	6767.8	10.58
		5	10	4.6	24.5	0.25	10	3.5	6861.6	11.18
		3	10	5.6	33.2	0.35	10	14.1	7028.5	13.02
		2	10	6.8	45.6	0.47	10	44.5	7297.1	14.60
		1	10	7.2	85.0	0.87	10	796.9	11963.0	38.14
200	**79600**	10	10	4.0	24.4	1.42	10	2.1	27410.1	101.92
		5	10	6.6	34.9	2.01	10	4.3	27643.6	120.95
		3	10	8.2	56.4	2.96	10	13.3	27995.3	129.25
		2	10	24.8	176.3	7.97	10	123.7	28982.2	153.07
		1	10	164.2	1165.8	45.30	9	6067.8	116059.3	891.38
300	**179400**	10	10	6.4	22.1	5.75	10	2.4	62384.8	399.66
		5	10	8.6	34.2	7.90	10	8.4	62855.8	433.46
		3	10	8.4	44.4	8.36	10	31.6	63579.2	464.77
		2	10	14.6	101.7	15.76	10	170.6	65024.3	539.21
		1	10	198.0	1390.6	184.63	6	6903.0	134278.5	2179.16
400	**319200**	10	10	5.6	25.4	15.37	10	1.9	112126.9	1198.64
		5	10	8.4	45.8	19.05	10	10.4	112842.1	1245.49
		3	10	14.6	84.5	29.07	10	44.6	113964.3	1408.12
		2	10	58.4	411.6	102.14	10	347.4	117252.2	1545.56
		1	9	323.2	2296.0	550.06	0	—	—	—
500	**499000**	10	10	6.4	26.0	30.04	10	2.3	176940.3	2531.90
		5	10	7.8	38.6	34.52	9	18.2	177933.3	2832.32
		3	10	26.4	157.4	89.77	8	135.4	179749.5	3011.38
		2	10	92.2	638.7	257.19	2	42.5	180003.5	2605.49
		1	9	237.2	1704.0	803.39	0	—	—	—

function $\mu c_0 + (1-\mu)a$ changes. For the spanning tree problem, the optimal solution depends on the order of objective function coefficients only. When μ varies in $(0,1)$, the order of entries in $\mu c_0 + (1-\mu)a$ can change at most $O(m^2)$ times, where m is the number of edges and hence variables in the problem. It follows that the number of breakpoints is polynomial and the best of the resulting solutions in the original objective function $c_0^\top x + \sqrt{a^\top x}$ can be determined efficiently by enumeration.

Since we are not aware of any software realizing this algorithm, we implemented it on our own. Our implementation proceeds as follows: let B denote the set of all points $\mu \in (0,1)$ where the order of entries in the vector $\mu c_0 + (1-\mu)a$ changes. Then B is computed by a bubble-sort like algorithm and sorted. The running time of this step is bounded by $O(|B| \log |B|)$. In particular, the running time is bounded by $O(m^2 \log m)$, but a small number of breakpoints leads to faster running times. In the second phase an optimal spanning tree is computed for each interval between two breakpoints using Kruskal's algorithm, yielding a worst case total running time of $O(m^3 \log m)$. Note that we speed up the computations by stopping Kruskal's algorithm whenever the best solution computed so far cannot be reached any more.

As in the two-scenario case our test instances consisted of complete graphs, with objective functions generated as described in Section 3.5. The SOCP-based approach again required a separation routine for cut inequalities to model the spanning tree polytope.

The three algorithms were able to solve instances of different sizes. While the SOCP-approach already failed to solve all instances with 30 nodes within the time limit of 1h per instance, the polynomial algorithm was able to solve all instances with up to 140 nodes, as can be seen in the right part of Table 5. Both methods are clearly outperformed by the decomposition-based branch and bound-algorithm, which solves all but one instance of size 600 within the time limit (see the left part of Table 5). The polynomial algorithm is slowed down by the large number of linear minimum spanning trees that have to be computed in order to determine the optimal solution. Our experiments show that for instances of size $n = 600$ the cardinality of the set B would already be of the order 10^9. In comparison, for the same number of variables, the decomposition approach needs to solve the linear spanning tree problem only between 3500 and 12500 times, on average.

4 Conclusion

We presented a new approach for robust combinatorial optimization using Lagrangean decomposition. We focus on two-scenario and uncorrelated ellipsoidal uncertainties and propose algorithms for solving the corresponding unconstrained binary minimization problem. These can be combined with any problem-specific algorithm for the underlying (linear) problem in order to obtain dual bounds. The latter are then used within a branch and bound-algorithm for solving the original robust problem.

Table 5. Results for the *minimum spanning tree problem with ellipsoidal uncertainty* on complete graphs

decomposition approach					polynomial algorithm		
n		**vars**			**n**	**vars**	**MSTs**
ε	#s	subs	iter	time/s	ε	#s	time/s
200		**19900**			**110**	**5995**	$\mathbf{5.28 \cdot 10^6}$
0.10	10	18.6	45.8	0.29	0.10	10	470.63
0.05	10	9.4	23.1	0.15	0.05	10	487.61
0.03	10	19.8	48.8	0.32	0.03	10	495.48
0.02	10	18.0	48.1	0.31	0.02	10	528.58
0.01	10	24.6	79.6	0.52	0.01	10	548.79
400		**79800**			**120**	**7140**	$\mathbf{7.41 \cdot 10^6}$
0.10	10	159.4	362.3	10.76	0.10	10	813.71
0.05	10	144.2	299.9	8.00	0.05	10	851.45
0.03	10	180.4	385.0	11.28	0.03	10	854.72
0.02	10	169.6	385.0	10.79	0.02	10	871.37
0.01	10	257.0	550.7	15.17	0.01	10	960.30
600		**179700**			**130**	**8385**	$\mathbf{1.03 \cdot 10e^7}$
0.10	10	6132.2	12439.1	836.54	0.10	10	1412.05
0.05	10	2217.4	4736.0	324.48	0.05	10	1439.44
0.03	9	1607.4	3427.7	283.51	0.03	10	1490.65
0.02	10	2133.6	4578.8	341.60	0.02	10	1527.21
0.01	10	2098.4	4535.2	337.05	0.01	10	1579.19
800		**319600**			**140**	**9730**	$\mathbf{1.39 \cdot 10^7}$
0.10	3	4199.7	8423.7	1132.12	0.10	10	2376.59
0.05	9	4334.6	8719.1	1117.10	0.05	10	2365.34
0.03	10	1865.8	3896.1	666.38	0.03	10	2407.78
0.02	8	4109.0	8318.8	1280.41	0.02	10	2453.49
0.01	8	1823.2	3875.6	743.80	0.01	10	2606.24
1000		**499500**			**150**	**11175**	$\mathbf{1.81 \cdot 10^7}$
0.10	1	4477.0	8959.0	2077.00	0.10	5	3524.36
0.05	3	849.0	1705.3	582.64	0.05	5	3544.72
0.03	1	6851.0	15836.0	3269.94	0.03	1	3592.62
0.02	4	2842.0	5724.2	1285.70	0.02	0	—
0.01	3	4434.3	9212.0	2192.46	0.01	0	—

It turns out that our approach is fast in practice for both types of uncertainties. While still being outperformed by CPLEX in the two-scenario case when the underlying combinatorial structure has a compact polyhedral description, the new approach is significantly faster than CPLEX both in the uncorrelated ellipsoidal case and when CPLEX needs to resort to a separation algorithm in order to model the underlying problem.

In contrast to algorithms based on mixed-integer models, our method does not require a tight polyhedral description of the underlying binary problem but can use any combinatorial algorithm as a black box. This makes our approach particularly competitive for problems not admitting a compact LP formulation such as the spanning tree problem. For the latter problem, we dramatically outperform CPLEX for both considered types of uncertainty.

In the uncorrelated ellipsoidal uncertainty case, the spanning tree problem admits a polynomial time algorithm. The reported experiments show that our general-purpose approach clearly outperforms this problem-specific polynomial time algorithm. Both algorithms are based on a successive solution of linear binary optimization problems, but our approach needs to solve a significantly smaller number of such problems in order to produce an optimal solution of the robust problem variant. This is due to the strong dual bounds computed by our algorithm as well as to the careful update and initialization of Lagrangean multipliers in our branch and bound-approach.

Bibliography

[1] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Approximating min-max (regret) versions of some polynomial problems. In Danny Z. Chen and D.T. Lee, editors, *Computing and Combinatorics*, volume 4112 of *Lecture Notes in Computer Science*, pages 428–438. Springer Berlin Heidelberg, 2006.

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

[3] Amitai Armon and Uri Zwick. Multicriteria global minimum cuts. *Algorithmica*, 46(1):15–26, 2006.

[4] Alper Atamtürk and Vishnu Narayanan. Polymatroids and mean-risk minimization in discrete optimization. *Operations Research Letters*, 36(5):618–622, 2008.

[5] Frank Baumann, Sebastian Berckey, and Christoph Buchheim. Exact algorithms for combinatorial optimization problems with submodular objective functions. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization – Festschrift for Martin Grötschel*, pages 271–294. Springer-Verlag, 2013.

[6] Frank Baumann, Christoph Buchheim, and Anna Ilyina. Lagrangean decomposition for mean-variance combinatorial optimization. In *International Symposium on Combinatorial Optimization – ISCO 2014*, 2014. To appear.

[7] George B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2), 1957.

[8] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co Ltd, 1979.

[9] Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.

[10] Panos Kouvelis and Gang Yu. *Robust Discrete Optimization and Its Applications (Nonconvex Optimization and Its Applications (closed))*. Springer, 1st edition, 1996.

[11] Joseph B. Kruskal, Jr. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[12] Evdokia Nikolova. Approximation algorithms for offline risk-averse combinatorial optimization, 2010.

[13] Robert C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

[14] Zuo-Jun Max Shen, Collette Coullard, and Mark S. Daskin. A joint location-inventory model. *Transportation Science*, 37(1):40–55, 2003.

[15] Peter Stobbe and Andreas Krause. Efficient minimization of decomposable submodular functions. *CoRR*, abs/1010.5511, 2010.

[16] Gang Yu. On the max-min 0-1 knapsack problem with robust optimization applications. *Operations Research*, 44(2):pp. 407–415, 1996.

[17] Gang Yu and Jian Yang. On the robust shortest path problem. *Computers and Operations Research*, 25:457–468, 1998.