Lagrangean Decomposition for Mean-Variance 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, focusing on the uncorrelated ellipsoidal uncertainty case, which corresponds to so-called mean-variance optimization. We present a branch and bound-algorithm for such problems that uses lower bounds obtained from Lagrangean decomposition. This approach allows to separate the uncertainty aspect in the objective function from the combinatorial structure of the feasible set. We devise a combinatorial algorithm for solving the unrestricted binary subproblem efficiently, while the underlying combinatorial optimization problem can be addressed by any black box-solver. An experimental evaluation shows that our approach clearly outperforms other methods for mean-variance optimization when applied to robust shortest path problems and to risk-averse capital budgeting problems arising in portfolio optimization.

Keywords: Robust combinatorial optimization, mean-risk optimization, Lagrangean decomposition

1 Introduction

Decision making under uncertainty is a challenge both from an economical and a mathematical perspective. In combinatorial optimization, where the constraints describe some problem-specific structure, the uncertainty usually appears in the objective function, i.e. the costs of the variables. We assume that a set $\mathcal{U}$ of potential cost vectors is given and aim at minimizing the value of a solution in its worst case scenario from this uncertainty set, i.e. we take a risk-averse attitude and consider the min-max criterion [1] to define solutions that are robust against variation of costs. That is we consider problems of the form

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

with $X \subseteq \{0,1\}^n$ defining the combinatorial structure of the feasible set. We focus on problems whose deterministic versions are easy to solve, i.e. where a linear objective function can be optimized quickly over the set X.

* 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

Reasonable choices of scenario sets $\mathcal{U}$ depend on the application at hand, but also on the (theoretical and practical) tractability of the resulting problems. Among the popular types of uncertainty we find interval uncertainties, discrete scenario sets and the so-called ellipsoidal uncertainty. In the latter case the set of all possible scenarios forms an ellipsoid in $\mathbb{R}^n$ and each point in this ellipsoid represents a possible cost vector.

In this paper we address uncorrelated ellipsoidal uncertainties. Compared with interval uncertainty, which is more commonly used and easier to deal with computationally, using ellipsoidal uncertainties can avoid overly 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 their extremes, which is very unlikely in practice. In the ellipsoidal case this is explicitely excluded. More precisely, 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.

Interest in robust optimization under ellipsoidal uncertainty has been steadily growing in recent years, with a focus on the special case of axis-parallel ellipsoids, where the problem is equivalently reformulated to a mean-variance optimization problem. Depending on the underlying combinatorial structure efficient algorithms for Problem (R) may exist. As an example, the mean-risk *spanning tree problem* can be solved in polynomial time, as shown by Nikolova [4], who also proposes general-purpose approximation schemes. For other underlying problems, such as the *shortest path problem*, the complexity of (R) is unknown.

Another general solution approach has been to solve the problem as a general mixed-integer quadratic program. Atamtürk and Narayanan [2] propose a SOCP-based branch and bound-algorithm for the robust *knapsack problem* with axis-parallel ellipsoidal uncertainty that additionally exploits the submodularity of the objective function.

In this paper we develop a new exact approach for min-max problems of type (R). 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 an efficient algorithm to solve (R) for $X = \{0,1\}^n$ in the case of uncorrelated ellipsoidal uncertainty. The combinatorial subproblem in the decomposition can be addressed by any black box-solver.

This paper is organized as follows. In Section 2 we present our Lagrangean decomposition approach for general binary nonlinear minimization problems. The special case of mean-variance combinatorial optimization is discussed in Section 3; we study the unconstrained binary optimization problem arising in the decomposition and devise an efficient algorithm that can deal with fixed variables. This allows us to embed the decomposition approach into a branch and bound-algorithm to compute provably optimal solutions. In Section 4 we evaluate our algorithm for the robust *shortest path problem* and the *risk-averse capital budgeting problem*. Extensive experimental studies show that our new algorithm clearly outperforms other approaches described in the literature.

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

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

The two minimization problems in ($L(\lambda)$) can be solved independently. The left problem is an unconstrained nonlinear minimization problem over binary variables, whereas the problem on the right is a linear instance of the underlying combinatorial problem. For any $\lambda \in \mathbb{R}^n$, ($L(\lambda)$) 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

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

for example with a subgradient algorithm. In each iteration the two subproblems of ($L(\lambda)$) have to be solved for a given λ. Note that

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

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

Lemma 1.

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

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 problem of ($L(\lambda)$), 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 ($L(\lambda)$) exactly, it is possible to solve relaxations of the subproblems. This might be advantageous when the subproblems are computationally hard or no exact algorithm is known. Even if the relaxation may decrease the quality of the resulting lower bounds and hence increase 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.

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 for warmstart.

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

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 constraints such as, e.g., non-negativity or triangle inequalities. In the following, we argue that our approach also works in this case. More precisely, we claim 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 appearing in the subproblem on the right in ($L(\lambda)$). To this end, one can show

Theorem 2. *Let* $\text{cone}(\mathcal{U})$ *denote the closed convex cone generated by* $\mathcal{U}$ *and let* $\text{cone}(\mathcal{U})^*$ *be its dual cone. Then*

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

Due to space restrictions the proof is omitted here. By Theorem 2 we can apply the Lagrangean relaxation approach directly to the problem

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

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).

Theorem 3. *Assume that* $\mathcal{C}$ *is a polyhedral cone with* $\mathcal{U} \subseteq \mathcal{C}$*. Then*

$$\max_{\lambda \in \mathcal{C}} \ L(\lambda) \quad = \quad \max_{\lambda \in \mathbb{R}^n} \ L(\lambda) \ .$$

Again, we have to omit the proof. By Theorem 3, any finite number of conditions on the objective function of one of the following types can be carried over from $\mathcal{U}$ to λ without weakening the lower bound:

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

Depending on the underlying combinatorial structure, such conditions may be crucial for linear optimization over X. This is true, e.g., when the underlying optimization problem asks for a *shortest path* or a *minimum cut* in a graph.

3 Uncorrelated Ellipsoidal Uncertainty

We now focus on the case of ellipsoidal uncertainty, i.e. the set $\mathcal{U}$ of all possible scenarios 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 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. The KKT optimality conditions yield

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

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

$$\min_{x\in\{0,1\}^n} (c_0 - \lambda)^\top x + \sqrt{x^\top A x}. \tag{2}$$

Here, c_0 and A can be interpreted as the mean 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 (2) to

$$\min_{x\in\{0,1\}^n} (c_0 - \lambda)^\top x + \sqrt{a^\top x}, \tag{3}$$

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

3.1 An Efficient Algorithm for the Unconstrained Problem

Problem (3) 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 [2]. 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 (3).

To this end, consider two solutions of (3) which differ in exactly one variable i. The difference between the corresponding objective values is

$$\Delta_i f(J) = (c_0 - \lambda)_i + \sqrt{\sum_{j\in J} a_j + a_i} - \sqrt{\sum_{j\in J} a_j}, \tag{4}$$

with J denoting the set of variables which are 1 in both solutions. The value (4) is also known as the discrete derivative of variable i [6]. 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

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

The functions C_i are strictly decreasing and therefore have at most one root each. The root r_i of C_i is 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, since the objective function of (3) is submodular.

Our basic idea for the construction of an optimal solution of (3) is that, due to the definition of r_i, a variable i cannot be 1 in an 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 observation.

Lemma 4. *There exists an optimal solution x^* of Problem (3) 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 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. □

For each i such that $-\sqrt{a_i} < (c_0 - \lambda)_i < 0$, the function C_i has exactly one positive root

$$r_i = \left(\frac{a_i - (c_0 - \lambda)_i^2}{2(c_0 - \lambda)_i} \right)^2 .$$

The algorithm for solving the unconstrained problem proceeds as follows: first variables are fixed according to Lemma 4, then the remaining non-fixed variables x_i are sorted by non-decreasing roots r_i. Finally, all binary vectors where the non-fixed entries have values that are non-increasing in the resulting order are enumerated and the best such solution is reported.

Theorem 5. *Problem (3) can be solved in time $O(n \log n)$.*

Proof. The algorithm can be implemented to run in linear time except for the sorting of variables which takes $O(n \log n)$ time. It thus remains to prove correctness. By Lemma 4 we may assume that no variable is fixed, 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

$$\sum_{i\in I} a_i \geq r_{j+1} \geq r_j \,. \tag{5}$$

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

$$\begin{aligned} f(x^1) - f(x^*) &= (c_0 - \lambda)_j + \sqrt{\sum_{i\in I} a_i + a_{j+1} + a_j} - \sqrt{\sum_{i\in I} a_i + a_{j+1}} \\ &< (c_0 - \lambda)_j + \sqrt{\sum_{i\in I} a_i + a_j} - \sqrt{\sum_{i\in I} a_i} \\ &\overset{(5)}{\leq} (c_0 - \lambda)_j + \sqrt{r_j + a_j} - \sqrt{r_j} = 0\,, \end{aligned}$$

which contradicts the optimality of x^*. □

Note that a similar algorithm for Problem (3) using a different sorting rule has been devised by Shen et al. [5].

It is easily verified that the fixings of variables arising in the branch and bound-scheme for the min-max problem do not affect the validity of our algorithm. The roots can be computed using the same formula, because an additional constant under the root does not change the order of the roots.

3.2 A Mixed-Integer SOCP Formulation

Due to the nonlinearity of the set $\mathcal{U}$, there is no straight-forward mixed-integer linear formulation of Problem (R) in the ellipsoidal uncertainty case. However, the problem can be modeled as a mixed-integer second-order cone program (SOCP), even in the correlated case: the objective function in (2) can be modeled by an SOCP constraint, while the feasible set X has to be modeled by a polyhedral description of $\text{conv}(X)$. In practice, mixed-integer SOCPs are much harder to solve than mixed-integer linear programs, making this approach much less competitive than similar linearization approaches used for min-max problems in the discrete scenario case. Moreover, if the polytope $\text{conv}(X)$ does not have a compact outer description, separation algorithm might be needed.

4 Applications

The Lagrangean decomposition approach presented in Section 3 is applicable to a wide range of robust combinatorial optimization problems. In the following we present numerical results for the *robust shortest path problem* and the

robust knapsack problem. We compare the performance of the decomposition algorithm with the standard mixed-integer SOCP approach as explained in Section 3.2, using CPLEX 12.5 to solve the resulting programs. The left subproblem of the decomposition ($L(\lambda)$), i.e. the unconstrained nonlinear binary minimization problem, was solved with the algorithm discussed in Section 3.1. The initial Lagrangean multipliers were chosen as the center of the ellipsoid.

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

4.1 The Shortest Path Problem With Ellipsoidal Uncertainty

For the uncorrelated ellipsoidal uncertainty variant of the *shortest path problem*, no polynomial time algorithm is known [4]. Here each arc in the graph is associated with a mean and a variance value. 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 of minimizing

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

over the set of shortest s,t-paths in a given directed graph falls into the class of problems considered in Section 3. The factor Ω leads to a scaling of the ellipsoid $\mathcal{U}$ by Ω^{-2}.

We solved the combinatorial subproblem of the decomposition ($L(\lambda)$) with the network simplex optimizer of CPLEX 12.5, allowing to deal with fixed variables easily. For the left subproblem of ($L(\lambda)$) the algorithm proposed in Section 3.1 is directly applicable.

All tests were done on directed grid graphs having the following form: $n \times n$ nodes are arranged on a grid, where n ranges from 100 to 500. 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 t 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. The ellipsoid center was generated by randomly choosing coefficients in the interval $[0, 100]$, then the variances were determined as squares of randomly chosen numbers in the interval between 0 and the ellipsoid center. We generated 10 instances of each size and type.

Table 1 shows our results compared to the SOCP solver of CPLEX. 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 in both cases, which can be seen in all performance indicators. While the number of subproblems is not substantially different in both approaches, CPLEX obviously spent a lot more iterations than our approach. Overall, our approach was faster than CPLEX by a factor of 10 to 100.

Table 1. Results for the *shortest path problem with ellipsoidal uncertainty* on $n \times n$ grid graphs with $n \in \{100, 200, 300, 400, 500\}$. The number of edges is $m = 2n(n-1)$

		decomposition approach				CPLEX SOCP			
vars	$\frac{1}{\Omega}$	#s	subs	iter	time/s	#s	subs	iter	time/s
19800	10	10	4.0	17.4	0.20	10	1.5	6774.1	8.98
	5	10	4.6	24.5	0.25	10	5.0	6874.4	11.43
	3	10	5.6	33.2	0.33	10	16.4	7033.9	12.83
	2	10	6.8	45.6	0.48	10	57.6	7337.5	15.03
	1	10	7.2	85.0	0.83	10	902.7	12405.1	35.45
79600	10	10	4.0	24.4	1.52	10	3.4	27482.8	91.51
	5	10	6.6	34.9	1.77	10	6.7	27725.9	109.95
	3	10	8.2	56.4	2.77	10	19.4	28076.3	130.95
	2	10	24.8	176.3	7.29	10	166.1	29214.9	159.11
	1	10	164.2	1165.8	42.93	8	4895.0	89498.8	637.97
179400	10	10	6.4	22.1	5.58	10	5.0	62455.9	369.34
	5	10	8.6	34.2	6.23	10	12.4	62903.1	405.86
	3	10	8.4	44.4	7.54	10	51.1	63674.3	453.39
	2	10	14.6	101.7	14.58	10	205.0	65409.2	529.34
	1	10	198.0	1390.6	144.99	5	9142.8	141693.6	2217.49
319200	10	10	5.6	25.4	14.11	10	4.3	112330.9	1168.52
	5	10	8.4	45.8	19.62	10	14.5	113040.9	1205.93
	3	10	14.6	84.5	24.75	10	67.4	114185.6	1364.80
	2	10	58.4	411.6	107.47	10	513.9	120765.1	1565.60
	1	9	323.2	2296.0	579.62	1	5324.0	160764.0	3350.32
499000	10	10	6.4	26.0	27.02	10	3.9	177395.6	2710.71
	5	10	7.8	38.6	31.54	10	21.6	178383.3	3076.77
	3	10	26.4	157.4	79.92	8	112.9	180086.5	3260.07
	2	10	92.2	638.7	285.77	2	52.5	180432.0	2970.01
	1	9	237.2	1704.0	849.53	0	–	–	–

Table 2. Results for the *knapsack problem with ellipsoidal uncertainty*

vars					**vars**				
ε	#s	subs	iter	time/s	ε	#s	subs	iter	time/s
1000					**4000**				
0.10	10	11970.4	26085.8	9.15	0.10	10	87337.6	200149.1	297.48
0.05	10	11999.0	26441.9	9.21	0.05	10	76084.4	173606.0	257.95
0.03	10	18363.6	40781.0	14.29	0.03	10	129048.6	296989.3	440.18
0.02	10	17761.0	40074.8	13.98	0.02	10	156201.2	355627.5	528.33
0.01	10	17861.4	38516.4	13.65	0.01	10	170836.2	397621.9	589.41
2000					**5000**				
0.10	10	39777.8	88595.8	63.12	0.10	10	128904.8	295368.9	558.42
0.05	10	43534.4	96112.8	68.76	0.05	9	243370.6	567948.6	1071.93
0.03	10	80259.2	182641.9	129.76	0.03	9	273028.8	629101.6	1200.04
0.02	10	57073.6	126914.7	90.43	0.02	10	217465.6	512935.8	972.53
0.01	10	46486.4	106990.0	76.42	0.01	10	380360.4	894375.6	1712.76
3000					**6000**				
0.10	10	72851.0	164835.7	185.49	0.10	10	214092.4	494438.2	1143.04
0.05	10	65032.8	147170.3	165.58	0.05	8	303761.0	701917.5	1632.30
0.03	10	73101.8	167410.3	187.52	0.03	9	294969.7	690688.8	1606.99
0.02	10	198490.0	461660.2	514.55	0.02	7	327128.7	754971.9	1754.51
0.01	10	127533.0	297333.5	332.10	0.01	9	258226.6	611664.8	1421.17

4.2 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 mean value as well as a variance. The mean 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 [3].

We concentrate on the *risk-averse capital budgeting problem* with binary variables [2]. In this variant of the mean-risk optimization problem we are given a set of possible investments characterized by their costs w, expected return values c_0 and variances a, as well as a number ε. The number $\varepsilon > 0$ characterizes the level of risk the investor is willing to take. Investment decisions are binary. The only constraint 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

$$\begin{aligned} \max\ & c_0^\top x - \sqrt{\frac{1-\varepsilon}{\varepsilon} a^\top x} \\ \text{s.t. } & w^\top x \leq b \\ & x \in \{0,1\}^n\ , \end{aligned} \tag{6}$$

which can easily be converted into a minimization problem of the form considered in Section 3. In this case the underlying combinatorial optimization problem is a *knapsack problem*. Note that here the scaling factor for $0 < \varepsilon < \frac{1}{2}$ is $\Omega = \frac{1-\varepsilon}{\varepsilon} > 1$, whereas for the *shortest path problem* with ellipsoidal uncertainty it was $\Omega \leq 1$.

We generated the objective function for our instances as described in Section 4.1. The constraints were created as follows: the (certain) rational weights w 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 [2] to avoid trivial instances. We generated 10 instances of each size between 1000 and 6000 and solved each instance for the values of ε given in Table 2. The legend of the table is as in Table 1.

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: already for $n = 75$ not all instances could be solved within the time limit of 1 hour.

Atamtürk and Narayanan [2] present an approach to improve the second-order cone program using additional cutting planes to strengthen the relaxation in each node of the enumeration tree. The cutting planes are derived from the submodularity of the objective function of Problem (6). 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: 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 in the branch and bound-tree, we see that only few calls per subproblem are necessary, showing the importance of reoptimization. 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.

References

[1] 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.

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

[3] Gerard Cornuéjols and Reha Tütüncü. *Optimization methods in finance*. Mathematics, finance, and risk. Cambridge University Press, Cambridge, U.K., New York, 2006.

[4] Evdokia Nikolova. Approximation algorithms for reliable stochastic combinatorial optimization. In *Proceedings of the 13th International Workshop on Approximation and the 14th International Workshop on Randomization and Computation*, APPROX/RANDOM'10, pages 338–351. Springer Berlin Heidelberg, 2010.

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

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