Polyhedral Approximation of Ellipsoidal Uncertainty Sets via Extended Formulations: A Computational Case Study*

Andreas Bärmann, Andreas Heidt, Alexander Martin,

Sebastian Pokutta and Christoph Thurner

andreas.baermann@math.uni-erlangen.de
andreas.heidt@math.uni-erlangen.de
alexander.martin@math.uni-erlangen.de
christoph.thurner@math.uni-erlangen.de

Lehrstuhl für Wirtschaftsmathematik
FAU Erlangen-Nürnberg
Erlangen, Germany

sebastian.pokutta@isye.gatech.edu

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA, USA

Abstract

Robust optimization is an important technique to immunize optimization problems against data uncertainty. In the case of a linear program and an ellipsoidal uncertainty set, the robust counterpart turns into a second-order cone program. In this work, we investigate the efficiency of linearizing the second-order cone constraints of the latter. This is done using the optimal linear outer-approximation approach by Ben-Tal and Nemirovski [2001] from which we derive an optimal inner approximation of the second-order cone. We examine the performance of this approach on various benchmark sets including portfolio optimization instances as well as (robustified versions of) the MIPLIB and the SNDlib.

Keywords: Robust Optimization – Approximation – Extended Formulations – Second-Order Cone Optimization – Mixed-Integer Programming – Portfolio Optimization

Mathematics Subject Classification: 90C31 – 90C59 – 90C20 – 90C11

Revision Date: 18.12.2015

1 Introduction

In practical applications, input data to optimization problems often results from measurements, estimations, or forecasts, which are prone to errors. These errors can have significant impact on the feasibility of the solutions obtained from the underlying model. This fact motivated the study of *robust optimization*, which sets out to deal with such data uncertainty. In fact, the sources of error can often be described stochastically using ellipsoidal probability distributions (e.g., normal distributions). In the context of robust optimization, it is therefore often justified to assume data perturbations belonging to an ellipsoidal uncertainty set. This choice for the

*Research reported in this paper was partially supported by NSF grant CMMI-1300144, BMBF grant 05M10WEC, WTT grant 12190-1, DLR grant 10-220210-C4 and ComplexWorld Research Network grant

uncertainty set leads to a robust counterpart which takes the form of a second-order cone program. In most state-of-the-art solvers, these are treated by subgradient-style approximations of the second-order cone or by interior-point methods.

In this paper, we examine a different approach, namely to linearize the ellipsoidal uncertainty sets by applying the polyhedral approximation of the second-order cone developed by Ben-Tal and Nemirovski [2001] and later slightly improved by Glineur [2000]. Compared to a gradient approximation, it has the advantage of a compact representation in terms of the number of variables and constraints. The aforementioned approach yields an outer approximation of the second-order cone. As a consequence, our polyhedral approximation of an ellipsoidal uncertainty set will be an outer approximation. This ensures that every scenario in the ellipsoidal uncertainty set will be considered in the robust counterpart. The advantage over other approaches using polyhedral uncertainty sets like those proposed by Bertsimas and Sim [2004] is the incorporation of the original assumption for the distribution of the uncertain coefficients including exact covariance information. Basically, our approach can be summarized as adopting ellipsoidal uncertainty sets in a polyhedral way by means of an approximation.

The typical way to obtain a compact representation of the arising robust counterpart is to introduce tight upper bounds on the security buffer in each row by dualizing the corresponding subproblem which checks for robust feasibility of a given solution. Here, we show how this can alternatively be done by employing an inner approximation of the second-order cone. To this end, we derive a new inner approximation based on the well-known outer approximation. The two behave identically with respect to the size of the linear system and the accuracy guarantees. Our computational results indicate the effectiveness of the method for broad problem classes.

Related work

The field of robust optimization is broad, thus we only mention the works most directly related to our study. For an introduction to robust optimization, we refer the interested reader to Ben-Tal et al. [2009]. In Bertsimas et al. [2011], a broad overview of research concerning data uncertainty in optimization problems is given, focusing on robust optimization. The tractability of the arising robust counterpart depending on the choice of the uncertainty set is summarized.

A model of data uncertainty that incorporates a budget of uncertainty for the coefficients of each constraint is introduced in Bertsimas and Sim [2004]. It restricts the number of coefficients per row that can deviate from their nominal values, which is equivalent to the choice of a specific polyhedral uncertainty set. This budget of uncertainty is less conservative than protecting the model against all possible deviations of the coefficients, which would result in an axis-parallel box as the underlying uncertainty set. The advantage of this approach is that the robust counterpart of a linear optimization problem remains linear. However, we also lose exact covariance information.

In Bertsimas and Sim [2003], the authors focus on combinatorial optimization problems and especially network flow problems. They derive a polynomial algorithm that solves the robust counterpart resulting from budgeted polyhedral uncertainty sets for polynomially solvable nominal combinatorial problems. Here, a sequence of $n + 1$ nominal problems is solved, where n is the number of binary variables. A similar result is obtained for problems for which a polynomial approximation algorithm is known.

In Bertsimas and Sim [2006], a framework is developed that uses approximations of the uncertainty sets to maintain the problem class of the nominal optimization problem in the robust counterpart. This is achieved by switching to uncertainty sets which are more conservative but easier to handle than the those obtained from ellipsoidal distribution assumptions. The disadvantage of this approach is that the idea is based on a weak relaxation, which means that the exact uncertainty set is contained in the relaxation without an approximation guarantee.

An analysis over a set of NETLIB problems (see Gay [1985]) regarding the effect of data uncertainty is performed in Ben-Tal and Nemirovski [2000]. The observation is that such un-

certainty may lead to highly infeasible nominal solutions. Therefore the level of constraint violation is determined, the nominal and the robust solutions are compared and the so-called *price of robustness* is computed. Since there is no given description of data uncertainty for the NETLIB instances, a certain perturbation set for the coefficients is assumed. A coefficient is said to be uncertain if it cannot be represented as a rational fraction $\frac{p}{q}$ with $1 \leq q \leq 100$. It is shown that the choice of ellipsoidal uncertainty sets lead to a high level of protection against data uncertainty. The robust counterparts are be SOCPs in this setup.

In Ben-Tal and Nemirovski [2001], an outer approximation of the second-order cone is developed. Key to the approach is a decomposition of the n-dimensional second-order cone $\mathbb{L}^n$ into copies of $\mathbb{L}^2$. Those are in turn approximated via a homogenized approximation of the unit disc $\mathbb{B}^2 \subset \mathbb{R}^2$. The obtained approximation of $\mathbb{B}^2$ is an extended formulation of an m-gon circumscribed to $\mathbb{B}^2$. Its size in terms of the number of variables and constraints is logarithmic in m, which is an advantage over the ordinary representation of the m-gon using two variables and m constraints and over a gradient approximation having to add a potentially large number of cuts. Furthermore, a lower bound of $\mathcal{O}(n \log \frac{1}{\epsilon})$ on the size of any polyhedral approximation of $\mathbb{L}^n$ with accuracy ϵ is established. In Glineur [2000], a slightly reduced representation of the extended formulation used for the regular m-gon is given. Moreover, an optimal choice for m for each copy of $\mathbb{L}^2$ is derived in order to obtain a prescribed accuracy for the approximation of $\mathbb{L}^n$ that achieves the bound established in Ben-Tal and Nemirovski [2001].

In Vielma et al. [2008], the authors apply the extended formulation approach of Ben-Tal and Nemirovski [2001] to approximate conic quadratic constraints and were able to tackle mixed-integer conic quadratic problems with linear solvers with applications to portfolio optimization. This approach allows for using the warm-start capabilities of modern solvers and the simplex method providing a significant advantage. It is also shown that the approach of lifted polyhedral approximations of conic constraints outperforms commercial interior-point solvers as well as solvers based on gradient approximations if the original problem contains integer variables.

Contribution

Our study can be seen as a thorough investigation of the effects observed in Vielma et al. [2008]. Motivated by those results and the need to account for data uncertainty in real-world applications, we implemented the second-order cone approximation of Ben-Tal and Nemirovski [2001] and Glineur [2000] and applied it to solve robust mixed-integer linear programs under ellipsoidal uncertainty. Staying within the realm of mixed-integer linear programming enables us to leverage the full power of modern integer programming solvers such as CPLEX and Gurobi.

To be able to derive upper bounds for the arising second-oder cone problems, we use the second-order cone approximation for a conservative approximation of the ellipsoidal uncertainty sets from outside. This allowed us to implement a generic automatic robustification scheme which can be applied as an external library in combination with Gurobi. For a given nominal problem and a description of the ellipsoidal uncertainty set, our program formulates the robust counterpart of the SOCP, as well as its linear approximation. Additionally, we develop a polyhedral inner approximation of the Lorentz cone from the polyhedral outer approximation stated in Glineur [2000] and investigate the relation between the two. In future implementations, the inner approximation will allow to avoid the tricky dualization of the outer approximations of the uncertainty sets required to compute the security buffer in the robust counterpart.

The remainder of this paper is structured as follows. In Section 2, we review the polyhedral outer-approximation approach of Ben-Tal and Nemirovski [2001] and Glineur [2000] for $\mathbb{L}^n$ and derive a similar polyhedral inner approximation. In Section 3, we give a short introduction to the methodology of robust optimization and derive the linear robust counterpart for a MILP for an uncertainty set chosen as a polyhedral approximation of an ellipsoidal set. In Section 4, we present computational results for our framework consisting of a case study on real-world

portfolio instances from Vielma et al. [2008] and instances from the MIPLIB. This is complemented by results for robust knapsack problems and robust network expansion problems built on SNDlib instances.

We confirm the superior performance of the extended-formulation based approximation approach observed in Vielma et al. [2008] for certain portfolio optimization problems. On other benchmark sets, it is vastly competitive and its performance is similar to the gradient linearization approach used within Gurobi. Remarkable is that in all cases, we obtain very good results with respect to the approximation quality.

2 Approximation of Second-Order Cone Programs

In this section, we start by briefly recalling the well-known outer approximation of the second-order cone of Ben-Tal and Nemirovski [2001] and Glineur [2000]. Then, building on their ideas, we show how their construction can be used to derive an inner approximation of the second-order cone which can be used to obtain upper bounds for second-order cone programs. Using these bounds will allow us later to approximate the conic quadratic robust counterpart to an uncertain optimization problem. This inner approximation behaves identically to the well-known outer approximation of Ben-Tal and Nemirovski [2001] and Glineur [2000] in terms of the size of the linear system and the accuracy guarantees.

2.1 Outer Approximation of the Second-Order Cone

We begin with the definition of the second-order cone, also called Lorentz cone:

Definition 2.1 (Second-order cone)**.** *The* second-order cone (SOC) $\mathbb{L}^n \subset \mathbb{R}^{n+1}$ *is defined as*

$$\mathbb{L}^n := \{(r, x) \in \mathbb{R} \times \mathbb{R}^n \mid \|x\|_2 \leq r\}.$$

The first step towards a polyhedral approximation of $\mathbb{L}^n$ is a decomposition into (a linear number of) copies of $\mathbb{L}^2$ via an *extended formulation*, i.e., we represent $\mathbb{L}^n$ as a projection of a higher-dimensional representation using additional variables. For an introduction to extended formulations, we refer the interested reader to the excellent surveys Conforti et al. [2010] and Kaibel [2011]. The decomposition of $\mathbb{L}^n$ into $\lfloor \frac{n}{2} \rfloor$ copies of $\mathbb{L}^2$ is as follows.

Theorem 2.2 (Ben-Tal and Nemirovski [2001], Glineur [2000])**.** *The convex set*

$$\left\{ (r, x, y) \in \mathbb{R}_+ \times \mathbb{R}^{n+\lfloor \frac{n}{2} \rfloor} \mid \quad x_{2i-1}^2 + x_{2i}^2 \leq y_i^2, 1 \leq i \leq \lfloor \frac{n}{2} \rfloor, \quad \left\{ \begin{array}{rcl} (r, y) & \in & \mathbb{L}^{\lceil \frac{n}{2} \rceil} \ (n \text{ even}) \\ (r, y, x_n) & \in & \mathbb{L}^{\lceil \frac{n}{2} \rceil} \ (n \text{ odd}) \end{array} \right\} \right\}$$

is an extended formulation for $\mathbb{L}^n$ with $\lfloor \frac{n}{2} \rfloor$ additional variables. The cone $\mathbb{L}^n$ is obtained by projection onto the (r, x)-space.

Iteratively applying Theorem 2.2, we can decompose $\mathbb{L}^n$ into $n-1$ copies of $\mathbb{L}^2$ using $n-2$ additional variables. Each auxiliary variable appears in exactly two different cones. Altogether, we obtain a recursion of logarithmic depth for the decomposition of $\mathbb{L}^n$. This construction is sometimes also referred to as the *tower of variables*.

In order to approximate $\mathbb{L}^n$ by a polyhedron, it suffices to do so for each copy of $\mathbb{L}^2$ in the decomposed representation of $\mathbb{L}^n$. This can be done via a homogenized approximation of the unit disc $\mathbb{B}^2$. To quantify the accuracy of an outer approximation for the latter, we use the notion of an outer ϵ-approximation. A polyhedron P is an outer ϵ-approximation of $\mathbb{B}^2$ if $\mathbb{B}^2 \subseteq P \subseteq \{(1+\epsilon)x \mid x \in \mathbb{B}^2\}$ holds. That means, an outer ϵ-approximation of $\mathbb{B}^2$ is a polyhedron which contains $\mathbb{B}^2$ and which is itself contained in a slightly larger copy of $\mathbb{B}^2$, namely $\mathbb{B}^2$ scaled up by $(1+\epsilon)$.

We choose the regular m-gon as a polyhedral approximation of $\mathbb{B}^2$ and, for the sake of exposition, we normalize the rotation. In the following, let P_m be the (unique) regular m-gon circumscribing the unit disc $\mathbb{B}^2$ with one vertex at $(\frac{1}{\cos(\frac{\pi}{m})}, 0)$. Given $\epsilon > 0$, it can be shown that any

$$m \geq \pi \arccos\left(\frac{1}{\epsilon+1}\right)^{-1} \approx \frac{\pi}{\sqrt{2\epsilon}}$$

yields a regular m-gon P_m with $\mathbb{B}^2 \subset P_m \subset (1+\epsilon)\mathbb{B}^2$. Thus, via the obvious way to describe P_m with two variables and m linear inequalities, it is necessary to double the number of inequalities in order to increase the accuracy by a factor of 4. For example, approximating $\mathbb{B}^2$ with an accuracy of 10^{-4} would already require 223 inequalities.

For the remainder of the paper, let $\gamma_i := \cos(\frac{\pi}{2^i})$ and $\sigma_i := \sin(\frac{\pi}{2^i})$ for $i \in \mathbb{Z}_+$. The following theorem presents an extended formulation for P_m that requires a significantly smaller number of defining inequalities relative to the level of accuracy.

Theorem 2.3 (Glineur [2000])**.** *Let $k \geq 2$. Then the polyhedron*

$$D_k := \left\{ (\alpha_0, \ldots, \alpha_k, \beta_0, \ldots, \beta_k) \in \mathbb{R}^{2k+2} \;\middle|\; \begin{array}{rcll} \alpha_{i+1} & = & \gamma_i\alpha_i + \sigma_i\beta_i, & (\forall i = 0, \ldots, k-1) \\ -\beta_{i+1} & \leq & \sigma_i\alpha_i - \gamma_i\beta_i, & (\forall i = 0, \ldots, k-1) \\ -\beta_{i+1} & \leq & -\sigma_i\alpha_i + \gamma_i\beta_i, & (\forall i = 0, \ldots, k-1) \\ 1 & = & \gamma_k\alpha_k + \sigma_k\beta_k & \end{array} \right\}$$

is an extended formulation for P_{2^k} with $proj_{\alpha_0,\beta_0}(D_k) = P_{2^k}$.

Remark 2.4. *The extended formulation in Theorem 2.3 can be understood as the projection of a deformed n-dimensional cube in $\mathbb{R}^n$ onto $\mathbb{R}^2$. We refer the interested reader to Fiorini et al. [2012] and Kaibel and Pashkovich [2011] for alternative constructions and interpretations.*

The construction in Theorem 2.3 leads to a very efficient representation where the number of variables and inequalities is only logarithmic in m. Its accuracy is $\epsilon \approx \frac{\pi^2}{2^{2k+1}}$ and hence increasing k by 1 (adding 2 variables, 1 equality, and 2 inequalities) improves the accuracy by a factor of 4. For example, an accuracy of 10^{-4} for the approximation of $\mathbb{B}^2$ can now be obtained for $k = 8$. The resulting extended formulation has 18 variables, 9 equalities, and 16 inequalities.

The notion of a polyhedral approximation of $\mathbb{B}^2$ naturally extends to $\mathbb{L}^n$ by homogenization. A polyhedron $\mathcal{L}$ is an *outer ϵ-approximation of* $\mathbb{L}^n$, if it satisfies $\mathbb{L}^n \subseteq \mathcal{L} \subseteq \{(r, x) \in \mathbb{R} \times \mathbb{R}^n \mid \|x\| \leq (1+\epsilon)r\}$. This leads to the following polyhedral approximation of $\mathbb{L}^2$.

Theorem 2.5 (Glineur [2000])**.** *The projection of the convex set*

$$\mathcal{L}^2_\epsilon := \left\{ (r, \alpha_0, \ldots, \alpha_k, \beta_0, \ldots, \beta_k) \in \mathbb{R}^{2k+3} \;\middle|\; \begin{array}{rcll} \alpha_{i+1} & = & \gamma_i\alpha_i + \sigma_i\beta_i, & (\forall i = 0, \ldots, k-1) \\ -\beta_{i+1} & \leq & -\gamma_i\beta_i + \sigma_i\alpha_i, & (\forall i = 0, \ldots, k-1) \\ -\beta_{i+1} & \leq & \gamma_i\beta_i - \sigma_i\alpha_i, & (\forall i = 0, \ldots, k-1) \\ r & = & \gamma_k\alpha_k + \sigma_k\beta_k & \end{array} \right\}$$

with $\epsilon > 0$ and $k = \lceil \log(\pi \arccos(\frac{1}{\epsilon+1})^{-1}) \rceil$ onto the variables (r, α_0, β_0) is an outer ϵ-approximation of $\mathbb{L}^2$.

The representation of $\mathcal{L}^2_\epsilon$ can be reduced further by eliminating variables as done in Glineur [2000]. Combining the results from above, we obtain a polyhedral outer approximation of $\mathbb{L}^n$.

Theorem 2.6 (Glineur [2000])**.** *Let $\mathcal{L}^n$ be the decomposition of $\mathbb{L}^n$ into $n-1$ copies of $\mathbb{L}^2$ obtained by the recursive application of Theorem 2.2. Approximating the $\mathbb{L}^2$-cones according to Theorem 2.5 with accuracy ϵ_l in the l-th stage, the decomposition yields a set $\mathcal{L}^n_\epsilon$ whose projection onto the (r, x)-coordinates is a polyhedral outer approximation of $\mathcal{L}^n$ with accuracy $\epsilon = \prod_{k=1}^{t}(1+\epsilon_l) - 1$, where $t = \lceil \log n \rceil$.*

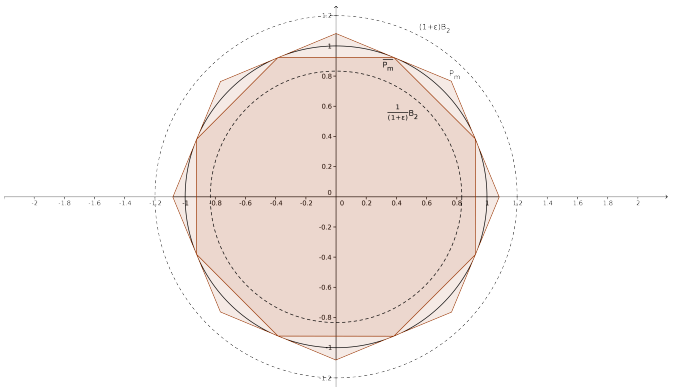

Figure 1: An illustration of the relationship between inner and outer m-gon approximations

It is shown in Glineur [2000], Theorem 2.4, that the choice

$$\epsilon_l = \cos(\frac{\pi}{2^{u_l}})^{-1} - 1, \text{where } u_l = \left\lceil \frac{l+1}{2} \right\rceil - \left\lfloor \log_4(\frac{16}{9\pi^2}\ln(1+\epsilon)) \right\rfloor,$$

in stage l of the decomposition leads to a polyhedral outer approximation $\mathbb{L}^n_\epsilon$ of $\mathbb{L}^n$ with $\mathcal{O}(n \log \frac{1}{\epsilon})$ variables and $\mathcal{O}(n \log \frac{1}{\epsilon})$ inequalities if $\epsilon < \frac{1}{2}$. This implies the choice of a homogenized 2^{u_l}-gon for the outer approximation of the $\mathbb{L}^2$-cones in stage l. It can be shown that this is the minimal size of any polyhedral approximation of $\mathbb{L}^n$ with accuracy ϵ as shown in Ben-Tal and Nemirovski [2001].

2.2 Inner Approximation of the Second-Order Cone

We now show, how the construction of the outer approximation can be used to derive a polyhedral *inner* approximation of $\mathbb{L}^n$ and characterize the relation between the two. The practical benefit of the inner approximation will be to provide a representation of the dualized outer approximation within the linear approximation of the second-order cone robust counterpart that is much easier to implement.

The inner approximation is also based on Theorem 2.2 for the recursive decomposition of $\mathbb{L}^n$ into $n-1$ copies of $\mathbb{L}^2$. However, $\mathbb{L}^2$ is now approximated via a homogenized *inner* approximation of the unit disc $\mathbb{B}^2$. An *inner ϵ-approximation of* $\mathbb{B}^2$ is a polyhedron $\bar{P}$ satisfying $\frac{1}{1+\epsilon}\mathbb{B}^2 \subseteq \bar{P} \subseteq \mathbb{B}^2$. Similarly, a polyhedron $\bar{\mathcal{L}}$ is an *inner ϵ-approximation of* $\mathbb{L}^n$ if it satisfies the inclusion $\{(r,x) \in \mathbb{R} \times \mathbb{R}^n \mid \|x\| \leq \frac{1}{1+\epsilon} r\} \subseteq \bar{\mathcal{L}} \subseteq \mathbb{L}^n$.

The natural choice for a polyhedral inner ϵ-approximation of $\mathbb{B}^2$ is a regular m-gon *inscribed* into $\mathbb{B}^2$, which we denote by $\bar{P}_m$. Given $\epsilon > 0$, it can easily be shown that any $m \geq \pi \arccos(\frac{1}{1+\epsilon})^{-1}$ yields a regular m-gon $\bar{P}_m$ with $\frac{1}{1+\epsilon}\mathbb{B}^2 \subseteq \bar{P}_m \subseteq \mathbb{B}^2$.

Remark 2.7. *Observe the close relationship between an outer and an inner ϵ-approximation. As illustrated in Figure 1, any outer ϵ-approximation of $\mathbb{B}^2$ can be used to obtain an inner ϵ-approximation of $\mathbb{B}^2$ by inverting it at $\mathbb{B}^2$. The same is true vice versa. A special case of this relationship is the approximation by regular m-gons. Let m be big enough such that P_m constitutes an outer ϵ-approximation of $\mathbb{B}^2$. Then $\bar{P}_m$ constitutes an inner ϵ-approximation of $\mathbb{B}^2$. Thus, it applies to both outer and inner approximation that any*

$$m \geq \pi \arccos\left(\frac{1}{1+\epsilon}\right)^{-1}$$

suffices to obtain an accuracy of ϵ.

In fact, for the regular m-gon the outer and inner approximations are polar to each other using the standard polar.

As before, it is not desirable to choose the obvious representation of $\bar{P}_m$ with linearly many constraints. A compact logarithmic-sized formulation as developed in the following will again allow us to obtain an accurary of $\epsilon \approx \frac{\pi^2}{2^{2k+1}}$ requiring $\mathcal{O}(k)$ variables and constraints.

Remark 2.8 (Inner approximation from outer approximation via scaling)**.** *Observe that an extended formulation for an m-gon inscribed into $\mathbb{B}^2$ could easily be obtained by replacing*

$$1 = \alpha_k \cos(\frac{\pi}{2^k}) + \beta_k \sin(\frac{\pi}{2^k}),$$

in the definition of the 2^k-gon-approximation D_k for $\mathbb{B}^2$, with the equation

$$\frac{1}{1+\epsilon_k} = \alpha_k \cos(\frac{\pi}{2^k}) + \beta_k \sin(\frac{\pi}{2^k}),$$

with $\epsilon_k = \frac{1}{\cos(\frac{\pi}{2^k})} - 1$. The resulting polytope is an extended formulation of a 2^k-gon $\bar{P}_{2^k}$ which satisfies

$$\frac{1}{1+\epsilon}\mathbb{B}^2 \subset \bar{P}_{2^k} \subset \mathbb{B}^2.$$

Consequently, replacing $r = \alpha_k \cos(\frac{\pi}{2^k}) + \beta_k \sin(\frac{\pi}{2^k})$ analogously in the approximation $\mathcal{L}^2_\epsilon$ of $\mathbb{L}^2$ yields an inner approximation of the latter. However, both polyhedral approximations are prone to numerical instability, as they contain the coefficient $\frac{1}{1+\epsilon} \approx 1$.

To avoid this instability (at least for moderate k), we will now derive a direct inner approximation of $\mathbb{B}^2$.

Theorem 2.9. *The polyhedron*

$$\bar{D}_k = \left\{ (p_0, \dots, p_{k-1}, d_0, \dots, d_{k-1}) \in \mathbb{R}^{2k} \;\middle|\; \begin{array}{rcll} p_{i-1} & = & \gamma_i p_i + \sigma_i d_i, & (\forall i = 1, \dots, k-1) \\ -d_{i-1} & \leq & \sigma_i p_i - \gamma_i d_i, & (\forall i = 1, \dots, k-1) \\ d_{i-1} & \leq & \sigma_i p_i - \gamma_i d_i, & (\forall i = 1, \dots, k-1) \\ p_{k-1} & = & \gamma_k & \\ -d_{k-1} & \leq & \sigma_k & \\ d_{k-1} & \leq & \sigma_k & \end{array} \right\}$$

for $k \geq 2$ is an extended formulation for $\bar{P}_{2^k}$ with $proj_{p_0,d_0}(\bar{D}_k) = \bar{P}_{2^k}$.

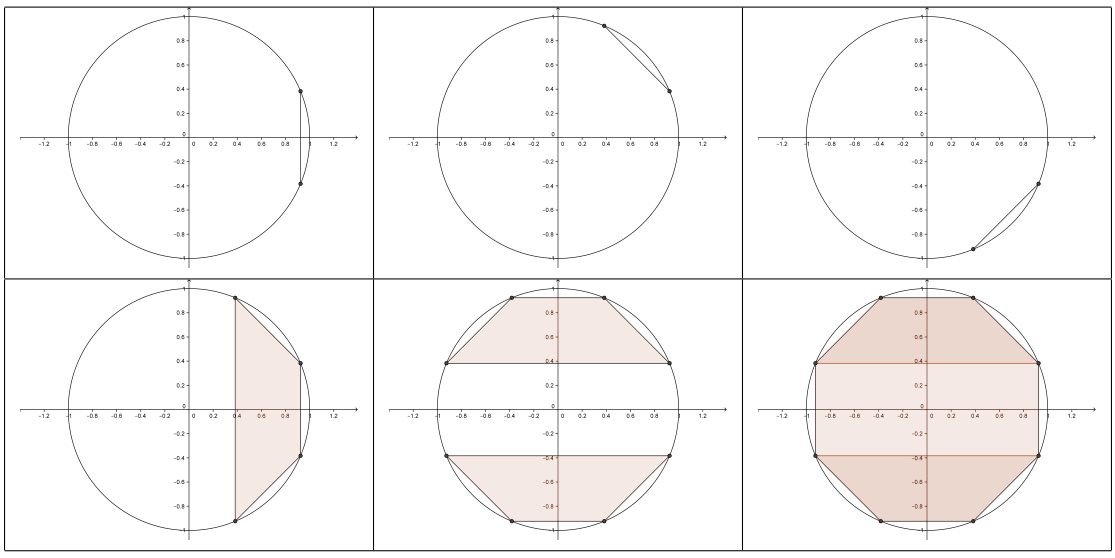

Figure 2: Construction of the inner approximation of the unit disc $\mathbb{B}^2$ for $k = 3$

Proof. In the following, we describe the construction of the inner approximation as an iterative procedure. We start by defining the polytope

$$P_{k-1} \coloneqq \{(p_{k-1}, d_{k-1}) \mid p_{k-1} = \gamma_k, -\sigma_k \leq d_{k-1} \leq \sigma_k\}.$$

Now, we construct a sequence of polytopes $P_{k-1}, P_{k-2}, \ldots, P_0$. Assume that polytope P_i has already been constructed. In order to obtain polytope P_{i-1} from polytope P_i, we perform the following actions which we will translate into mathematical operations below:

1. Rotate P_i anticlockwise by an angle of $\theta_i = \frac{\pi}{2^i}$ around the origin to obtain a polytope P_i^1,
2. Reflect P_i^1 at the x-axis to obtain a polytope P_i^2,
3. Form the convex hull of P_i^1 and P_i^2 to obtain polytope P_{i-1}.

The first step is a simple rotation and can be represented by the linear map

$$\mathcal{R}_\theta : \mathbb{R}^2 \mapsto \mathbb{R}^2, \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

The reflection at the x-axis corresponds to the linear map

$$\mathcal{M} : \mathbb{R}^2 \mapsto \mathbb{R}^2, \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Thus, the composition $\mathcal{M}\mathcal{R}_{\theta_i}$ which first applies $\mathcal{R}_{\theta_i}$ and then $\mathcal{M}$, is given by

$$\mathcal{M}\mathcal{R}_{\theta_i} : \mathbb{R}^2 \mapsto \mathbb{R}^2, \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

With this, we obtain $P_i^1 = \mathcal{R}_{\theta_i}(P_i)$ and $P_i^2 = (\mathcal{M}\mathcal{R}_{\theta_i})(P_i)$. Finally, adding the two constraints

$$-d_{i-1} \leq \sigma_i p_i - \gamma_i d_i$$

and

$$d_{i-1} \leq \sigma_i p_i - \gamma_i d_i$$

yields a polyhedron whose projection onto the variables (d_{i-1}, p_{i-1}) is $P_{i-1} = conv(P_i^1, P_i^2)$. Keeping this correspondence in mind, we show that $P_0 = \bar{P}_{2^k}$.

In each iteration, P_i is rotated anticlockwise by an angle of θ_i around the origin, such that the vertex of P_i with minimal vertical coordinate is rotated to (γ_k, σ_k), therefore $P_i^1 = \mathcal{R}(P_i)$. It is $|\mathcal{V}(P_i^1)| = |\mathcal{V}(P_i)|$ and P_i^1 lies strictly above the horizontal axis. Applying $\mathcal{M}$, we obtain $P_i^2 = \mathcal{M}(P_i^1)$ which satisfies $|\mathcal{V}(P_i^2)| = |\mathcal{V}(P_i^1)|$ and lies strictly below the horizontal axis. Then $P_{i-1} = conv(P_i^1, P_i^2)$ satisfies $|\mathcal{V}(P_{i-1})| = 2|\mathcal{V}(P_i)|$ because all vertices $v \in \mathcal{V}(P_i^1) \cup \mathcal{V}(P_i^2)$ remain extreme points of P_i. We obtain polytope P_0 after $k-1$ iterations of the above procedure, which has $|\mathcal{V}(P_0)| = 2^k$ vertices. As the interior angles at each vertex of P_0 are of equal size, it follows $P_0 = \bar{P}_{2^k}$. This proves the correctness of our construction. □

The intermediate steps of the construction are depicted in Figure 2 for the case $k = 3$, which leads to an octagon-approximation. The upper left picture shows the initial polytope P_2, which is an interval on the line $x = \gamma_k$. The upper middle and upper right picture show its rotation by 45° counterclockwise and clockwise, thus representing P_2^1 and P_2^2, respectively. The lower left picture shows P_1 as the convex hull of P_2^1 and P_2^2. The lower middle picture contains both P_1^1 and P_1^2 as a rotation of P_1 by 90° in both directions. Finally, the lower right picture shows $P_0 = \bar{P}_{2^3}$ as the convex hull of P_1^1 and P_1^2.

Observe that the proof above can also be understood in terms of the framework of Kaibel and Pashkovich [2011] using the notion of reflection relations. From Theorem 2.9 follows that the inner ϵ-approximation $\bar{D}_k$ of $\mathbb{B}^2$ can be used to yield an inner ϵ-approximation of $\mathbb{L}^2$ by homogenizing it:

Corollary 2.10. *The projection of the set*

$$\bar{\mathcal{L}}^2_\epsilon = \left\{ (s, p_0, \ldots, p_{k-1}, d_0, \ldots, d_{k-1}) \in \mathbb{R}^{2k} \;\middle|\; \begin{array}{rcll} p_{i-1} & = & \gamma_i p_i + \sigma_i d_i, & (\forall i = 1, \ldots, k-1) \\ -d_{i-1} & \leq & \sigma_i p_i - \gamma_i d_i, & (\forall i = 1, \ldots, k-1) \\ d_{i-1} & \leq & \sigma_i p_i - \gamma_i d_i, & (\forall i = 1, \ldots, k-1) \\ p_{k-1} & = & \gamma_k s & \\ -d_{k-1} & \leq & \sigma_k s & \\ d_{k-1} & \leq & \sigma_k s & \end{array} \right\}$$

with $\epsilon > 0$ and $k = \lceil \log(\pi \arccos(\frac{1}{\epsilon+1})^{-1}) \rceil$ onto the variables (s, p_0, d_0) is an inner ϵ-approximation of $\mathbb{L}^2$.

This corollary allows for the derivation of an inner ϵ-approximation of $\mathbb{L}^n$ using a similar construction as for the outer approximation presented in Section 2.1. Obviously, employing the decomposition technique from Theorem 2.2 and replacing the $\mathbb{L}^2$-cones by the inner ϵ-approximation of the above corollary yields an inner approximation of $\mathbb{L}^n$. Similar to Theorem 2.6 we have

Theorem 2.11. *Let $\mathcal{L}^n$ be the decomposition of $\mathbb{L}^n$ into $n-1$ copies of $\mathbb{L}^2$ as obtained by the recursive application of Theorem 2.2. Approximating these $\mathbb{L}^2$-cones according to Theorem 2.9 with accuracy ϵ_l in the l-th stage of the decomposition yields a set $\bar{\mathcal{L}}^n_\epsilon$, whose projection onto its (r, x)-coordinates is a polyhedral inner approximation of $\mathbb{L}^n$ with accuracy $\epsilon = \prod_{l=1}^{t}(1+\epsilon_l) - 1$, where $t = \lceil \log n \rceil$.*

Proof. To prove the accuracy claimed in the statement, it suffices to show $\{(r, x) \in \mathbb{R} \times \mathbb{R}^n \mid \|x\| \leq \frac{1}{1+\epsilon} r\} \subseteq proj_{r,x}(\bar{\mathcal{L}}^n_\epsilon)$, as the inclusion $proj_{r,x}(\bar{\mathcal{L}}^n_\epsilon) \subseteq \mathbb{L}^n$ is immediate.

Let $(r, x) \in \{(r, x) \in \mathbb{R} \times \mathbb{R}^n \mid \|x\| \leq \frac{1}{1+\epsilon} r\} = \{(r, x) \in \mathbb{R} \times \mathbb{R}^n \mid \|x\| \leq \frac{1}{1+\epsilon} r\}$, which is equivalent to $\|(1+\epsilon)x\| \leq r$. Thus, we have

$$r^2 \geq \|(1+\epsilon)x\|^2 = (1+\epsilon)^2 \sum_{i=1}^{n} x_i^2.$$

With this in mind, we first consider the case of even n. In the first stage of the decomposition, the condition $(r, x) \in proj_{r,x}(\bar{\mathcal{L}}^n_\epsilon)$ requires us to find suitable values of y_i with $i = 1, \ldots, \frac{n}{2}$, such that both

$$(x_i^2 + x_{i+1}^2) \leq \frac{y_i^2}{(1+\epsilon_1)^2} \text{ for } i = 1, \ldots, \frac{n}{2} \tag{1}$$

and

$$\sum_{i=1}^{\frac{n}{2}} y_i^2 \leq \frac{r^2}{(1+\epsilon')^2} \tag{2}$$

hold. In Condition (2), ϵ' denotes the accuracy chosen for the approximation of the remaining cone $\mathbb{L}^{\frac{n}{2}}$. Condition (1) can easily be satisfied by choosing $y_i \geq 0$, such that

$$y_i^2 = (1+\epsilon_1)^2 (x_{2i-1}^2 + x_{2i}^2) \text{ with } i = 1, \ldots, \frac{n}{2}.$$

We need to ensure that this choice also satisfies (2). For this observe

$$\sum_{i=1}^{\frac{n}{2}} y_i^2 = (1+\epsilon_1)^2 \sum_{i=1}^{\frac{n}{2}} (x_{2i-1}^2 + x_{2i}^2) = (1+\epsilon_1)^2 \|x\|^2 \leq (1+\epsilon_1)^2 \frac{r^2}{(1+\epsilon)^2}.$$

Thus, Condition (2) holds, if

$$\frac{(1+\epsilon_1)^2}{(1+\epsilon)^2} \leq \frac{1}{(1+\epsilon')^2}$$

is satisfied, which in turn holds for

$$(1+\epsilon)^2 \geq (1+\epsilon_1)^2(1+\epsilon')^2.$$

This results in a guarantee of $\epsilon = (1+\epsilon_1)(1+\epsilon') - 1$ for the accuracy of our inner approximation of $\mathbb{L}^n$.

To attain the same guarantee for odd n, we need to show that our above choice for the y_i's satisfies

$$x_n^2 + \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} y_i^2 \leq \frac{r^2}{(1+\epsilon')^2}$$

instead of (2). This is possible by using

$$x_n^2 + \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} y_i^2 = x_n^2 + (1+\epsilon_1)^2 \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} (x_{2i-1}^2 + x_{2i}^2) \leq (1+\epsilon_1)^2 \sum_{i=1}^{n} x_i^2 = (1+\epsilon_1)^2 \|x\|^2.$$

From here, we can proceed as in the even case to obtain a guarantee of $\epsilon = (1+\epsilon_1)(1+\epsilon') - 1$.

The proof for the first stage of the decomposition can be applied recursively to the remaining cone in each subsequent stage. Altogether, we have shown that the accuracy of the inner approximation is $\epsilon = \prod_{l=1}^{t}(1+\epsilon_i) - 1$, where $t = \lceil \log n \rceil$ is the number of stages of the decomposition. □

Remark 2.12. *The above proof indicates that there is no obvious gain in accuracy by choosing different accuracies for the $\mathbb{L}^2$-cones across a given stage of the decomposition. For accuracies ϵ_1^i, $i = 1, \ldots, \lfloor \frac{n}{2} \rfloor$, in stage 1, we estimate*

$$\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} y_i^2 = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} (1+\epsilon_1^i)^2 (x_{2i-1}^2 + x_{2i}^2) \leq (1 + \max_{i=1,\ldots,\lfloor \frac{n}{2} \rfloor} \epsilon_1^i)^2 \|x\|^2.$$

This can be made reasonably tight, so that the overall accuracy in a given stage depends mainly on the worst accuracy of any $\mathbb{L}^2$-approximation therein.

Having established a guarantee for the quality of the inner approximation of $\mathbb{L}^n$, the remaining question is how to choose the values of ϵ_l in stage l of the decomposition in order to obtain a representation of small size, which can be answered using Remark 2.7. Furthermore, we can see from the proof of Theorem 2.11 that outer and inner approximation of $\mathbb{L}^n$ obey the same formula for the dependence of the overall accuracy. Thus the size of our inner ϵ-approximation of $\mathbb{L}^n$ is $\mathcal{O}(n \log \frac{1}{\epsilon})$ variables and $\mathcal{O}(n \log \frac{1}{\epsilon})$ constraints if $\epsilon < \frac{1}{2}$:

Corollary 2.13. *Choosing intermediate accuracies*

$$\epsilon_l = \cos(\frac{\pi}{2^{u_l}})^{-1} - 1, \text{where } u_l = \left\lceil \frac{l+1}{2} \right\rceil - \left\lfloor \log_4(\frac{16}{9\pi^2} \ln(1+\epsilon)) \right\rfloor,$$

for the inner $\mathbb{L}^2$-approximations in stage l of the decomposition allows for an inner approximation of $\mathbb{L}^n$ using $\mathcal{O}(n \log \frac{1}{\epsilon})$ variables and $\mathcal{O}(n \log \frac{1}{\epsilon})$ constraints if $\epsilon < \frac{1}{2}$.

Regardless of potential number issues, scaling our inner ϵ-approximation for $\mathbb{B}^2$ by $(1+\epsilon)$ yields an outer approximation of $\mathbb{B}^2$. Hence, the construction yields an outer ϵ-approximation of $\mathbb{L}^n$ and is therefore subject to the lower bound established in Ben-Tal and Nemirovski [2001], showing that it is optimal in terms of size.

Corollary 2.14. *The size of the inner ϵ-approximation of $\mathbb{L}^n$ stated in Theorem 2.11 using the intermediate accuracies of Corollary 2.13 is optimal in the sense that there is no other inner ϵ-approximation of $\mathbb{L}^n$ using fewer variables and constraints.*

2.3 Upper Bounds for SOCPs via linear Approximation

In order to derive a linear approximation for the SOCP robust counterpart of an LP or MIP, we need to be able to find tight upper bounds for SOCPs. Therefore, we consider the following second-order cone program (P) with a single second-order cone constraint:

$$\begin{array}{lrcl} \max & c^T \begin{pmatrix} r \\ x \end{pmatrix} & & \\ \text{s.t.} & A \begin{pmatrix} r \\ x \end{pmatrix} + b & = & 0 \\ & (r, x) & \in & \mathbb{L}^n. \end{array}$$

A straightforward upper bound can be obtained by solving the associated dual problem (D):

$$\begin{array}{lrcl} \min & b^T y & & \\ \text{s.t.} & A^T y - \begin{pmatrix} \rho \\ \xi \end{pmatrix} + c & = & 0 \\ & (\rho, \xi) & \in & \mathbb{L}^n. \end{array}$$

This primal-dual pair has several convenient properties, of which the following strong duality result is the most important one for us:

Theorem 2.15 (Glineur [2001])**.** *If both (P) and (D) possess a strictly feasible solution, both problems attain their respective optimal values p^* and d^* and the latter two coincide, i.e., there are optimal solutions (r^*, x^*) to (P) and (y^*, ρ^*, ξ^*) to (D) with*

$$c^T \begin{pmatrix} r^* \\ x^* \end{pmatrix} = p^* = d^* = b^T y^*.$$

The above theorem allows us to substitute the SOC maximization subproblem of the form (P) by the corresponding dual (D) retaining the same optimal value. Furthermore, replacing the constraint $(\rho, \xi) \in \mathbb{L}^n$ by a polyhedral inner relaxation yields valid upper bounds for the optimal value of (P). We will make use of this relation when approximating the auxiliary optimization problem which finds a worst-case scenario from an ellipsoidal uncertainty set to a given solution to the nominal problem, as introduced in the next section.

3 Approximation of the Conic Quadratic Robust Counterpart

We will now turn our attention to robust optimization with ellipsoidal uncertainty sets and the approximation of the so-arising SOC robust counterpart.

3.1 Robust Optimization

Consider the following optimization problem:

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{s.t.} & \hat{A} x & \leq & b \\ & x & \in & \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p} \end{array} \tag{3}$$

for a matrix $\hat{A} \in \mathbb{R}^{m \times n}$, vectors $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ and $1 \leq p \leq n$. We call (3) the *nominal* optimization problem. If the constraint matrix $\hat{A}$ is affected by data uncertainty according to an uncertainty set U, its *robust counterpart (RC)* is given by

$$\begin{array}{lrcll} \min & c^T x & & & \\ \text{s.t.} & Ax & \leq & b & (\forall A \in U) \\ & x & \in & \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p}. & \end{array} \tag{4}$$

A vector $x \in \mathbb{R}^n$ that is feasible for (4) is called *robust feasible*. Given the nominal matrix $\hat{A}$, the uncertainty set U can be formulated as

$$U = \{A \in \mathbb{R}^{m \times n} \mid A = \hat{A} + \tilde{A}, \quad \tilde{A} \in S\}, \tag{5}$$

where S is a closed set, the so-called *perturbation set*. Assuming this form of the uncertainty set U, we can give a more intuitive formulation of the robust counterpart. In the following, let A_i denote the i-th row of matrix A.

Theorem 3.1 (Bertsimas et al. [2004])**.** *The robust counterpart (RC) for an uncertainty set U given in the form* (5) *can be stated as*

$$\begin{array}{lrcll} \min & c^T x & & & \\ \text{s.t.} & \hat{A}_i x + \max\limits_{\tilde{A} \in S} \tilde{A}_i x & \leq & b_i & (\forall i = 1, \ldots, m) \\ & x & \in & \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p}. & \end{array}$$

This characterization suggests the following interpretation of the robust counterpart: Each constraint of the nominal problem is immunized against uncertainty in the data by adding a security buffer. The latter is chosen according to the worst possible outcome of the data realization for that specific row. Its evaluation for a fixed $\bar{x}$ amounts to solving the following subproblems for $i = 1, \ldots, m$:

$$\begin{array}{lrcl} \max & \tilde{A}_i \bar{x} & & \\ \text{s.t.} & \tilde{A} & \in & S. \end{array} \tag{6}$$

Thus, it suffices to consider constraint-wise uncertainty, i.e., the case of a separate uncertainty set U_i for each row A_i of the constraint matrix A. For the sake of exposition and without loss of generality, we therefore confine ourselves to nominal problems with one constraint from now on unless stated otherwise.

3.2 The Linear Robust Counterpart

We first consider the case where the perturbation set S is given as a polyhedron, i.e.,

$$U = \{a \in \mathbb{R}^n \mid a = \hat{a} + \tilde{a}, D\tilde{a} \leq d\}, \tag{7}$$

where D is a matrix in $\mathbb{R}^{r \times n}$, and d is a vector in $\mathbb{R}^r$. In this case (4) can be formulated as a linear program again.

Theorem 3.2 (Bertsimas et al. [2011])**.** *The robust counterpart* (4) *of the nominal problem* (3) *with $\hat{A} \in U$ and U as in* (7) *can be written as:*

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{s.t.} & \hat{a}^T x + d^T \pi & \leq & b \\ & D^T \pi & = & x \\ & x & \in & \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p} \\ & \pi & \geq & 0. \end{array} \tag{8}$$

The above also holds for mixed-integer problems as long as we consider continuously valued data uncertainty, as the derivation only requires strong duality for subproblem (6): the x-variables are fixed here and hence (6) only consists of continuous variables.

3.3 Linear Approximation of the SOC Robust Counterpart

The aim of our computational study is to analyze the performance of solving robust mixed-integer optimization problems with ellipsoidal uncertainty sets where the latter are approximated via the polyhedral approximation of the second-order cone. The choice of ellipsoidal uncertainty sets is motivated by the approximation so-called *probabilistic* or *chance constraints* as well as mean-variance optimization problems (see Ben-Tal et al. [2009] for a detailed treatment). We consider the robust counterpart for an ellipsoidal uncertainty set U of the form:

$$U = \{a \in \mathbb{R}^n \mid a = \hat{a} + Bz, \|z\| \leq 1\}. \tag{9}$$

The corresponding robust counterpart is a conic quadratic optimization problem.

Theorem 3.3 (Ben-Tal and Nemirovski [2000])**.** *The robust counterpart* (4) *of* (3) *with* U *as in* (9) *can be written as*

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{s.t.} & \hat{a}^T x + \|B^T x\| & \leq & b \\ & x & \in & \mathbb{Z}^p_+ \times \mathbb{R}^{n-p}_+. \end{array} \tag{10}$$

Thus, optimizing the robust counterpart (10) involves the solution of a second-order cone problem. Especially in the presence of integral variables, genuine quadratic methods for the solution of such problems, like interior-point methods (e.g., Bonnans et al. [2009]), suffer from the lack of warm-start capabilities. Thus, in a branch-and-bound framework, they cannot reuse the optimal solution to a node in the branch-and-bound tree in the solution of the child nodes, which is a significant disadvantage. In contrast to that, the branch-and-bound method for the solution of mixed-integer programs can make efficient use of this information. Additionally, the solution process is enhanced by other advanced MIP techniques like cutting planes and preprocessing, which are implemented in standard MIP solvers.

To benefit from these advantages, we derive a linear approximation of the second-order cone robust counterpart (10) which exploits the structure of the ellipsoidal uncertainty set. Replacing the unit-ball $\mathbb{B}^n$ in the definition of U by an outer ϵ-approximation $\mathcal{B}^n$ obviously yields an outer ϵ-approximation of $\mathcal{U}$ of U. This is advantageous, since $\mathcal{U}$ contains all the scenarios in U and therefore the approximate robust counterpart according to $\mathcal{U}$ is guaranteed to produce feasible robust solutions to the original robust counterpart according to U. On the other hand, an outer ϵ-approximation ensures that $\mathcal{U}$ is not overly conservative, as its scenarios are either contained in U or at least very close to other scenarios in U.

Let U be given as in (9). Substituting $z = B^{-1}(a - \hat{a})$ yields

$$U = \{a \in \mathbb{R}^n \mid \|B^{-1}(a - \hat{a})\| \leq 1\} = \{a \in \mathbb{R}^n \mid B^{-1}(a - \hat{a}) \in \mathbb{B}^n\}.$$

Now let $\mathcal{B}^n$ be a polyhedral outer ϵ-approximation of $\mathbb{B}^n$ given by $\mathcal{B}^n = \{z \mid Kz \leq k\}$. Note that this notation assumes the outer approximation to live in the same space of variables as the unit ball itself, which would be the situation after applying projection to the outer approximation from Section 2.1. However, transferring the following considerations to the case of an approximation given by an extended formulation is straightforward. Replacement of $\mathbb{B}^n$ by $\mathcal{B}^n$ in the last equation yields the approximate uncertainty set $\mathcal{U}$, which can be stated as follows:

$$\begin{array}{rcl} \mathcal{U} & = & \{a \in \mathbb{R}^n \mid B^{-1}(a - \hat{a}) \in \mathcal{B}^n\} \\ & = & \{a \in \mathbb{R}^n \mid K(B^{-1}(a - \hat{a})) \leq k\} \\ & = & \{a \in \mathbb{R}^n \mid KB^{-1}a \leq k + KB^{-1}\hat{a}\}. \end{array}$$

Thus, for the choice $D := KB^{-1}$ and $d := k + KB^{-1}\hat{a}$, the system $Da \leq d$ defines the polyhedron $\mathcal{U}$ approximating U. Using Theorem 3.2, this leads to the following form of the approximate robust counterpart:

Corollary 3.4. *Let the uncertainty set $\mathcal{U}$ be given by $\mathcal{U} = \{a \in \mathbb{R}^n \mid B^{-1}(a - \hat{a}) \in \mathcal{B}^n\}$, where $\mathcal{B}^n$ is an outer ϵ-approximation of $\mathbb{B}^n$ with $\mathcal{B}^n = \{z \mid Kz \leq k\}$. Then the robust counterpart* (8) *takes the form*

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{s.t.} & \hat{a}^T x + k^T \pi & \leq & b \\ & B^T x - K^T \pi & = & 0 \\ & x & \in & \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p} \\ & \pi & \in & \mathbb{R}_+^r. \end{array}$$

Building on the discussion in Section 2.3, it as well possible to state the approximate robust counterpart in terms of an inner approximation of $\mathbb{L}^n$. To see this, consider Subproblem (6), which returns a worst-case scenario from the set U for a given solution $\bar{x}$ to the original problem (3). For U as in (9), it is an SOCP of the following form:

$$\begin{array}{lrcl} \max & \bar{x}^T a & & \\ \text{s.t.} & B^{-1} a & \in & \mathbb{B}^n. \end{array} \tag{11}$$

Its dual can be written as:

$$\begin{array}{lrcl} \min & \rho & & \\ \text{s.t.} & (\rho, B^T \bar{x}) & \in & \mathbb{L}^n. \end{array}$$

Now, using Theorem 2.15, we obtain an upper bound for subproblem (11) by replacing $\mathbb{L}^n$ in the dual problem 3.3 by an inner approximation. Let this inner approximation be given by the polyhedral set $\tilde{\mathcal{L}}^n = \{(\rho, \xi) \mid W\xi \leq w\rho\}$, again assuming a representation in the original variable space. Then the approximate subproblem reads:

$$\begin{array}{lrcl} \min & \rho & & \\ \text{s.t.} & WB^T \bar{x} & \leq & w\rho. \end{array} \tag{12}$$

This allows us to state the corresponding approximate linear robust counterpart in a closed form:

Theorem 3.5. *Let the uncertainty set $\mathcal{U}$ be given by $\mathcal{U} = \{a \in \mathbb{R}^n \mid B^{-1}(a - \hat{a}) \in \mathbb{B}^n\}$, and let $\tilde{\mathcal{L}}^n = \{(\rho, \xi) \mid W\xi \leq w\rho\}$ be an inner approximation of $\mathbb{L}^n$. Then the projection of the feasible set of the optimization problem*

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{s.t.} & \hat{a}^T x + \rho & \leq & b \\ & WB^T x & \leq & w\rho \\ & x & \in & \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p} \end{array}$$

onto variable x is a subset of the feasible set of the SOCP robust counterpart (10).

The immediate consequence of the above reasoning is that we can establish a guarantee of factor $(1 + \epsilon)$ on the approximation of the security buffer of an uncertain constraint, as this is the optimal value of the approximate subproblem (12) if $\mathbb{L}^n$ is approximated with accuracy ϵ from inside. In the case of outer ϵ-approximation of $\mathbb{L}^n$, the same bound follows from inspection of the corresponding subproblem 6.

As a result, we see that a tight outer approximation of the uncertainty yields a tight upper bound on the security buffer in the approximately robustified constraint. Thus, we can hope to obtain a tight approximation of the optimal robust objective value, too, although this is not guaranteed, especially in the presence of integral variables. The computational results in Section 4 include an empirical study on the influence of the accuracy ϵ on the quality of the approximate robust objective value.

Remark 3.6. *When either using Corollary 3.4 in conjunction with the outer approximation of $\mathbb{L}^n$ from Section 2.1 or instead using Theorem 3.5 together with the inner approximation of $\mathbb{L}^n$ from Section 2.2, the two corresponding approximate robust counterparts are equivalent with respect to the approximation guarantee. This is a consequence of the above discussion. Moreover, the size of the resulting linear system is roughly the same in both cases. However, an implementation in terms of an inner approximation is much easier to accomplish as it avoids the complicated dualization of the nested decomposition according to Theorem 2.2. Rather than first applying Theorem 2.2 to the* outer *approximations of $\mathbb{L}^2$ cones and then dualizing this tower-of-variables construction, we can directly use the primal representation of the decomposition together with* inner *approximations of $\mathbb{L}^2$ cones. Preliminary computational tests comparing the implementations with the L^2 building blocks being either dualized outer approximations or primal inner approximations indicated that they behave the same in terms of solution quality.*

4 Computational Results

In this section, we present the results of our approximate robust optimization framework for different benchmarks sets. We compare our results to those obtained for the exact quadratic formulation of the robust problems with respect to different performance measures. We begin with a description of implementation details and parameter choices and then describe the instances of the benchmark sets. Finally, we present the computational results.

4.1 Implementation and Test Instances

We implemented an exact quadratic robust optimization framework as well as our approximate framework described in the previous sections. We therefore developed a command line tool that takes as input the nominal problem in any standard format for linear programs together with a file stating the uncertain parameters and the matrices defining their ellipsoidal uncertainty sets. It then solves either the quadratic or the approximating linear robust counterpart, as specified. For the implementation, we used the C++-Interface of Gurobi 5.0 (Gurobi Optimization, Inc. [2013]).

Our implementation uses the formulation given in Corollary 3.4 instead of that from Theorem 3.5. This is because our computational results actually preceded and motivated the development of the inner approximation. Getting a correct implementation with the required nested dualizations is cumbersome and nontrivial. These complications completely disappear when employing the inner approximation directly. In terms of solution quality these two formulations will be equivalent – one might think of the inner approximation being the nested dualizations made explicit (see Remark 3.6).

In the following, we compare the exact quadratic model and our approximate polyhedral approach on different benchmark sets. These are the the portfolio optimization instances used in Vielma et al. [2008], robustified versions of MIPLIB instances (cf. Koch et al. [2011]), random robust knapsack problems, as well as network expansion problems created from SNDlib instances (cf. Orlowski et al. [2007]). In the tables and diagrams presented here, "QR" generally refers to results for the solution of the quadratic robust counterpart while "LR-ϵ" stands for the linearized robust counterpart with an approximation accuracy of ϵ. For the latter, we will particularly focus on the quality of the approximation as well as its computational complexity. The computations were performed on a compute server with six-core AMD Opteron 2435 processors and 64 GB RAM as well as a time limit of one hour per instance. Unless stated otherwise, we used Gurobi's default parameter settings.

4.2 Results on Portfolio Instances from the Literature

In this section, we consider a set of portfolio instances from Vielma et al. [2008], which were used to test their algorithm for mixed integer conic quadratic programming problems. We apply our approach to these instances originating from real-world stock market data and use them to calibrate the parameter choices for our method. Our results are vastly consistent with those presented in Vielma et al. [2008].

4.2.1 Instances

The authors of Vielma et al. [2008] test their algorithm on three different versions of the portfolio optimization problem, which they denote by *classical*, *shortfall*, and *robust* instances. The first version – classical – maximizes the expected return of the portfolio while keeping its risk below a certain predefined threshold. The shortfall instances differ in the risk measure used, while the robust instances consider the returns as uncertain. We use the classical instances, in which the risk is bounded by a second-order cone constraint, to evaluate the consistency of our results to theirs and to calibrate the parameters of our method. The problem is formulated as the following mathematical program:

$$\begin{array}{lrcll} \max & a^T y & & & \\ \text{s.t.} & \|Q^{1/2} y\|_2 & \leq & \sigma & \\ & y_i & \leq & x_i & \forall i \in \{1, \ldots, n\} \\ & \sum_{i=1}^{n} x_i & \leq & K & \\ & \sum_{i=1}^{n} y_i & = & 1 & \\ & x & \in & \{0,1\}^n & \\ & y & \in & [0,1]^n, & \end{array} \tag{13}$$

where $a \in \mathbb{R}^n$ is the vector of expected returns for a given number n of available assets. Matrix $Q^{1/2} \in \mathbb{R}^{n \times n}$ is the positive semidefinite matrix square root of the covariance matrix of these returns. The aim is then to invest into at most $K < n$ different assets to maximize the return of the entire portfolio while keeping its risk below a level of σ.

Variable x_i indicates whether asset $i = 1, \ldots, n$ is chosen to invest into, while variable y_i indicates the fraction of the total investment allocated to that asset. The first constraint restricts the risk of the investment to ensure that it is not higher than σ. It is formulated as a second-order cone constraint. The other constraints limit the number of different assets that can be held to at most K and enforce the investment of the entire budget into the portfolio.

The left-hand side of the second-order cone risk constraint can be interpreted in the context of robust optimization. It can be seen as the security term arising as the optimal value of robustification subproblem (11), which ensures that the solution remains feasible under all possible scenarios within a given ellipsoidal uncertainty set. The latter is defined by covariance matrix Q. In this interpretation, Problem (13) is the corresponding SOCP robust counterpart (10). To fit the problem into our computational framework, we use the Cholesky root of the covariance matrix instead of the matrix square root, which in theory makes no difference for the evaluation of the norm term.

In Vielma et al. [2008], results for instances with $n \in \{20, 30, 50, 100, 200\}$ were reported. As the case $n = 30$ is vastly similar to the case $n = 20$, and because the instances for $n = 200$ were still too difficult to solve within the time limit by any of the methods, we present results for their complete instance set with $n \in \{20, 50, 100\}$. These are 100 instances each for the first two cases and 10 instances for $n = 100$.

4.2.2 Computational Results for Small Instances

In the following, we summarize our results on instances with $n = 20$ assets, which we use to calibrate our framework, and describe the parameter settings arising from the observations on these instances.

Solution Times To get a proper idea of the complexity of these easier instances, we begin by stating the solution times for all algorithms under consideration in Table 1.

	QR		LR			
time [s]	IP	GL	$\epsilon = 1$	$\epsilon = 0.1$	$\epsilon = 0.05$	$\epsilon = 0.01$
min	0.10	0.03	0.01	0.01	0.04	0.02
max	2.38	28.38	0.49	0.36	0.41	0.53
mean	0.43	2.05	0.14	0.15	0.19	0.23

Table 1: Solution time statistics in seconds for instances with $n = 20$ assets

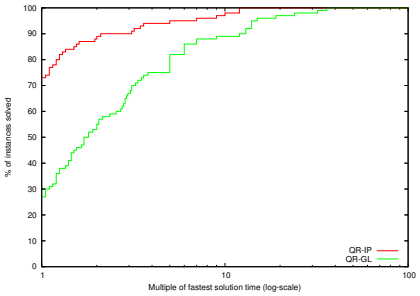

Figure 3: Performance profile for $n = 20$ assets and the two quadratic algorithms with solution time as performance metric

Gurobi has two built-in algorithms which can be applied to solve the SOC robust counterpart, whose solution times are found under "QR". The first is its interior-point solver, and the second its gradient linearization algorithm, denoted by "IP" and "GL", respectively. The abbreviation "LR" refers to our linear relaxation of the uncertainty set to approximate the SOC robust counterpart. We used four different approximation accuracies, namely $\epsilon \in \{0.01, 0.05, 0.1, 1\}$.

It can be seen that this subset of instances does not present a big problem for any of the algorithms, although certain differences are obvious. The interior-point algorithm performs much better than the gradient linearization. The average solution time of the former is 5 times faster, while in the worst case, the gradient linearization takes 14 times longer.

This is supported by the performance plot in Figure 3, which gives a visualization of the solution times of the two Gurobi algorithms. Such a performance plot is to be read as follows. Depending on the performance metric, the horizontal axis shows the multiples of the given metric, here the multiples of the fastest solution time of an instance. For any multiple p on the horizontal axis and any method m, the value f plotted in the performance profile indicates the percentage of instances that were solved by m within p times the value of the performance metric required by the best method m^*. In this case, f is the percentage of instances for which the solution by method m took at most p times longer than solution by the fastest method. In particular, the intercepts of the plots with the f-axis (corresponding to $p = 1$) represent the percentage of all instances for which the respective method m performed best. A more detailed introduction to performance profiles can be found in Dolan and Moré [2002].

For example, in Figure 3 it becomes visible that the IP solver was faster on more than 70 % of all instances, and that the gradient linearization does not catch up even when allowing higher multiples of the fastest solution time. Note that in all our performance plots, the horizontal axis is *log*-scaled, as it is usually done.

However, from Table 1 it also becomes obvious that both Gurobi algorithms are dominated by any of our linear approximations of the robust counterpart. The mean solution time of even the finest approximation is only half of that of the IP algorithm. And the maximal solution time is at most one fourth of that of IP. We note in advance that the trend visible in the increasing mean solution time for a finer approximation becomes much more distinctive for many of the harder benchmark instances in this paper.

Parameter Settings As shown in Theorem 2.11, we can obtain a polyhedral approximation of the second-order cone with an a priori approximation accuracy of ϵ, which consists of $\mathcal{O}(n \log \frac{1}{\epsilon})$ variables and inequalities for dimension n. Obviously, there is a trade-off between a tighter polyhedral relaxation for smaller values of ϵ and a smaller linear system describing the approximation for bigger values of ϵ. Hence, we want to determine an ϵ which ensures good results in terms of small approximation errors while maintaining reasonable problem sizes, which has a major impact on computation times.

In Table 2, we show the minimum, maximum, and average error as well as the standard deviation for optimal solution of the polyhedral approximation in comparison to the exact optimal value for the instances with $n = 20$.

error [%]	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 1$
min	0.11	0.22	0.81	2.46
max	0.27	0.74	2.81	10.67
mean	0.19	0.48	1.70	6.80
std	0.03	0.10	0.44	1.66

Table 2: Approximation error in per cent for different a priori accuracies ϵ for $n = 20$ assets

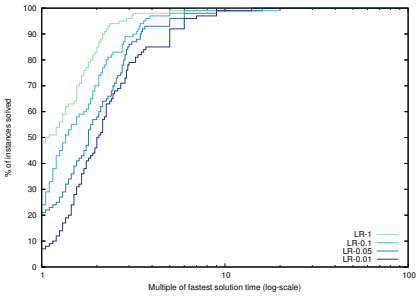

Figure 4: Performance profile for $n = 20$ assets and varying values of ϵ with solution time as performance metric

As expected, we see that the approximation error decreases monotonously in ϵ. These computations suggest to choose $\epsilon = 0.01$ or $\epsilon = 0.05$ as both yield approximation errors below 0.5 %. A performance plot of the solution times of the approximations is given in Figure 4. It is consistent with the figures in the Table 1 and shows a clear solution time advantage for $\epsilon = 0.05$ to $\epsilon = 0.01$. As this dominance becomes more evident for larger instances, up to the point of non-competitiveness of the latter, we choose $\epsilon = 0.05$ for all further comparisons in the paper. However, we note that a coarse approximation using $\epsilon = 1$ might be an efficient heuristic in many practical cases due to the fast solvability.

To cope with the size of the linear system arising for $\epsilon = 0.05$ in larger instances, the root node of the branch-and-bound tree is always solved using the barrier algorithm in the case of the linear approximation. All later nodes are solved via the dual simplex algorithm in order to profit from its warm-start capabilities. The two Gurobi algorithms are always run with standard parameter settings.

4.2.3 Computational Results for Larger Instances

The results presented in the following are for larger portfolio instances with $n = 50$ and $n = 100$ assets. We compare the performance of an approximation with $\epsilon = 0.05$ to that of exact solution via own two Gurobi's solver in Tables 3 and 4.

	QR		LR
time [s]	IP	GL	$\epsilon = 0.05$
min	0.35	∞	0.31
max	2048.11	∞	581.50
mean	91.59	∞	16.42

Table 3: Solution time statistics in seconds for $n = 50$ assets

	QR		LR
time [s]	IP	GL	$\epsilon = 0.05$
min	141.11	∞	88.89
max	∞	∞	∞
mean	2375.53	∞	1056.66

Table 4: Solution time statistics in seconds for $n = 100$ assets

At first sight, it becomes visible that Gurobi's gradient linearization is not competitive on these instances as it could not solve any of them within the time limit. This is in sharp contrast to the results obtained for our other benchmark sets, where it is the other way round. A reason for this behaviour might be that the portfolio problem involves the "robustification" of an otherwise empty row of the problem. This might lead to a high number of gradient cuts which are necessary to achieve a tight linear description of the quadratic constraint.

For $n = 50$ assets, both the IP solver and our linear approximation with accuracy $\epsilon = 0.05$ could solve all of the instances to optimality. The average solution time of our approximation is

more than 5 times faster than the time required by the exact IP solver. For the largest instances in this benchmark set, $n = 100$ assets, neither method could solve all the instances. The linear approximation still prevails on these instances by a factor of two.

We complement the results on the portfolio instances with by two more performance plots in Figures 5 and 6. They exclusively compare the IP solver to the linear approximation.

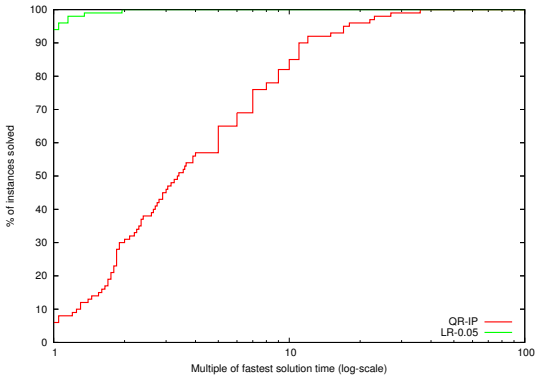

Figure 5: Performance profile for $\epsilon = 0.05$ and cone size $n = 50$ with solution time as performance metric

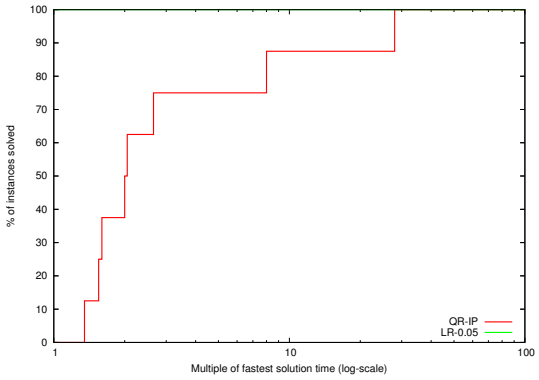

Figure 6: Performance profile for $\epsilon = 0.05$ and cone size $n = 100$ with solution time as performance metric

In the case $n = 50$, almost 95 % of all instances were solved fastest by our approximation and it took large multiples of time for the IP algorithm to catch up. In the case $n = 100$, all solvable instances were solved fastest by our method. The conclusion on this set of benchmark instances is that the linear approximation is well suited to obtain high quality solutions within relatively short time in the context of portfolio-like optimization problems.

4.3 MIPLIB Case Study

For the MIPLIB benchmark, we chose a subset of the instances, namely those MIP problems which are solvable within one hour using a commercial solver, as specified on its homepage. We left out the more difficult instances, as slight insight is to be gained from robustifying unsolvable problems. Furthermore, instances *harp2* and *mspp16* had to be excluded due to numerical instability. The problems in this library arise from a wide variety of applications and mostly cannot be clearly assigned to certain problem classes. Having a wide variety of instances from different sources, we can test our approach on a broad range of practically relevant problem types. Since MIPLIB is not a library of robust optimization problems, we needed to specify uncertainty sets against which we robustified the nominal instances. This is described in more detail in the following section.

4.3.1 Instances

In order to obtain a benchmark set for our robustification approach based on MIPLIB instances, we adopted an approach similar to that in Ben-Tal and Nemirovski [2000] to choose the uncertainty sets for the nominal instances: a coefficient shall be *uncertain* if its encoding length exceeds a certain threshold. This follows the assumption that coefficients with large encoding length are more likely to represent estimated or measured values in contrast to coefficients determining the structure of the problem. For our computations, we chose that (arbitrary) threshold to be equal to 9 (the encoding length of 32), which led to uncertainty sets of a reasonable dimension while being relatively sure that the coefficients determining the structure of the problem are not touched. We hasten to stress that this choice does not necessarily reflect the actual intention of the (nominal) model: we consider the MIPLIB study as a proof of concept of

our polyhedral relaxation framework rather than a case study on the robustification paradigm itself.

The next step is the definition of the uncertainty sets. As we saw before, the robust optimization framework can be applied constraint-wise, i.e., we can restrict ourselves to the case of a separate uncertainty set for each constraint. We call a constraint *uncertain*, if it contains at least one uncertain coefficient. As the exact quadratic robust counterpart and our polyhedral approximation are equivalent for constraints containing only one uncertain coefficient, we considered only those instances which have at least one constraint with two or more uncertain coefficients. Furthermore, we required at least 10 coefficients of the whole instance to be uncertain to consider it to interesting enough. To be able to control the difficulty of the problems, we introduced an upper bound u on the number of uncertain coefficients per constraint to protect the solutions against. We then added the first u uncertain coefficients of each constraint to its uncertainty set. For this case study, we chose $u \in \{10, 200\}$.

We further assume that each uncertain coefficients a_{ij} of an uncertain contraint $a^T x \leq b$ has a standard deviation $\delta = 0.01$ around its expected nominal value $\hat{a}_i$. We obtain a robust formulation of that constraint of the following general form

$$\hat{a}^T x + \Phi^{-1}(\alpha)\sqrt{x^T C x} \leq b,$$

where Φ is the cumulative distribution function of the normal distribution of the coefficient vector a and α is the maximal allowed probability of a constraint violation due to data perturbations. The latter are discribed by the covariance matrix C.

From the assumptions above, it follows that the variances of the uncertain coefficients are $Var(\hat{a}_i) = \delta^2 \cdot \hat{a}_i$. As we did not consider correlations between uncertain coefficients for this benchmark set, the covariance matrix C defining the ellipsoidal uncertainty set is diagonal. For the protection level, we chose a value of three times the standard deviation of the coefficients, which means $\Phi^{-1}(\alpha) =: \Omega = 3$, to avoid the time-consuming evaluation of the α-quantiles of the multivariate normal distribution.

4.3.2 Computational Results for MIPLIB instances

In the following, we report our results for the MIPLIB benchmark set comparing the approximate linearization approach against an exact solution via Gurobi. First of all, preliminary computations showed that Gurobi's IP solver was not competitive on these instances - hardly any of them were solved to optimality. The algorithm was vastly outperformed by Gurobi's exact gradient linearization in contrast to the results on the portfolio problems. This applies to all subsequent computations in the paper. Therefore, we present a comparison of the gradient linearization of the quadratic robust counterpart, denoted in short by "QR", and the linear approximation with $\epsilon = 0.05$, denoted by "LR-0.05". This notation is kept for the rest of the paper.

Performance for up to $u = 10$ Uncertain Coefficients The first benchmark set comprises those robustified MIPLIB instances with up to $u = 10$ uncertain coefficients per constraint; this can be considered as the case with sparse uncertainty. The approximation requires the introduction of new constraints and variables, which significantly increases the problem size. On the other hand, the quadratic model had to be slightly reformulated to enable Gurobi to recognize its SOCP structure. Therefore, there is an increase in size relative to the nominal problem in both cases. Tables 5 and 6 show by which factors the number of variables and constraints the quadratic model and the linear approximation increase on average compared to the nominal problem formulation. This comparison is for the figures after presolve to highlight the robustification impact on the main problem structure and omit modeling issues. Note that only instances that were solved by both methods within the time limit of one hour are considered in this statistic. As expected, the problem sizes increase by quite a large factor especially for the linearized formulation, which is obvious since our formulations is static and approximates the

var-increase	QR	LR-0.05
min	0.07	0.07
max	5.78	31.28
mean	0.92	2.31

Table 5: Variable increase factor after presolve for cone size $u = 10$

cons-increase	QR	LR-0.05
min	0.04	0.04
max	10.69	41.67
mean	1.75	5.13

Table 6: Constraint increase factor after presolve for cone size $u = 10$

second-order cone equally well everywhere. Thus, a fixed number of variables and constraints is added to the problem for a given a priori accurary ϵ for each second-order cone constraint. In contrast, the adaptive linearization, refines the formulation only where necessary.

In the following, we analyze the computational performance of the two approaches. Note that for instances that were solved to optimality within the given time limit of one hour by at least one of the two methods, the performance metric is given by the solution time. For the other instances, we chose the remaining optimality gap as performance metric. In this case, a performance curve not attaining 100 % for any multiple on the horizontal axis is due to the instances for which either no feasible solution or no finite bound was obtained. Figures 7 and 8 show the respective performance profiles for $\epsilon = 0.05$ and up to $u = 10$ uncertain coefficients. Interestingly, our results show that even though the relaxed linear formulation is much larger than the exact quadratic formulation, the performance of the approximation outperforms Gurobi's approach. To explore this result further, Tables 7 and 8 show the average solution times and

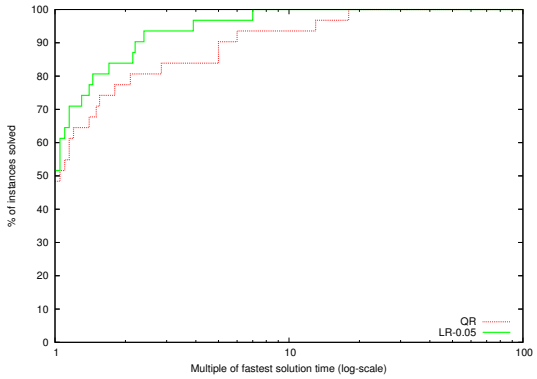

Figure 7: Performance profile for $\epsilon = 0.05$ and cone size $u = 10$ with runtime as performance metric

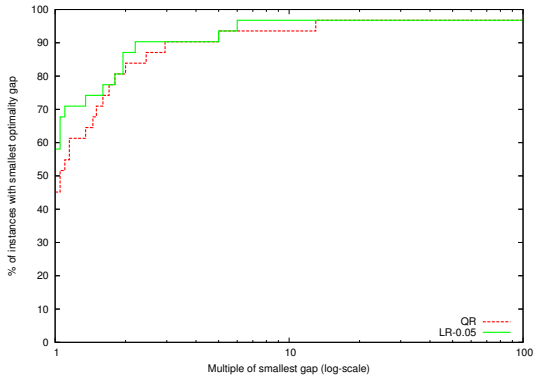

Figure 8: Performance profile for $\epsilon = 0.05$ and cone size $u = 10$ with gap as performance metric

the average number of iterations per branch-and-bound node, each comparing the quadratic model with our approximation. In Table 7, we see that the quadratic formulation can solve significantly more nodes per second compared to the linear approximation. It suggests that the nodes that have to be solved for the approximation are harder. This statement is supported by the fact that the number of simplex iterations per node yields a similar indication, as can be seen in Table 8. For the approximate method, the first value states the number of simplex iterations per branch-and-bound-node and the second value states the number of iterations of the barrier solver at the root node.

As a consequence of these statistics, we find that the nodes become harder to solve for the relaxation-based model. However, Figures 7 and 8 indicate a better performance of the relaxation approach regardless. This is in line with the fact that the total number of nodes needed by the relaxation is significantly smaller, as can be observed in Table 9. Consequently, the branch and bound nodes become harder on the one hand, but the total number of branching nodes decreases, hence the performance of the linear relaxation is better. We suspect that the relaxation approach extracts more information per node.

nodes/s	QR	LR-0.05
min	0.01	0.01
max	12583.33	10622.58
mean	1264.59	871.09

Table 7: Nodes per second for up to $u = 10$ uncertain variables per constraint

it./node	QR	LR-0.05
min	1.78	3.56/13.00
max	71978.00	54057.00/376.00
mean	5142.97	6137.72/75.86

Table 8: Number of iterations per node for up to $u = 10$ uncertain variables per constraint

nodes	QR	LR-0.05
min	1.00	1.00
max	45300001.00	18004501.00
mean	3099852.96	967741.11

Table 9: Number of nodes for up to $u = 10$ uncertain variables per constraint

var-increase	QR	LR-0.05
min	0.08	0.09
max	21.53	140.55
mean	1.84	8.50

Table 10: Variable increase factor after presolve for cone size $u = 200$

cons-increase	QR	LR-0.05
min	0.08	0.09
max	33.89	151.68
mean	3.46	13.05

Table 11: Constraint increase factor after presolve for cone size $u = 200$

Performance for up to $u = 200$ Uncertain Coefficients Now, we present the results for the same MIPLIB instances as before with up to $u = 200$ uncertain coefficients per constraint, which involves the approximation of higher-dimensional second-order cones. We undertake the same analysis as in the previous paragraph. Tables 10 and 11 depict the statistics on the minimal, maximal, and average increase factor of the problem size after presolve for the robust counterpart compared to the nominal model. We see that the polyhedral relaxation incurs by far the bigger growth in problem size, which can be explained by the static relaxation of the second-order cone as already described in the previous section.

Figures 9 and 10 show the performance profiles for $\epsilon = 0.05$ and up to $u = 200$ uncertain coefficients per constraint; the former using the solution time as performance metric for the solvable instances, the latter using the optimality gap for the unsolvable instances. We see that the polyhedral approximation approach performs comparably to the Gurobi's gradient linearization, however Gurobi's performance is better on the solvable instances. For the unsolvable instances, the polyhedral approximation tends to be better in closing the optimality gap within the time limit. To explain this performance, we consider the statistics for the solu-

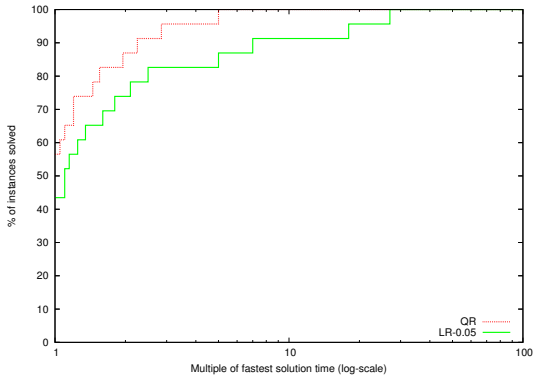

Figure 9: Performance profile for cone size $u = 200$ and runtime as performance metric

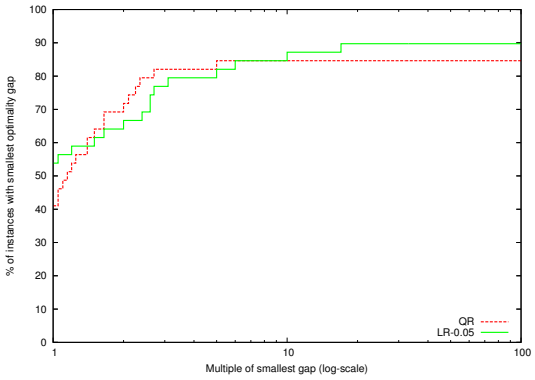

Figure 10: Performance profile for cone size $u = 200$ and optimality gap as performance metric

tion process as in the previous section. From Table 12, we see that the average number of nodes solved per second is again higher for the quadratic solver than for the linear approach. This is due to the significantly higher increase in problem size for the latter. Table 13 yields a similar finding for the number of iterations per branch-and-bound node. Hence, the conclusion is that as before the nodes are harder to solve in the case of our approach. We consider the total num-

nodes/s	QR	LR-0.05
min	0.01	0.01
max	2648.57	2261.29
mean	237.46	156.62

Table 12: Nodes per second for up to $u = 200$ uncertain variables per constraint

it./node	QR	LR-0.05
min	11.63	19.04/28.00
max	89460.00	111047.00/376.00
mean	7960.43	10921.33/131.00

Table 13: Number of iterations per node for up to $u = 200$ uncertain variables per constraint

nodes	QR	LR-0.05
min	1.00	1.00
max	235497.00	169681.00
mean	17700.50	11356.60

Table 14: Number of nodes for up to $u = 200$ uncertain variables per constraint

ber of nodes solved to get an idea of the overall performance of the algorithm (see Table 14). As in the case of sparse uncertainty, we see that the number of nodes is the smaller for our approach.

From the previous section, we learned that the increased difficulty of the branching nodes is partially offset by the decreased total number of branching nodes. However in the case of large cones, we find that this trade-off between the difficulty of the branch-and-bound nodes and the total number of nodes becomes worse for larger cones, which explains why Gurobi's relative performance improves (see Figures 9 and 10). Consequently, the biggest effort has to be put into keeping this trade-off of hardness and number of branch and bound nodes effective. In this case, this can be done by introducing further separation frameworks in order to reduce the increase in problem size and hence achieve better performance for the single branching nodes. Comparing both cases of sparse uncertainty and dense uncertainty, the biggest difference is the higher complexity, which results from the homogeneous approximation of the whole second-order cone.

To complete the results on this benchmark set, we give an overview of the approximation accuracies attainable by the four different choices of $\epsilon \in \{0.01, 0.05, 0.1, 1\}$ over the instances with $u = 200$ solvable to optimality. They are stated in Table 15. It becomes obvious that the

error [%]	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 1$
min	0.0000	0.0000	0.0000	0.0000
max	0.0016	0.0036	0.0348	1.0645
mean	0.0002	0.0004	0.0031	0.0652
std	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	0.0024

Table 15: Approximation error in % for different a priori accuracies ϵ and cone size $u = 200$. We keep a higher precision here as the differences are below our usual threshold of two decimals

linear relaxations all obtain very good approximation results. They surpass those stated for the Portfolio instances by orders of magnitude, which is most probably explained by the smaller impact of the quadratic security term in comparison to the nominal part of an uncertain constraint. This finding suggests that to avoid the complexity of solving the tight approximations for $\epsilon = 0.05$, it might be a viable alternative to use a coarser approximation, to obtain a good robust solution in shorter time. Even in the case of $\epsilon = 1$, the approximation error stays in the order of at most 1 %.

4.4 The Robust Knapsack Problem

This testset as well as the following one consists of instances with a specific problem structure. We start with instances representing robust multiple knapsack problems.

4.4.1 Instances

We consider the following knapsack problem with a multiple knapsack constraints:

$$\begin{array}{lrcl} \max & c^T x & & \\ \text{s.t.} & Wx & \leq & b \\ & x & \in & \{0,1\}^n, \end{array}$$

where $c \in \mathbb{R}^n$, $W \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. The weights $w_{ij} \in \{20, 21, \ldots, 29\}$ and the benefits $c_i \in \{16, 17, \ldots, 77\}$ are chosen randomly using the Boost Random Number Library (cf. Boost C++ Libraries [2013]). The knapsack capacity vector b is chosen as $b_i = 20n$, $i = 1, \ldots m$. For this testset, we set the number of variables to either 10 or to 200, where all of them are considered uncertain. The number of constraints is chosen from the set $\{1, 10, 100, 1000, 10000\}$. The uncertainty shall exclusively lie in the weights of the items, which are assumed to deviate at most 10 % around the nominal value independently, i.e., $\delta = 0.1$. The protection level was at $\Omega = 3$. For each combination we created 10 instances randomly.

4.4.2 Computational Results

The comparison again features Gurobi's exact gradient linearization and our linear approximation with $\epsilon = 0.05$. Figures 11 and 12 show the corresponding results.

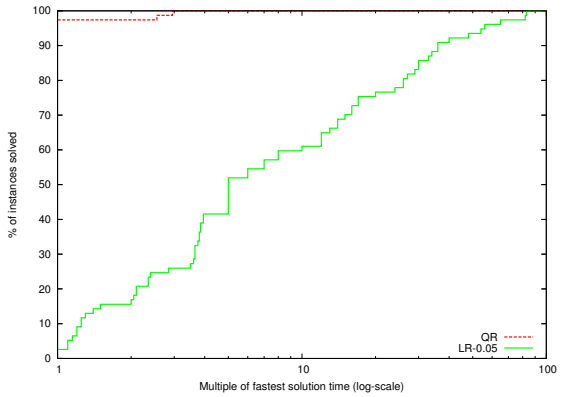

Figure 11: Performance profile for the robust knapsack problem and runtime as performance metric

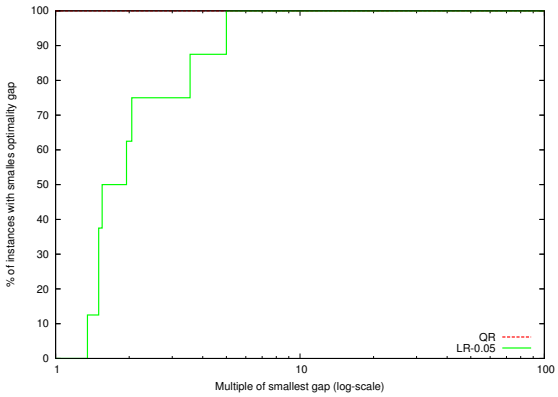

Figure 12: Performance profile for the robust knapsack problem and optimality gap as performance metric

The diagram in Figure 11 shows the solution time performance for the instances which could be solved to optimality within the time limit, while Figure 12 shows performance with respect to the remaining optimality gap for the unsolvable instances. In summary, we can say that for the knapsack problems, the quadratic robust approach performs much better considering both the solution time (better performance on more than 95 % of the instances) and the optimality gap (lower gap for 100 % of the instances). This finding is especially representative for those instances with a multiple number of knapsack constraints (see also Table 16).

stat	QR	LR-0.05
cons-increase	106.00	496.08
vars-increase	1534.26	20481.25
nodes/s	267.32	116.53
it./node	1205.92	8050.51
no. of nodes nodes	283.42	1040.13

Table 16: Solution statistics for the knapsack instances

Again, the linear approximation of the robust counterpart exhibits a much higher growth in problem size on average than the quadratic formulation used by Gurobi. This results in significantly harder to solve LP relaxations. Unlike the finding in the MIPLIB study, the approximation this time also requires the solution of a higher number of branch-and-bound-nodes in

total, explaining the results in Figures 11 and 12. One possible reason for this computational behavior could be that our approximate framework prevents Gurobi from fully exploiting the knapsack structure of the problem. However, computational tests without the use of cutting planes yield a similar result. Furthermore, the substitution of the binary quadratic terms x_i^2 in the quadratic robust counterpart by x_i led to worse bounds and hence did not improve the results. It is not clear where this strong unexpected difference in performance arises from. We presume various presolving methods that are available to Gurobi for its own formulation to be the reason.

4.5 A Network Design Example

In this section, we consider another class of combinatorial problem instances, namely network design problems. Given a directed graph $G = (V, A)$, the objective is to route a given demand across the graph which is represented as a function $d : R \to \mathbb{R}_+$, where $R \subseteq \{r \in V^2 \mid r_1 \neq r_2\}$ is the set of demand relations. For such a demand relation $r \in R$, we call r_1 its origin and r_2 its destination. Besides flow conservation, the main constraint is to respect the capacities of the arcs. We are given a function $C : A \to \mathbb{R}_+$ indicating their base capacities. Where these base capacities are not sufficient to route the demand, we are to extend it in integral multiples of the extension capacity C, which is equal for each arc and comes at a price according to the function $c : A \to \mathbb{R}_+$ per unit. We need to route all the demand at a minimal extension cost for the arcs of the network (routing costs could also be considered, but are omitted here). This problem can be formulated as the following MIP:

$$
\begin{aligned}
\min \quad & \sum_{a \in A} c_a x_a \\
\text{s.t.} \quad & \sum_{a=(i,j)\in A} y_a^r - \sum_{a=(j,i)\in A} y_a^r = \begin{cases} 1, & \text{if } i = r_1 \\ -1, & \text{if } i = r_2 \\ 0, & \text{otherwise} \end{cases} && (\forall i \in V) \\
& \sum_{r \in R} d_r y_a^r \leq C_{ij} + C \cdot x_a && (\forall a \in A) \\
& x \in \mathbb{Z}_+^{|A|} \\
& y \in [0,1]^{|A|}.
\end{aligned}
$$

We choose variables x_a, $a \in A$ as the extension decisions indicating in which multiples of extra capacity to install. Variables y_a^r, $r \in R$ and $a \in A$, represent the flow of relation r along arc a giving us a certificate for the validity of our extension decisions. The MIP then gives us a least-cost plan for the extension of the network such that all demand can be routed and all are capacities are respected.

This is a very convenient and compact model in the situation when we are sure about the demand we would like to route. But as soon as there is some uncertainty in the demand, we can no longer be sure that its solution yields a valid extension plan. Of courde, we could demand some fixed security buffer and just install some extra capacities on top of those calculated. But this could result in unnecessarily high costs however, so we utilize the robust optimization paradigm.

4.5.1 Instances

Our test instances emanate from the SNDlib benchmark set of survivable network design instances which we adapted to fit the above model. We adopted the network topologies, base capacities for the arcs, and the multi-commodity demand specifications of these instances. In the SNDlib instances, there are several expansion modules per instance which are available for each arc. The per-unit cost for the installation of a module is arc-specific. To decrease the complexity of the nominal problem, we allow only one expansion module with equal capacity value C for each arc. We chose it as the average of all expansion capacities occurring in an instance. The per-unit cost for that module on a certain arc was chosen as the mean cost of the

original modules on that arc. To define the uncertainty sets for the demand vectors, we chose the standard deviation δ_i of each demand d_i with $i \in \{1, 2, \ldots, u\}$ to be 10 % of the nominal value d_i. The ellipsoidal uncertainty sets then consist of all scenarios in the ellipsoid defined by this diagonal variance matrix of the uncertain demand vectors (i.e., $\Omega = 1$). For the test sets with correlated uncertainties, we chose a covariance matrix whose lower triangle Cholesky factor $C = (c_{ij})$ has the entries $c_{ij} = \delta_i$ for $i = j$, $c_{ij} = (-1)^{i+j} \cdot 10^{-4} \cdot \delta_i \cdot \delta_j$ for $i < j$ and $c_{ij} = 0$ otherwise. The constraints to be protected against the uncertainty are the capacity constraints.

4.5.2 Computational Results

The computational results for the network design instances will be summarized in the following. The parameter choices are the same as for the previous two testsets. This set contains the 24 SNDlib instances with different choices on the number of uncertain demands, each once uncorrelated and once correlated as described above. The number of uncertain coefficients per constraint is again bounded by $u \in \{10, 200\}$.

Results for Uncorrelated Uncertainties For our first test run, we chose the number of uncertain demands to be $u = 10$. In Table 17, we show the results for those 8 instances, which were solved to optimality within the time limit via at least one of the two methods. For the other 16 instances which were not solved to optimality using either formulation, Figure 13 shows the performance diagram with the remaining optimality gap as performance measure.

	QR		LR-0.05	
instance	time [s]	gap [%]	time [s]	gap [%]
abilene	1362.06	0.00	302.68	0.00
dfn-bwin	1465.17	0.00	-	6.41
di-yuan	788.89	0.00	1539.05	0.00
nobel-germany	522.58	0.00	78.09	0.00
nobel-us	394.19	0.00	664.23	0.00
pdh	3.44	0.00	24.43	0.00
polska	484.67	0.00	238.69	0.00
ta2	646.16	0.00	-	50.61

Table 17: Optimal solution times for $u = 10$ uncorrelated uncertainties

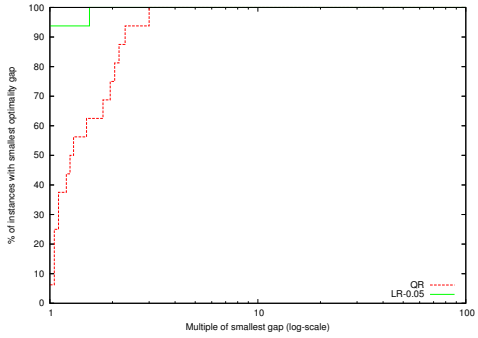

Figure 13: Gaps of unsolved instances for $u = 10$ uncorrelated uncertainties

Table 17 shows that the linear approximation with accuracy $\epsilon = 0.05$ is the better formulation for three of eight instances solved within the time limit. These instances are based on small to medium-sized graphs with a small number of demands to be routed. The solution times differ from a seven times faster solution of the linear formulation up to a seven times faster solution via the quadratic formulation, which makes it hard to predict the winner in advance. Figure 13 shows a much clearer picture for the remaining optimality gaps for the instances not solved within the time limit in either formulation. Our linear formulation yields the lowest optimality gap for 90 % of the instances and in the worst-case its gap is a little more than twice as high as for the quadratic model. This is due to the fact that in most of the unsolved instances the approximated robust counterpart is able to find the better solution within the time given. In a majority of them, it has the better bound at this point as well.

In Table 18, we summarize several statistics on the solution process for each formulation. Rows "cons-increase" and "row-increase" show the mean factor over all instances by which the number of constraints and variables grows, both after preprocessing. The increase in the number of constraints for the linearization is about 1.5 times higher than for the quadratic robust counterpart on average, and about 50 % higher in the number of variables. This naturally influences the difficulty of the nodes in the branch-and-bound tree. While for the quadratic

robust counterpart 231.13 nodes are solved per second on average, this figure is 176.06 for the approximation. On the other hand, we observed that the our approach usually needs to solve significantly less nodes in total to obtain an optimal solution, for this set of instances only half as many as the exact approach. This explains the much better performance on the harder unsolved instances.

stat	QR	LR-0.05
cons-increase	1.36	2.57
vars-increase	1.08	1.48
nodes/s	231.13	176.03
it./node	202.98	260.45
no. of nodes	142230.00	77845.17

Table 18: Solution statistics for $u = 10$ uncorrelated uncertainties

Our second test set comprises the same nominal instances as above, this time with up to 200 uncertain demands. All other parameters remain the same. In Table 19, we show the solution times and remaining optimality gaps of those 5 instances solved within 3600 seconds with at least one of the two formulations. This time, the quadratic model performs better in four of the five solved instances. Contrary to the results on those easier instances, the performance of the linearization is still better for those 19 instances not solvable to optimality within the time limit as can be seen in the diagram in Figure 14. It prevails for 52 % of the instances and is the only formulation for which a feasible solution can be found for all instances within the time limit. Our results show that our approach found the better primal solution within the time limit for the majority of those instances, the quadratic robust counterpart found the better lower bound in most cases.

	QR		LR-0.05	
instance	time [s]	gap [%]	time [s]	gap [%]
di-yuan	1113.91	0.00	-	0.54
nobel-germany	-	5.42	3186.46	0.00
pdh	6.77	0.00	57.78	0.00
polska	3539.18	0.00	-	4.54
ta2	1222.98	0.00	-	85.92

Table 19: Optimal solution times for $u = 200$ uncorrelated uncertainties

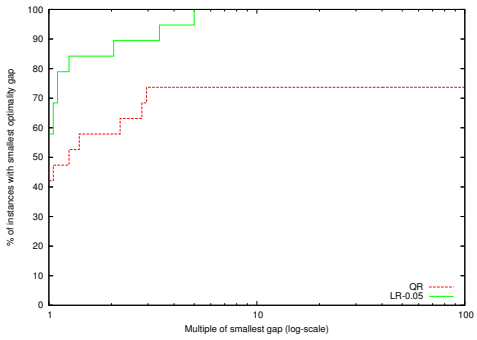

Figure 14: Gaps of unsolved instances for $u = 200$ uncorrelated uncertainties

From Table 20, we see that for this testset the growth in size is much bigger for the linearization as for the quadratic robust counterpart. This also applies to the computational effort for the solution of the problems. As an example the average number of nodes solved per second for instance *pdh* is 23.75 for the linearized version compared to 168.85 for the quadratic problem. In the former case, it takes 278.60 iterations on average to solve each node, while it takes 65.07 iterations for the latter. Still, the linear relaxations obtains lower gaps at the end of the time limit for a majority of instances. This again can be explained by its ability to find better solutions (or any at all) early in the solution process, which results in a better overall performance of the linearized robust counterpart.

stat	QR	LR-0.05
cons-increase	3.41	13.39
vars-increase	1.55	5.23

Table 20: Statistics for $u = 200$

Results for Correlated Uncertainties For our last two test sets, we generated uncertainty sets corresponding to correlated demand uncertainties, as described above. This is to demonstrate the ability of this approach to incorporate exact correlation data of the parameters while working with polyhedral uncertainty sets, which is an advantage over the choice of other polyhedral uncertainty concepts. We again perform these tests for different dimensions of the uncertainty sets and thus different dimensions of the cones to approximate.

	QR		LR-0.05	
instance	time [s]	gap [%]	time [s]	gap [%]
abilene	793.65	0.00	433.88	0.00
di-yuan	12.46	0.00	1815.99	0.00
janos-us	-	0.50	20.28	0.00
nobel-germany	62.61	0.00	139.23	0.00
nobel-us	577.91	0.00	657.56	0.00
pdh	5.30	0.00	48.97	0.00
polska	317.13	0.00	645.80	0.00
ta2	430.62	0.00	-	79.48

Table 21: Optimal solution times for $u = 10$ correlated uncertainties

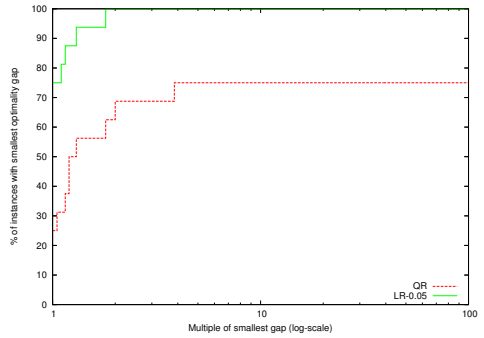

Figure 15: Gaps of unsolved instances for $u = 10$ correlated uncertainties

We start with 10 uncertain demands per instance. In Table 21, we list results for those instances which were solved to optimality by at least one of the two formulations. It shows that the linearized formulation is solved faster for 3 of 8 instances solvable to optimality, the other 5 instances are solved faster in the quadratic formulation. For those 16 instances not solved within the time limit, Figure 15 shows the performance profile of the two formulations with respect to the remaining gap. As the linear formulation is the only one to find feasible solutions to all the instances, it significantly outperforms the quadratic one on those more difficult instances.

	QR		LR-0.05	
instance	time [s]	gap [%]	time [s]	gap [%]
dfn-gwin	3095.53	0.00	-	85.6
di-yuan	29.72	0.00	-	3.25
pdh	8.49	0.00	122.67	0.00
sun	732.97	0.00	-	8.04
ta2	2990.82	0.00	-	-

Table 22: Optimal solution times for $u = 200$ correlated uncertainties

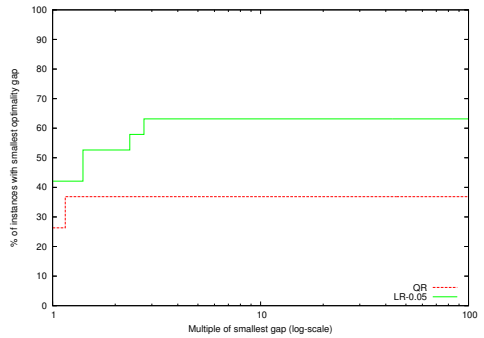

Figure 16: Gaps of unsolved instances for $u = 200$ correlated uncertainties

We also repeated this test with 200 uncertain demands per instance. The results are shown in Table 22 and Figure 16. While the quadratic robust counterpart now solves all 5 solvable instances faster than the approximation, the latter performs better on those instances not solved within the time limit and is able to find feasible solutions for a higher percentage of those in-

stances, resulting in an overall better performance on the more difficult instances. We remark that the mean solution times of the branch-and-bound nodes and the mean number of simplex iterations to solve them behave similar as in the uncorrelated case when going from 10 uncertain demands to 200 uncertain demands. The increase in node complexity is significantly greater for the linearized robust counterpart due to its much larger growth in size. For example, instance *ta2* with 200 correlated uncertainties leads to a linearized robust counterpart with 361875 constraints and 775784 variables after preprocessing, compared to 161841 constraints and 440552 variables as well as 216 quadratic constraints in the quadratic robust counterpart. Choosing a coarser approximation might again be a suitable alternative to obtain good solutions in shorter time.

5 Conclusions and Future Work

Ellipsoidal uncertainty sets are the most frequently chosen uncertainty sets, inter alia, because they are most suitable to represent information on the distribution of the uncertain parameters. Nevertheless, they require a second-order cone problem to be solved as robust counterpart, which is difficult to solve computationally. Therefore, several approaches for robust optimization using polyhedral uncertainty sets exist (e.g., Bertsimas and Sim [2003]). They lead to robust counterparts which are easier to solve, but lack the potential to incorporate exact information on the variance and covariance of the uncertain coefficients. Our approach aims at combining the best of both worlds by approximating ellipsoidal uncertainty sets polyhedrally using a compact linear formulation.

In the computational tests for our approach, we demonstrated that our formulation outperforms the exact quadratic model on many test instances. It performs highly superior on the portfolio instance from Vielma et al. [2008], confirming the observations stated therein. On the other instances it is vastly competitive with the exact gradient linearization in Gurobi. Furthermore, the approach seems to be more suited for finding feasible solutions early in the solution process.

A problem to cope with are the large linear systems arising from linearization of the second-order cone, which make the node relaxations in the branch-and-bound process difficult to solve. To improve the performance of our approach, it is thus a promising direction for future research either to transform the static approximation with a-priori accuracy into an adaptive approximation of the uncertainty sets or to modify the static approximation such that it describes only the relevant regions of the uncertainty sets with a higher accuracy. This could drastically decrease the sizes of the linear robust counterparts to be solved leading to much smaller solution times. It is not obvious how to achieve this, however.

Most striking is the high quality of the approximation results, which allows to obtain very good solutions already for a coarse approximation of the ellipsoidal uncertainty sets. Even the coarsest possible selection $\epsilon = 1$ yields an approximation error only in the order of 1 %. To use a coarse approximation of the uncertainty set as a heuristic for solving the SOCP robust counterpart thus seems a very worthwhile consideration.

References

A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming A*, 88:411–424, 2000.

A. Ben-Tal and A. Nemirovski. On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, 26:193–205, 2001.

A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

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

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

D. Bertsimas and M. Sim. Tractable approximations to robust conic optimization problems. *Mathematical Programming B*, 107:5–36, 2006.

D. Bertsimas, D. Pachamanovab, and M. Sim. Robust linear optimization under general norms. *Operations Research Letters*, 32:510–516, 2004.

D. Bertsimas, D. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464 – 501, 2011.

J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical Optimization – Theoretical and Practical Aspects*. Springer, 2009.

Boost C++ Libraries. Boost random number library. `http://www.boost.org/doc/libs/1_38_0/libs/random`, 2013.

M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *4OR*, 8(1):1 – 48, 2010.

E. Dolan and J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming A*, 91(2):201 – 213, 2002.

S. Fiorini, T. Rothvoß, and H. Tiwary. Extended formulations for polygons. *Discrete Computational Geometry*, 48(3):658–668, 2012.

M. Gay. Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Bulletin*, 13:10–12, 1985. Data available at `http://www.netlib.org/netlib/lp`.

F. Glineur. Computational experiments with a linear approximation of second-order cone optimization. Image Technical Report 001, Faculté Polytechnique de Mons, 2000.

F. Glineur. Conic optimization: an elegenat framework for convex optimization. *Belgian Journal of Operations Research, Statistics and Computer Science*, 41(1 - 2):5 – 28, 2001.

Gurobi Optimization, Inc. Gurobi optimizer reference manual. `http://www.gurobi.com`, 2013.

V. Kaibel. Extended formulations in combinatorial optimization. *Optima*, 85:2–7, 2011.

V. Kaibel and K. Pashkovich. Constructing extended formulations from reflection relations. In *Proceedings of IPCO 2011*, pages 287–300, 2011.

T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. Bixby, E. Danna, G. Gamrath, A. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. Steffy, and K. Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011.

S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, 2007.

J. Vielma, S. Ahmed, and G. L. Nemhauser. A lifted linear programming branch-and-bound algorithm for mixed integer conic quadratic programs. *INFORMS Journal on Computing*, 20 (3):438 – 450, 2008.