ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Master Thesis Operations Research And Quantitative Logistics

COMPARING TWO METHODS FOR ROBUSTIFICATION OF A SOCP AFFECTED BY SIMPLE ELLIPSOIDAL UNCERTAINTY

Author:
Niels Floris Leonardus in 't Veld
Student ID:
547817

Supervisor:
Dr. K.S. Postek

Second Assesor:
Dr. O. Kuryatnikova

May 1, 2021

Abstract

Robust optimization (RO) is a paradigm for solving optimization problems affected by uncertainty. In RO, two key methods are the robust counterpart (RC) method and the adversarial approach. We present these two methods applied to a Second Order Cone Programming (SOCP) perturbed by an ellipsoidal uncertainty set. The adversarial approach is based on two steps, the optimization- and pessimization step. First a SOCP is optimized for a finite set of uncertainties, then a pessimization for this optimal solution is found. The latter is a problem of maximizing a convex function, which has a hidden convexity structure. This makes it equivalent to minimizing a convex optimization problem. Such problems can be solved using the trust region subset problem. The RC method derives a deterministically equivalent problem, the robust counterpart. For a SOCP affected with ellipsoidal uncertainty the RC can be reformulated exactly to an explicit SDP reformulation, which is computationally tractable. Our aim is to thoroughly explain this reformulation. The size of the resulting SDP is a lot larger than the size of the original problem, making it inferior to the adversarial approach for large scale problems. Finally, in modest numerical experiments we show that the SDP-reformulation breaks down for large scale problems. Conversely, for small problems it out-preforms the adversarial approach.

Acknowledgements

First and foremost, I would like to use this opportunity to thank my thesis supervisor Krzysztof Postek for his guidance, bookkeeping skills and invaluable feedback sessions. He extended me the opportunity to use this thesis project to satisfy my interest in conic programming, a topic which caught my attention when it was touched upon briefly in one of his courses. Olga Kuryatnikova thank you for taking the time to be the second reader of my thesis. Next, I thank my family and friends for supporting me throughout this turbulent year. In particular, I would like to express my appreciation to Rianne 't Jong, for her support and linguistic feedback, and Coen De Jong for his friendly ear to my monologues about, convex functions, cones, matrices and so on.
To whomever it may concern, I hope you find this report enjoyable and informative.

Dordrecht, May 1, 2021

Niels Floris Leonardus in 't Veld

Contents

1 Introduction **3**

2 Theory **5**
- 2.1 Mathematical Concepts . . . 5
  - 2.1.1 Convex Analysis . . . 5
  - 2.1.2 Conic Mappings . . . 6

3 Mathematical Optimization **8**
- 3.1 Optimization Problems . . . 8
  - 3.1.1 Linear Optimization . . . 8
  - 3.1.2 Convex Optimization . . . 8
  - 3.1.3 Conic Optimization . . . 9
- 3.2 Uncertainty in Optimization . . . 10
  - 3.2.1 Robust Optimization . . . 10

4 Applying Robust Optimization **13**
- 4.1 The Adversarial Approach . . . 13
  - 4.1.1 Optimization-step . . . 14
  - 4.1.2 Pessimization-step . . . 14
  - 4.1.3 The Cutting Set Method . . . 14
- 4.2 The Robust Counterpart Method . . . 15
  - 4.2.1 Tractability Condition . . . 15
  - 4.2.2 The SDP reformulation . . . 16

5 Numerical Experiment **19**
- 5.1 Experimental Setup . . . 19
  - 5.1.1 Sampling . . . 19
  - 5.1.2 Averaging of the Experiments . . . 20
  - 5.1.3 Implementation Problems . . . 20
- 5.2 Results . . . 22
- 5.3 Summary . . . 23

6 Conclusion **25**

A Relevant Mathematical Concepts **26**
- A.1 The Subspaces of Skew-Symmetric Matrices . . . 26

Bibliography **28**

1 | Introduction

In the early 20th century the relatively new area of applied mathematics began to develop. The invention of modern-day computers ushered in a whole new field called (Mathematical) Optimization, focussing on selecting the best element out of a set subject to various criteria. These problems arise in most quantitative disciplines ranging from computer science and engineering to operations research and stock markets. In modern day life optimization problems are deeply embedded into our society.
Linear Programming (LP) is the most widely used class of optimization problems. It arises in all sorts of application and can be solved numerically in a very efficient way. Conic Programming (CP) is a generalization of LP, which is able to deal with more complexly constrained problems. Such constraints arise in various engineering problems, e.g. in signal processing or antenna design. In this thesis we will be focussing on Second Order Cone Programming (SOCP), i.e. problems with constraints in the following form

$$||Ax + b||_2 \leq c^\top x + d.$$

The theory behind CP is not nearly finished, many applications are still waiting to be discovered. A relatively newly emerging sub-field deals with optimization problems affected by uncertainty.
One of the key paradigms for solving such problems is Robust Optimization. Two methods will be discussed, the adversarial approach and the robust counterpart (RC) Method . The first method, is based on iteratively constructing a finite list of similar additional (deterministic) constraints. It 'cuts' away infeasible realizations and ensures that the optimal solution satisfies the original uncertain constraint. Whereas the RC method is based on mathematically deriving a deterministically equivalent constraint, called the robust counterpart. We will study both methods applied to a SOCP affected by ellipsoidal uncertainty. Such problems have constraints of the form

$$||A(z)x + b(z)||_2 \leq c(z)^\top x + d(z) \quad : ||z||_2 \leq r. \tag{1.1}$$

In the adversarial approach, we split up the constraint into two problems. An optimization problem in the form of minimizing the decision variables (x) for a given finite set of uncertainties. And the adversarial[1] problem of maximizing the uncertainty (z) for a given optimal solution x^*, i.e. selecting the worst possible realization. From now one we will call this the pessimization problem, since the goal is to find the the most pessimistic value for z. It turns out this problem has a hidden convexity structure which makes it efficiently solvable. In the RC method we will robustify the constraint directly. This will lead to a linear matrix inequality, which can be seen as an exact Positive Semi-definite Programming (SDP) reformulation. Finally, the computational advantages and disadvantages of both methods are evaluated by means of sampling. We end this section by giving a short motivational example.

[1]It is common to refer to the uncertainty as 'the adversary', synonymous to the enemy, because in worst-case analysis you assume it will always try to find the worst possible realization.

Example 1.0.1:
Consider a field where we need to place a signal towers as efficiently as possible, in order to get a 'decent' signal at every point. Mathematically we want to maximize the signal at every point in the entire field subject to constraining the signal-to-noise ratio. Let $g(x, z) \geq 0$ be the function of the signal itself and $||f(x, z)||$ be the norm of the noise. If we want to constraint the signal to noise ratio to be less then 10%, we would need constraints of the form

$$\frac{||f(x, z)||}{g(x, z)} \leq 0.1,$$

where f, g are bi-affine in (x, z). Notice, these constraints are not linear constraints, but they can be rewritten in the form of Constraint (1.1). If there was no uncertainty, i.e. $z = 0$, then the problem could be solved by any convex programming solver. But the dependence on z makes it difficult. There are decision variables x for which some realization of z would lead to

$$\frac{||f(x, z)||}{g(x, z)} \not\leq 0.1,$$

In order to find the best solution x^* we need to take this into consideration. Therefore, we need a way to 'find' the solution x^* that is robustly optimal, i.e. it is optimal and feasible for all realization of z. Robust Optimization, is a paradigm for deriving optimization problems that find robustly optimal solutions.

2 | Theory

In order to clearly explain the relevant methods, we need to digest some theory. This chapter contains a clear and concise overview of all the related definitions, lemmas and theorems. For the reader who has sufficient knowledge of Optimization (or in general Mathematics) we advice reading Section 2.1.2 and then immediately skip to Section 3.2.1.

2.1 Mathematical Concepts

2.1.1 Convex Analysis

For brevity, we will list the relevant definitions

- A **line** through x_1 and x_2 consists of points of the form

  $$y = \theta x_1 + (1-\theta)x_2, \quad \forall \theta \in \mathbb{R},$$

  restricting $\theta \in [0,1]$ leads to the **line segment** between x_1 and x_2.

- A set $A \in \mathbb{R}^n$ is called **affine** if the line through any two distinct points in A lies in A, i.e.

  $$\forall x_1, x_2 \in A \quad \forall \theta \in \mathbb{R} : \theta x_1 + (1-\theta)x_2 \in A$$

- A set $C \in \mathbb{R}^n$ is called **convex** if the line segment between any two distinct points in A lies in A, i.e.

  $$\forall x_1, x_2 \in C \quad \forall \theta \in [0,1] : \theta x_1 + (1-\theta)x_2 \in C$$

- A set $K \in \mathbb{R}^n$ is called a **cone**, if $\forall x \in K$ and $\theta \geq 0$ we have $\theta x \in K$. And it is called a **convex cone** if it is convex and a cone, i.e.

  $$\forall x_1, x_2 \in K \quad \forall \theta_1, \theta_2 \geq 0 : \theta_1 x_1 + \theta_2 x_2 \in K.$$

  it is **proper** if it satisfies the following, K is convex, closed[1], solid, i.e. $\operatorname{int} K \neq \emptyset$, and pointed, i.e. $x, -x \in K \Rightarrow x = 0$.

- Let $K \in \mathbb{R}^n$ be a cone, then the set $K^* = \{y \in \mathbb{R}^m | y^\top x \geq 0 \forall x \in K\}$ is called the dual cone of K.

CP revolves around proper cones, because they have convenient properties. They can be extended by direct products, i.e. for proper cones $K_1, \ldots, K_m$, the direct product $K_1 \times \cdots \times K_m$ is also a proper cone. And they are self-dual, i.e. $K^* = K$. These facts are the key reason why CP's can be solved efficiently.

[1]A set is called closed if its complement is open.

Next, we will touch on some important examples of proper cones:

- The nonnegative orthant (also called a LP-cone)

$$\mathbb{R}^n_+ = \{x \in \mathbb{R}^n : x \geq 0\}$$

- Lorentz or second-order or ice-cream cone (also called a SOCP-cone)

$$\mathcal{L}^{n+1} = \{(x,t) \in \mathbb{R}^{n+1} : ||x||_2 \leq t\}$$

- Positive semidefinite cone (also called a SDP-cone)

$$\mathbb{S}^n_+ = \{Z \in \mathbb{S}^n : u^T Z u \geq 0 \forall u \in \mathbb{R}^n\}$$

For the SOCP problems in question, the constraints are elements of a SOCP-cone, and we will show that its RC is a special type of matrix, which is an element of the SDP-cone.

2.1.2 Conic Mappings

To properly introduce the exact way the RC of Constraint (1.1) is constructed we need some seemingly incoherent complex mathematical concepts. We will state them in form of theorems and lemmas without providing proofs, because that would involve a complex linear algebra going beyond the scope of this thesis.
Starting with the fact that every linear mapping can be represented by a matrix. We begin with two definitions about mapping one cone into another, which give rise to special cones.

Definition 1. Let $K_1 \subset \mathbb{R}^m, K_2 \subset \mathbb{R}^n$ be proper cones. We call a linear map $\mathcal{B} : \mathbb{R}^m \to \mathbb{R}^n$ **K_1-to-K_2 positive** if $\mathcal{B}[K_1] \subset K_2$. The cone of K_1-to-K_2 positive mappings is itself a proper cone (of matrices) in $\mathbb{R}^{n \times m}$, called **the K_1-to-K_2 positive cone**. Moreover, if both K_1 and K_2 are Lorentz cones, then we call the linear mapping **Lorentz-positive**.

This definition means that if a linear mapping is K_1-to-K_2 positive, then it maps all the elements of cone K_1 into the cone K_2. Thus, for a linear mapping $\mathcal{B}$, it holds that

$$\mathcal{B}[K_1] \subset K_2 \iff \text{the linear map } \mathcal{B} \text{ is } K_1\text{-to-}K_2 \text{ positive}$$

Hildebrand's Theorem

R. Hildebrand proved that there exists an explicit representation for the cone of Lorentz-Positive $[n \times m]$-matrices [1]. Technically, he found an explicit representation for the set

$$P_{m,n} = \{A : \mathbb{R}^m \to \mathbb{R}^n | A[\mathcal{L}^m] \subset \mathcal{L}^n\},$$

i.e. the set of matrices mapping $\mathcal{L}^m$ into $\mathcal{L}^n$. The theorem is based upon the link between the SOCP-cone and SDP-cone, which can be stated in the following lemma

Lemma 1. $(x_1, x_2, \cdots, x_n) \in \mathcal{L}^n$ *if and only if*

$$\begin{pmatrix} x_n + x_1 & x_2 & \cdots & x_{n-1} \\ x_2 & x_n - x_1 & & \\ \vdots & & \ddots & \\ x_{n-1} & & & x_n - x_1 \end{pmatrix} \succeq 0,$$

where $A \succeq 0$ means that A is a symmetric positive definite matrix, i.e. $A \in \mathbb{S}^n_+$. The explicit representation relies heavily on the use of two slightly overwhelming definitions, namely

Definition 2. Define a linear mapping a $A \mapsto \mathcal{W}(A)$ from $\mathbb{R}^{n \times m}$, the space of real $[n \times m]$-matrices, into $\mathbb{S}^N$, the space of N-symmetric matrices, where $N = (n-1)(m-1)$, as follows.
For two vectors $u \in \mathbb{R}^n, v \in \mathbb{R}^m$ we have $\mathcal{W}[u, v] = W_n[u] \otimes W_m[v]$, where $\otimes$ is the Kronecker product, which is defined in appendix section A.1, and $W_k[u]$ is given by[2]

$$u \mapsto \begin{pmatrix} u_k + x_1 & u_2 & \cdots & u_{k-1} \\ u_2 & u_k - u_1 & & \\ \vdots & & \ddots & \\ u_{k-1} & & & u_k - u_1 \end{pmatrix}.$$

The marix $\mathcal{W}$ is symmetric[3], with entries that are bilinear in the entries of u and v. And it only works on rank 1 matrices, as follows. Let A be of rank 1, i.e. $A = uv^\top$, then

$$\mathcal{W}(A) = W_n[u] \otimes W_m[v].$$

But this can be extended by linearity. Ending up with a linear mapping $\mathcal{W}(A)$ that linearly depends on A, i.e. linear in the entries A_{ij}.

Next we need to build up a matrix-set comprised of the Kronecker product of skew-symmetric matrices, i.e. $A^\top = -A$.

Definition 3. Define $\mathcal{A}(k)$ as the space of skew-symmetric matrices of size k. Next denote with $\mathcal{A}(n) \otimes \mathcal{A}(m) \subset \mathbb{S}^N$ a linear subspace, of symmetric matrcies, defined as the linear span of the Kronecker product of skew-symmetric matrices of size n and m, respectively.

A detailed explanation of elements of this linear subspace is given in Appendix A.1. No we can state Hildebrand's theorem.

Theorem 1 (Hildebrand's Theorem)**.** *Let* $\min\{m, n\} \geq 3$. *Then an* $[n \times m]$*-matrix* A *maps* $\mathcal{L}^m$ *into* $\mathcal{L}^n$ *if and only if* A *can be extended to a feasible solution to the explicit system of linear matrix inequalities*

$$\mathcal{W}(A) + X \succeq 0, \quad X \in \mathcal{A}(n) \otimes \mathcal{A}(m),$$

in variables A, X.

The lesson we can draw from this theorem is that there is a linear matrix inequality which encapsulates the Lorentz-Positivity of a mapping, i.e. the restriction $A[\mathcal{L}^m] \subset \mathcal{L}^n$, which we will later see is of major importance to the main problem of this thesis.

[2]Notice the similarity with Lemma 1.

[3]Because the Kronecker-product of symmetric matrices, is also symmetric.

3 | Mathematical Optimization

In this chapter we will formally introduce Mathematical Optimization and the sub-field Robust Optimization. In general, optimization focusses on minimizing (or maximizing) a decision function over a set decision variables under a set of constraints. For both fields, we will mainly focus on convex and conic optimization problems.

3.1 Optimization Problems

A generic optimization problem (OP) has the following form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq b_i, \quad i = [m], \end{aligned} \tag{3.1}$$

where $x \in \mathbb{R}^n$ is called the decision variable, the function f_0 is the objective function, the functions f_i are the inequality-constraint functions, b_i's are called bounds for the constraints and $[m] = \{1, \dots, m\}$.
A decision variable, or solution, x is called feasible if every constraint is not violated. Otherwise it is called infeasible. A feasible solution is called optimal if it minimizes the objective function the most, notated with x^*. We call two OP's equivalent if they have the same optimal solution value. Specifically interesting are classes of optimization problems characterized by particular forms of objective and constraint functions.

3.1.1 Linear Optimization

The most well known class of OP is called linear programming (LP). An OP is called linear if both the objective function and the constraints functions are linear, i.e. it holds that

$$\forall x, y \in \mathbb{R}^n : f_i(ax + by) = af_i(x) + bf_i(y).$$

A generic LP has the following form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & G_i x \leq h_i, \; i = [l], \end{aligned} \tag{3.2}$$

where $G_i \in \mathbb{R}^{m \times n}$, $h_i \in \mathbb{R}^m$. OP's that are not linear are called non-linear.

3.1.2 Convex Optimization

Convex Programming is a generalization of LP. For an elaborate explanation of convex programming we refer to [2]. An OP is called convex if both the objective function and the constraints functions are convex, i.e. it holds that

$$\forall x, y \in \mathbb{R}^n : f_i(ax + by) \leq af_i(x) + bf_i(y), \quad \forall a, b \geq 0, a + b = 1.$$

For convex functions every local minimum is also a global minimum. Hence, minimizing a convex function is inherently easy.

3.1.3 Conic Optimization

The constraint functions of Problem (3.1) could be replaced by cone constraints, such that the decision variables "live" in a specific cone. This gives rise to Conic Optimization (CP). CP is a major sub-field of convex optimization, mainly because there are fast algorithms for solving special subclasses of these problems. In CP a linear objective function is minimized over the intersection of an affine subspace and a proper cone. A generic CP has the following form

$$(\mathrm{CP}_K)\ \underset{x}{\text{minimize}} \quad \{c^\top x : A_i x + b_i \in K_i,\ i = [l]\}, \tag{3.3}$$

where $x \in \mathbb{R}^n$ is the decision vector, $K_i \subset \mathbb{R}^m_i$ are proper cones and

$$x \mapsto A_i x + b_i,$$

are given affine mappings from $\mathbb{R}^n$ to $\mathbb{R}^{m_i}$. The three examples of proper cones, from Section 2.1.1, give rise to the three most prominent CP subclasses. First and foremost, LP,

$$\underset{x}{\text{minimize}} \quad \{c^\top x : A_i x + b_i \in \mathbb{R}^m_+,\ i = [l]\}, \tag{3.4}$$

which is equivalent to Problem (3.2).

SOCP

Secondly, Second Order Cone Optimization (SOCP),

$$\underset{x}{\text{minimize}} \quad \left\{c^\top x : \begin{pmatrix} A_i \\ c_i^\top \end{pmatrix} x + \begin{pmatrix} b_i \\ d_i \end{pmatrix} \in \mathcal{L}^m, i = [l]\right\} = \{c^\top x : ||A_i x + b_i||_2 \leq c_i^\top x + b_i \geq 0,\ i = [l]\},$$

SDP

And Lastly, Semi Definite Optimization (SDP),

$$\underset{x}{\text{minimize}} \quad \{c^\top x : Ax + B \in \mathbb{S}^k_+\} = \{c^\top x : Ax + B \succeq 0\},$$

where

$$x \mapsto \mathcal{A}x - B \equiv \sum_j^n (x_j A^j) - B$$

is an affine mapping from $\mathbb{R}^n$ to $\mathbb{S}^k$.[1] The $\succeq$-constraint in question is called a linear matrix inequality (LMI). Thus, it is the problem of minimizing a linear objective over a LMI.

A word of note

CP has a lot of structure, due the convexity and conic nature. Furthermore, within the subclasses of CP there is an additional structure, namely

$$LP \subset SOCP \subset SDP \subset CP, \tag{3.5}$$

[1]For completeness, in this sense, the space $\mathbb{S}^n$ is treated as an Euclidean Space equipped with the Forbenius inner-product, i.e. $\langle A, B\rangle = \mathrm{Tr}(AB) = \sum_{i,j} A_{ij}B_{ij}$.

indicating that every LP can be written as a SOCP and every SOCP can be written as a SDP. It turns out that most convex optimization problems can be described by just three families of proper cones: the non-negative orthants, lorentz cones and semidefinite cones [3]. Making it an incredibly rich field of optimization.

3.2 Uncertainty in Optimization

OP foccusses mostly on real world problems. Data for real-world problems is more often than not uncertain. Mathematicians came up with all sorts of way to deal with this uncertainty. Usually, uncertainty is expressed in the form of an uncertainty set $\mathcal{Z}$. A generic OP affected by an uncertainty set $\mathcal{Z}$ has the following form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & f_i(x,z) \leq b_i(z),\ i = [m],\ z \in \mathcal{Z}. \end{aligned}$$

The crux of these problems is that choosing an x could lead to an infeasible solution, due to inability to control z. Hence, we would like to restrict our choice of x to be feasible for all realizations of z, in order to safely optimize over x. The way this is done is heavily dependent on the structure of both the optimization problem and the uncertainty set.

3.2.1 Robust Optimization

A methodology for handling such problems is offered by Robust Optimization, focussing on worst-case analysis. For an extensive overview of the theory we refer to [4]. Since we will only be discussing SOCP's affected by ellipsoidal uncertainty, we will restrict ourselves to the convex and conic optimization domain. Without first going into all the technical details surrounding our exact uncertain SOCP, we will discuss two prominent methods, the adversarial approach and the Robust Counterpart method in a more general setting. Both methods try to robustify the problem by making sure that the solution of the robustified problem is optimal and feasible for all the realizations of z. In a way, the dependence on z is factored out.

The Adversarial Approach

The adversarial approach is a generic procedure for robustifying convex optimization problems. It boils down to finding a finite list for the worst-case realizations of z. It is based on two steps. First, the *optimization step*, in which the problem is solved for a finite set of uncertainties, called scenarios. This will produce an optimal solution x^*. Then the *pessimization step* will find a realization z^*, which violates the current optimal solution the most. This z^* will be added to the other scenarios. This process is repeated until no more realizations of z, which violate the original constraint, can be found. If the optimization an pessimization step are both exact, then the resulting solution is known to be robustly optimal. If one of them is approximate, then you cannot guarantee optimality.

Notice, in the optimal step every scenario produces an additional constraint, of the same structure as the original uncertain constraint but with fixed uncertainty. Since for CP's the constraints are convex in x we can solve the optimization step in an efficient way. The pessimization step is governed by the structure of the individual constraints and the structure of the uncertainty set, which could potentially make it a difficult problem. Luckily in our case of ellipsoidal uncertainty it is relatively simple. We will fully elaborate this in the next chapter.
Usually, every constraint produces a pessimization. But this z does not necessarily have to violate the original constraint. Only if it does it will become a new scenario for that specific constraint. Thus, every constraint has its own set of scenarios, denoted as $\mathcal{Z}_i$ for constraint i. Clearly, there is room for variation. For instance, one could also put all the scenarios in one set, but this would greatly increase the number of constraints (added in the next uncertainty step). We will stick to the specified version of one pessimization per constraint, possibly leading to a scenario.

The Robust Counterpart Method

Instead of iteratively building to a robustified solution, the robust counterpart method derives it in one go, by finding a deterministically equivalent constraint for each uncertain constraint. This deterministic equivalent is called the robust counterpart (RC). Let $z \in \mathcal{Z}$ be the uncertainty set in question. In the case of CP perturbed by uncertainty we can write it in the following form

$$\underset{x}{\text{minimize}} \quad \{c^\top x : A_i(z)x + b_i(z) \in K_i,\ i = [m],\ z \in \mathcal{Z}\}. \tag{3.6}$$

A conic constraint i is violated for some realization $(A_i(z), b_i(z))$ if the affine mapping is not part of the cone any-more, i.e. $A_i(z)x + b_i(z) \notin K_i$. We want to safeguard for this by rewriting the original constraint into its RC

$$\underset{x}{\text{minimize}} \quad \{c^\top x : A_i x + b_i \in K_i,\ i = [m],\ \forall z \in \mathcal{Z}\}. \tag{3.7}$$

By restricting the feasible set in this way we make sure that all possible decision variables x are feasible for the constraints of (3.6) for every realization of z. We know (3.6) and (3.7) are not equivalent. But they are related, the optimal solution of the RC will always be higher than the nominal solution of the original problem, i.e. $z = 0$.[2]
Unfortunately, in general, finding an RC is computationally intractable, since it has to be feasible for all realizations of z. This uncertainty set can have infinitely many members. For instance, in [5] they showed even for simple LPs finding an RC becomes computationally intractable. However, as we will see ellipsoidal uncertainty is a special case, where things become computationally tractable.

Unsubstantiated Claim

If we consider the case that the uncertainty set $\mathcal{Z}$ is a proper cone, which technically we are considering[3], then using the conclusions from Section 2.1.2,

[2]Technically, both optimal solutions could be equivalent, but this would in reality, where things are more often than not subject to randomness, never happen.

[3]The norm-cone $\{(z, r) : ||z|| \geq r\}$ is a closed,c convex, solid and pointed. Hence, it is a proper cone.

additional linear algebra, combined with the fact that the CP mapping is affine, one can deduce that Problem (3.7) is equivalent to

$$\underset{x}{\text{minimize}} \quad \{c^\mathsf{T} x : \mathcal{B}_i[\mathcal{Z}] \subset K_i\},$$

where the $\mathcal{B}_i$ is a linear map. Hence, the inclusion $\mathcal{B}_i[\mathcal{Z}] \subset K_i$ is equivalent to $\mathcal{B}_i$ being a $\mathcal{Z}$-to-K positive mapping. Concluding, the tractability of the robust counterpart depends on the availability of a description for the $\mathcal{Z}$-to-K positive cone.

But not to get ahead of ourselves, there are three main reason for proposing this unsubstantiated claim. Firstly, to hint on how in general this is done. Secondly, to point out that a tractable RC revolves around finding out if the constraint maps one cone into another cone. And finally, to shed light on the extensiveness of possibilities coming forth from the conic structure of CP. In the next chapter we will substantiate this claim for the constraint in question specifically.

4 | Applying Robust Optimization

In the previous chapters the ground work for applying the RC method and the adversarial approach has been sketched. In this chapter we will discuss the application of both methods in detail. Consider the following SOCP affected by ellipsoidal uncertainty

$$\begin{aligned} &\underset{x}{\text{minimize}} \quad c^\top x \\ &\text{subject to} \quad ||A_i(z)x + b_i(z)||_2 \leq c_i^\top(z)x + d_i(z),\ z \in \mathcal{Z},\ i = [l], \end{aligned} \tag{4.1}$$

where $x \in \mathbb{R}^k$ are the decision variables, the uncertainty z 'lives' in

$$\mathcal{Z} = \{z \in \mathbb{R}^{m-1} : ||z||_2 \leq r\},$$

which is an ellipsoidal uncertainty set, and $A_i(z) \in \mathbb{R}^{(n-1)\times k}$, $b_i(z) \in \mathbb{R}^{n-1}$, $c_i(z) \in \mathbb{R}^k$, $d_i(z) \in \mathbb{R}$ are affine in z, l is the number of constraints involved and $c \in \mathbb{R}^k$. It is worth nothing that $(n-1)$ is the size of the vector in the norm. Without loss of generality we can assume that the uncertainty set $\mathcal{Z}$ is an euclidean ball in $\mathbb{R}^{m-1}$, i.e $\mathcal{Z} = \{z \in \mathbb{R}^{m-1} : ||z||_2 \leq 1\}$. Hence, Problem (4.1) is equivalent to the following CP

$$\underset{x}{\text{minimize}} \quad \left\{ c^\top x : \begin{pmatrix} A_i(z)x + b_i(z) \\ c_i^\top(z)x + d_i(z) \end{pmatrix} \in \mathcal{L}^n,\ ||z||_2 \leq 1,\ i = [l] \right\} \tag{4.2}$$

which clearly is an uncertain SOCP.

4.1 The Adversarial Approach

First we will apply the adversarial approach. Notice that (4.1) can be rewritten into the form

$$\begin{aligned} &\underset{x}{\text{minimize}} \quad c^\top x \\ &\text{subject to} \quad f_i(x,z) \leq 0, \quad z \in \mathcal{Z}, \quad i = [l], \end{aligned} \tag{4.3}$$

where l is the number of constraints in question and

$$f_i(x,z) = ||A_i(z)x + b_i(z)||_2 - c_i^\top(z)x - d_i(z).$$

Hence, we conclude f is convex in both x and z. As explained the optimization step will be simple. But in the pessimization step we want to maximize $f_i(x,z)$ for a fix x. This will pose problems because of the convexity in z.[1] However, in [6] it is shown that the pessimization problem has some kind of hidden convexity structure, which can be exploited. More explicitly, it is equivalent to a convex optimization problem.[2] And for convex problems every local minimum is also

[1]Maximizing a concave function is easy, it is equivalent to minimizing a convex function. A function f is concave if $-f$ is convex.

[2]One can show our problem is actually a quadratically constrainted quadractic problem. And using some basic assumptions we can establish an equivalence to a min-max-convex problem. This allows for recovering of the optimal solution of the non-convex problem via their equivalent convex counterparts. Hence, the name hidden convexity.

a global minimum. Therefore, the pessimization step, finding the worst-case z for a fixed x, is actually a simple problem.

4.1.1 Optimization-step

Let a finite collection of scenarios be given by

$$\mathcal{Z}_i = \{z_{i,1}, \dots, z_{i,k_i}\} \subset \mathcal{Z},\ i = [l],$$

where k_i indicates the number of scenarios for constraint i. Then we can solve the sampled robust problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & f_i(x, z_{i,j}) \leq 0,\ i = [1],\ j = [k_i], \end{aligned} \tag{4.4}$$

which is a basic convex optimization problem. Hence, it can be solved efficiently resulting in an optimal solution x^*.

4.1.2 Pessimization-step

Given such a x^* and using some extensive bookkeeping one can rewrite[3]

$$||A_i(z)x^* + b_i(z)||_2 \leq c_i^\top(z)x^* + d_i(z) \Rightarrow z^\top Q_i(x^*)z + p_i^\top(x^*)z + r_i(x^*) \leq 0,\ i = [l].$$

Thus, for the pessimization step of constraint i, we get an optimization problem of the form

$$\underset{z}{\text{maximize}} \quad \{z^\top Q_i z + p_i^\top z + r_i : |z||_2 \leq 1\}. \tag{4.5}$$

This can be solved efficiently using the hidden convexity structure. Finding such a solution z in the literature is called the Trust Region Problem, which goes beyond the scope of this thesis. But for completeness we refer to [7] in which the method is described in detail. From now on we will refer to this as the oracle i for finding z for constraint i.

4.1.3 The Cutting Set Method

Now it is time to introduce the adversarial approach as described in [8]. To this extent define

$$F_i(x) = \sup_{z \in \mathcal{Z}} f_i(x, z),$$

the worst-case constraint function, and

$$V(x) = \max_{i=1,\dots,l} F_i(x),$$

the maximum constraint violation. The whole procedure can best be summarized into an algorithm, as is done in Algorithm 1. To initialize the algorithm we start with finding the nominal solution by solving Problem (4.3) and setting $z = 0$ for all constraints.
The method is called the Cutting-Set method, because it cuts away subsets of the feasible region. Similar to the way the cutting-planes method cuts away the feasible regions of a LP with (hyper)planes. It can be seen as a non-linear extension and its convergence is proved in the paper.

[3]This is possible because the constraint is bi-affine in x and z.

Algorithm 1 The Cutting-Set Method

```
Set, the stopping criterion $V^{tol} > 0$ and $\mathcal{Z}_i := \{0\}, \quad i = [l]$
repeat
    (1) Optimization
    Solve sampled problem (4.4) with $\mathcal{Z}_i,\ i = [l]$ and return a solution $x^*$
    (2) Pessimization
    for $i = 1, \dots, l$ do
        (a) Call oracle $i$ to find $z(x^*)$
        (b) Calculate $F_i(x^*)$
        if $F_i(x^*) > 0$ then
            add $z(x^*)$ to $\mathcal{Z}_i$
until $V(x^*) \le V^{tol}$
```

Conclusion

Based on the algorithm we will draw some preliminary conclusions. If we apply the adversarial approach, then at each step a deterministic SOCP has to be solved, which will grow in size as the algorithm progresses. And after that a constant number of l problems of the form (4.5) is solved, resulting in at most l new scenarios, but probably less. Solving the pessimization problem is simple, as is solving SOCP's. But if the original problem contains many constraints, then this might result in an enormous amount of SOCP-constraints in the final steps.

4.2 The Robust Counterpart Method

Instead of iteratively building to a robustified solution, we can derive the RC as an explicit SDP reformulation. Existence of such an explicit representation was a long-standing open question [4]. For reference, a concrete example for deriving the explicit SDP-reformulation is added in the end of this chapter.

4.2.1 Tractability Condition

For convenience, we define $\zeta = \begin{pmatrix} z \\ 1 \end{pmatrix} \in \mathbb{R}^m$, and then we reformulate (4.2) into

$$\underset{x}{\text{minimize}} \quad \left\{ c^\top x : B_i(z)x + \beta_i(z) \in \mathcal{L}^n,\ \forall \zeta \in \mathcal{L}^m,\ i = [l] \right\}, \tag{4.6}$$

where, $B_i(z) = [A_i(z), c_i^\top(z)]^\top$ and $\beta_i(z) = [b_i(z), d_i(z)]^\top$.

Looking at a specific constraint i, this leads to

$$B(z)x+\beta(z) = \begin{pmatrix} A(z)x + b(z) \\ c^\top(z)x + d(z) \end{pmatrix} = \begin{pmatrix} A_1x + b_1 & A_2x + b_2 & \cdots & A_mx + b_m & A_0x + b_0 \\ c_1^\top x + d_1 & c_2^\top x + d_2 & \cdots & c_m^\top x + d_m & c_0^\top x + d_0 \end{pmatrix} \begin{pmatrix} z_1 \\ \vdots \\ z_m \\ 1 \end{pmatrix} = \mathcal{B}_x\zeta,$$

where $\mathcal{B}_x : \mathbb{R}^m \to \mathbb{R}^n$ describes an affine mapping of x. Following Section 2.1.2 motivates rewriting Problem (4.6) into

$$\underset{x}{\text{minimize}} \quad \{c^\top x : \mathcal{B}_x[\mathcal{L}^m] \subset \mathcal{L}^n\}. \tag{4.7}$$

Intuitive Reasoning

In order to make a CP robust we need to make sure that every possibility of x is feasible for every instance of z. This not the case if $\mathcal{B}_x\zeta \notin \mathcal{L}^n$,. The matrix $\mathcal{B}_x$ acts on vector ζ 'living' in the Lorentz cone $\mathcal{L}^m$ and projects it onto $\mathbb{R}^n$. But this can be anywhere and being the 'adversary' it will pick the worst possible outcome, which is not inside of $\mathcal{L}^n$. Requiring $\mathcal{B}_x$ to project all ζ into $\mathcal{L}^n$ is equivalent to $\mathcal{B}_x[\mathcal{L}^m] \subset \mathcal{L}^n$, which makes the affine mapping $\mathcal{B}_x$ robust, i.e. feasible $\forall\zeta$.

Hence, we can conclude that the RC is computationally tractable if and only if $\mathcal{B}_x$ is a Lorentz-positive mapping.

4.2.2 The SDP reformulation

Using Theorem 2.1.2 we can derive the RC. We state the following corollary from [4]

Corollary 1. *when $m-1 = \dim \mathcal{Z} \geq 2$ and $n = \dim \beta(x) \geq 3$, then the explicit $(n-1)(m-1) \times (n-1)(m-1)$-LMI*

$$\mathcal{W}(\mathcal{B}_x) + X \succeq 0, X \in \mathcal{A}(n-1) \otimes \mathcal{A}(m-1),$$

where $\mathcal{B}_x = [B(x), \beta(x)]$ in variables x and X is an equivalent SDP representation of the constraints of Problem (4.1).

Remember,

- $B \mapsto \mathcal{W}(B)$ is a linear mapping from the space of $[n \times m]$-matrices ($\mathbb{R}^{n\times m}$) to space of symmetric matrices $\mathbb{S}^N$, where $N = (m-1)(n-1)$.[4]
- and $\mathcal{A}(n-1) \otimes \mathcal{A}(m-1)$ is a linear subspace $\mathbb{S}^N$ spanned by the pair-wise kronecker product of skew-symmetric matrices of size $(n-1)$ and $(m-1)$. For details, see Appendix A.1.

It is worth noting that the bounds on the dimensions of both x and z can always be maintained by adding zero-columns or zero-rows to $\mathcal{B}_x$, respectively. The $\mathcal{W}(\mathcal{B}_x)$ can be seen as the SOCP-constraint elevated or reorganized into a specific matrix structure, which is the SDP reformulation of the original cone structure. The special structure of matrix X acts as a restriction on the reformulation, in a sense forcing the Lorentz-positivity. If such an X cannot be found then there exists no explicit RC. But if it exists than the RC is conveniently given by $\mathcal{W}(\mathcal{B}_x) + X \succeq 0$.

[4]It is worth nothing that technically it only maps rank 1 matrices, but this can be extended by linearity.

Going back to our original Problem (4.6) we can write the RC as

$$\underset{x, X_i}{\text{minimize}} \quad \left\{ c^\top x : \mathcal{W}(\mathcal{B}_x^i) + X_i \succeq 0 \quad i = [l] \right\}, \tag{4.8}$$

which is a basic SDP. Hence it can be solved efficiently resulting in a robustly optimal solution x^*.

Conclusion

Based on the reformulation we can draw some preliminary conclusions. Firstly, it is interesting to note that the RC of a SOCP can be formulated as a SDP, which shows an interesting additional connection in structure of CP as a whole. Especially, if you consider this was the by-product answering a different question, namely if there exists an SDP reformulation of the cone of Lorentz-positive mappings. In hindsight it seems completely logical, but this can definitely be said about more facts in mathematics. Anyhow, even though the results are profound they probably are not practically relevant. Considering Problem (4.1) with relevant dimensions (n, m) the RC is an SDP of dimension $N = (n - 1)(m - 1)$ the size of the SDP grows enormously once the original problem becomes bigger. Large scale SDP turned out to be practically unsolvable with current day computers. However, it is interesting to see what happens if n and m are still small. Before going to the numerical experiment we provide an elaborate example.

Example 4.2.1:
Let $\mathcal{Z} = \{z \in \mathbb{R}^2 : ||z||_2 \leq 1\}$, thus $\zeta \in \mathcal{L}^3$. We consider the following uncertain SOCP

$$\underset{x \geq 0}{\text{minimize}} \quad x_3 : \left\| \begin{pmatrix} z_1 & 1 & 1 \\ 1 & z_2 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix} \right\|_2 \leq x_3 \tag{4.9}$$

Rewriting this CP into the form of (4.6).

$$\mathcal{B}_x = \begin{pmatrix} A_1 x + b_1 & A_2 x + b_2 & A_0 x + b_0 \\ c_1^\top x + d_1 & c_2^\top x + d_2 & c^\top x + d_0 \end{pmatrix} = \begin{pmatrix} x_1 & 0 & x_2 + x_3 + 1 \\ 0 & x_2 & x_1 + x_3 + 1 \\ 0 & 0 & x_1 + x_2 \\ 0 & 0 & x_3 \end{pmatrix} \tag{4.10}$$

Thus, in the form of CP (4.2), is given by

$$\underset{x}{\text{minimize}} \quad \{c^\top x : \mathcal{B}_x[\mathcal{L}^3] \subset \mathcal{L}^4\}. \tag{4.11}$$

Checking the Lorentz-positivity of $\mathcal{B}_x$ leads to the question if there exists a $X \in \mathbb{A}(3) \otimes \mathbb{A}(2)$ such that $\mathcal{W}(\mathcal{B}_x) + X \succeq 0$. Firstly, we know X is of the form

$$X = \begin{pmatrix} 0 & 0 & 0 & X_{14} & 0 & X_{16} \\ 0 & 0 & -X_{14} & 0 & -X_{16} & 0 \\ 0 & -X_{14} & 0 & 0 & 0 & X_{36} \\ X_{14} & 0 & 0 & 0 & -X_{36} & 0 \\ 0 & X_{16} & 0 & -X_{36} & 0 & 0 \\ X_{16} & 0 & X_{36} & 0 & 0 & 0 \end{pmatrix} \in \mathbb{S}^6$$

Secondly, remember $\mathcal{W}(\mathcal{B}_x) = (W_4 \otimes W_3)(\mathcal{B}_x)$ is only defined on rank 1 matrices ($A = aa^\top$). So we need to decompose $\mathcal{B}_x$ using a rank 1 decomposition as follows

$$\mathcal{B}_x = \begin{pmatrix} x_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} e_1^\top + \begin{pmatrix} 0 \\ x_2 \\ 0 \\ 0 \end{pmatrix} e_2^\top + \begin{pmatrix} x_2 + x_3 + 1 \\ x_1 + x_3 + 1 \\ x_1 + x_2 \\ x_3 \end{pmatrix} e_3^\top,$$

where e_i are the standard basis vectors which span $\mathbb{R}^3$. Leading to

$$\begin{aligned} \mathcal{W}(\mathcal{B}_x) &= W_4\left(\begin{pmatrix} x_1 \\ 0 \\ 0 \\ 0 \end{pmatrix}\right) \otimes W_3(e_1) + W_4\left(\begin{pmatrix} 0 \\ x_2 \\ 0 \\ 0 \end{pmatrix}\right) \otimes W_3(e_1) + W_4\left(\begin{pmatrix} x_2+x_3+1 \\ x_1+x_3+1 \\ x_1+x_2 \\ x_3 \end{pmatrix}\right) \otimes W_3(e_1) \\ &= \begin{pmatrix} x_1 & 0 & 0 \\ 0 & -x_1 & 0 \\ 0 & 0 & -x_1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + \begin{pmatrix} 0 & x_2 & 0 \\ x_2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \begin{pmatrix} 2x_3+x_2+1 & x_1+x_3+1 & x_1+x_2 \\ x_1+x_3+1 & -x_2-1 & 0 \\ x_1+x_2 & 0 & -x_2-1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} x1+x_2+2x_3+1 & 0 & x1+x_3+1 & x_2 & x_1+x_2 & 0 \\ 0 & -x1+x_2+2x_3+1 & x_2 & x1+x_3+1 & 0 & x_1+x_2 \\ x1+x_3+1 & x_2 & -x_1-x_2-1 & 0 & 0 & 0 \\ x_2 & x1+x_3+1 & 0 & x_1-x_2-1 & 0 & 0 \\ x1+x_2 & 0 & 0 & 0 & -x_1-x_2-1 & 0 \\ 0 & x1+x_2 & 0 & 0 & 0 & x_1-x_2-1 \end{pmatrix}. \end{aligned}$$

Therefore Problem (4.9) is equivalent to

$$\underset{x \geq 0, X \in \mathbb{S}^6}{\text{minimize}} \quad \{c^\top x : \mathcal{W}(\mathcal{B}_x) + X \succeq 0\}, \tag{4.12}$$

which is an SDP in the variables x, X, where X is a matrix containing three decision variables X_{14}, X_{16}, X_{36}.

5 | Numerical Experiment

In the previous chapter we set out two methods for solving Problem (4.1) in a robustly optimal fashion. From a theoretical perspective both methods are able to solve this problem efficiently.[1] However, in general, SDP suffers from an extremely high time complexity. Current SDP solvers scale as $O(N^{4.5})$ or worse [9], where N is the dimension of the the relevant matrices. This makes solving large scale SDP's practically impossible. In our case, for a relatively small problem of say $x \in \mathbb{R}^{10}$ and $z \in \mathbb{R}^5$, the SDP consists of matrices of $N = 45$. So even though the RC might be computationally tractable, the resulting SDP would probably still be too computationally intensive due to its intense size. By comparing the numerical performance of both the adversarial Approach and the Robust Counterpart method we are able to draw more practical conclusions.

5.1 Experimental Setup

For the numerical experiment we used Matlab (version 9.8 R2020a) to write the code itself and for solving the optimization problems we made use of CVX (version 2.2) which utilized SDPT3 (version 4.0), a non-comercial all purpose SDP-solver. The code was run on Mac OS X (x86_64 version 10.15.7) with a 2,5 GHz Dual-Core Intel Core i5 processor and a total RAM of 8 GB (1600 MHz DDR3).

5.1.1 Sampling

Different problem sizes of Problem (4.1), which will be named experiments, are explored, with respect to n the dimension of x, m the dimension of the uncertainty z, l the number of constraints. Remember, constraint i looks like

$$\left\| \left(A_{i0} + \sum_{j=1}^{m} z_j A_{ij} \right) x + \left(b_{i0} + \sum_{j=1}^{m} z_j b_{ij} \right) \right\|_2 \leq \left(c_{i0} + \sum_{j=1}^{m} z_j c_{ij} \right)^\top x + \left(d_{i0} + \sum_{j=1}^{m} z_j d_{ij} \right).$$

Without loss of generality, we restrict the vector in the norm to be of size n. For each experiment 50 instances are sampled as follows. Each entry of A_{ij},b_{ij} and c_{ij} is sampled uniformly from $[-1, 1]$ and then normalized in the following way

$$\begin{aligned} A_{ij} &= \frac{A_{ij}}{S_{i1}} && \text{where } \left\| [A_{i0}, \cdots, A_{im}]^\top \right\|_{2,2}, \\ b_{ij} &= \frac{b_{ij}}{S_{i1}} && \text{where } \left\| [b_{i0}, \cdots, b_{im}]^\top \right\|_{2,2}, \\ c_{ij} &= \frac{c_{ij}}{S_{i1}} && \text{where } \left\| [c_{i0}, \cdots, c_{im}]^\top \right\|_{2,2}, \end{aligned}$$

where $||A||_{2,2} = \max_{||x||_2 \leq 1} ||Ax||_2$. To ensure Slater feasibility of the sampled problem we set $d(z) = -1$ to be fixed and unperturbed by uncertainty, facilitating that at least one solution is strictly feasible.

[1]In the case of the Robust Counterpart Method, efficiency is meant in the sense of the RC being computationally tractable.

Table 5.1 describes the different experiments that were run. In the first set we are interested in the change of the problem size, i.e. the n. In the second set we focus on the dimensionality of the uncertainty, i.e the m. And in the last set only the number of constraints was changed.

Table 5.1: Three sets of Experiments for comparing the numerical performance of the adversarial approach and the RC method applied to Problem 4.1. n **is the dimension of the vector in norm,** m **the dimension of the uncertainty and** l **the number of constraints.**

Set 1	n	m	l
experiment 1	10	2	1
experiment 2	20	2	1
experiment 3	50	2	1
Set 2	n	m	l
experiment 4	20	5	1
experiment 5	20	10	1
Set 3	n	m	l
experiment 6	10	2	2
experiment 7	10	2	5

5.1.2 Averaging of the Experiments

For an experiment we can easily average the running time of the Robust Counterpart Method. Conversely, averaging the adversarial approach requires some considerations. Every instance has a different number of cycles after which the algorithm terminates. Every cycle k of instance i consists of a pair $(t_{ik}, \epsilon_{ik}(t_{ik}))$, where $\epsilon_{ik}(t_{ik})$ indicates the optimality gap at moment t_{ik}. Let $\hat{t}_{ik}$ be the round-off of t_{ik} upto 2 decimal points. Then

$$\hat{\epsilon}(t) = \frac{1}{n} \sum_{ik:t=\hat{t}_{ik}} \epsilon_{ik},$$

is the arithmetic mean of the optimality gap for a discretized time-point t. Using the points $(t, \hat{\epsilon}_k)$ we can construct the moving average of the scenario. The averaging of the maximum constraint violation (V) is done in an analogous manner.[2]

5.1.3 Implementation Problems

During the implementation of both methods we ran into the following problem. The robustly optimal solution of the RC method was always more pessimistic than the optimal solution computed by the adversarial approach. Most of the time the difference was minimal (around 5%) but there were significant outliers. This is, possibly, due to a minor bug in the adversarial approach. But as of yet it has not been discovered. This bug will be visible in the plots as the adversarial approach not reaching 100% of optimality. However, the methods still 'converges', in the sense of having found all pessimization which violate the original constraint. Thus, we are still able to compare both methods.

[2]Only the round off point of t_{ik} was set to 1 decimal.

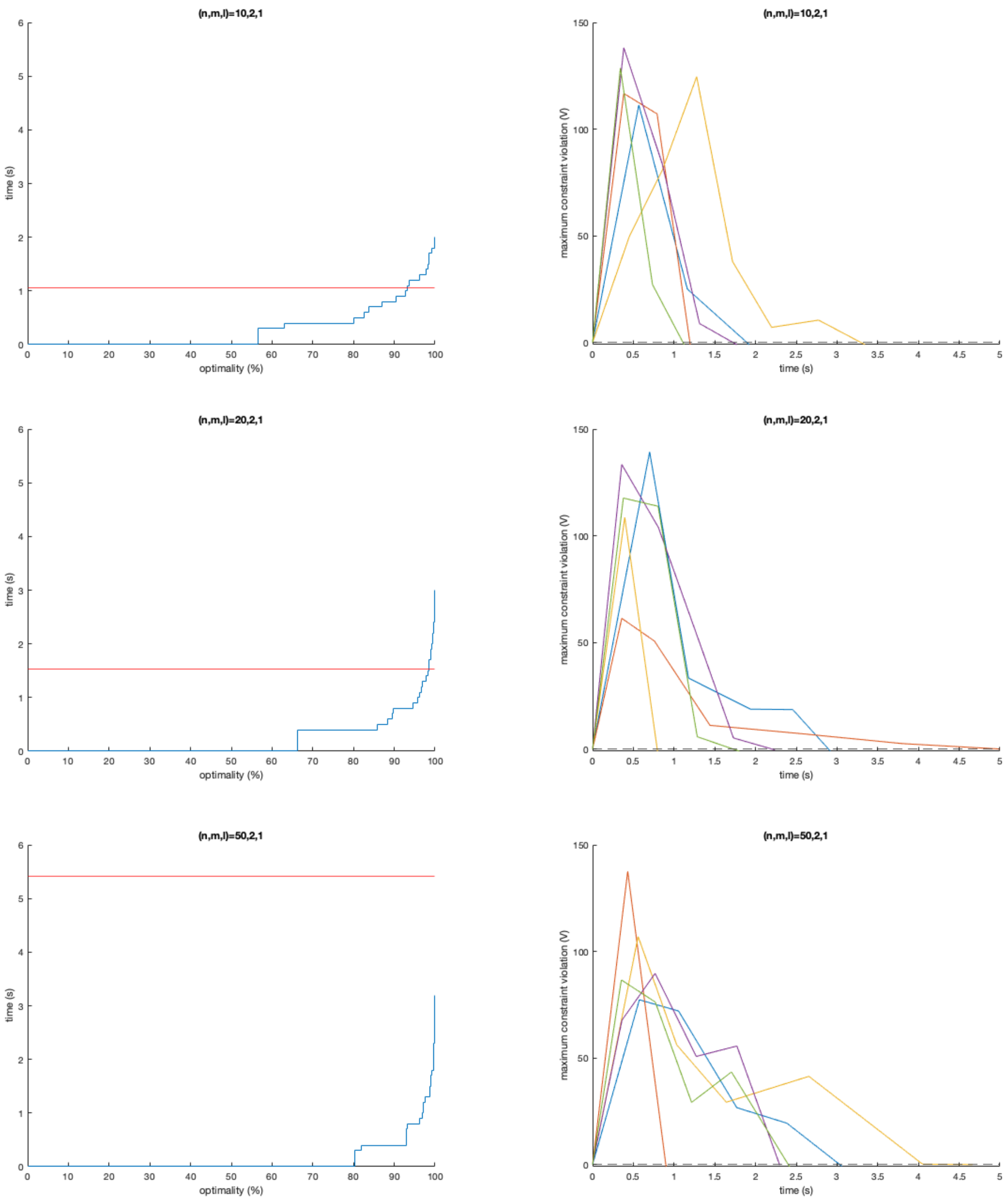

Figure 5.1: Optimality-time plots and max violation constraint plots for the first set of experiments. Red indicates the average running time of the RC method, blue the moving average running time of the adversarial approach. The right plots indicate trajectories of the maximum constraint violations for 5 instances.

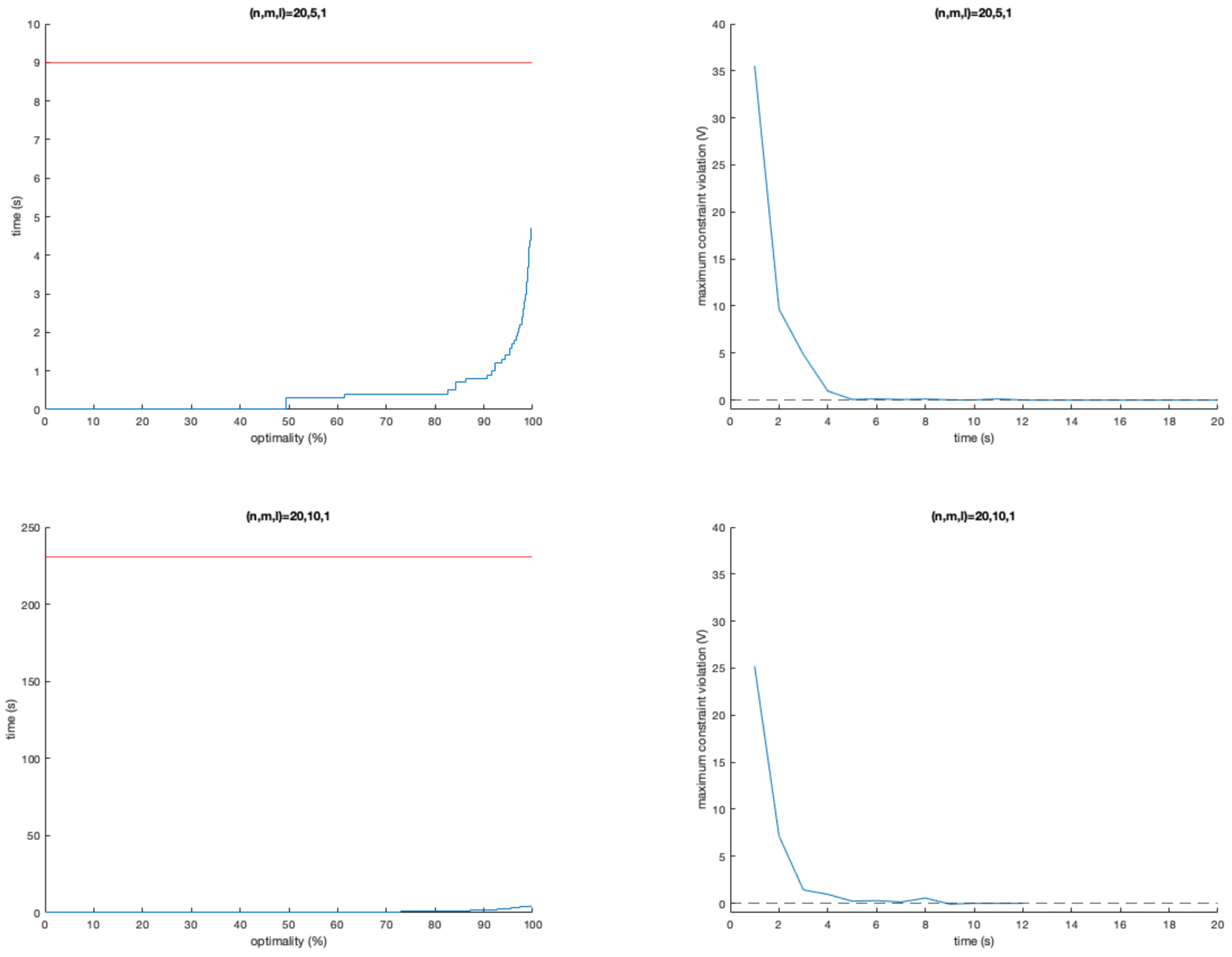

Figure 5.2: Optimality-time plots and max violation constraint plots for the second set of experiments. Red indicates the average running time of the RC method. The blue indicates the moving average running time of the adversarial approach in the left plot and the moving average of the max constraint violation in the right.

5.2 Results

As explained three difference sets of experiments are set out. For every optimality-time plot the red line indicates the average running time of the RC method and the blue line is the moving average of the adversarial approach. In Figure 5.1 the first set of experiments is visible. Because the averaging of the maximum violation of one constraint is not interesting, a sample of 5 instances is given. This gives an indication of the performance and to verifying the termination of the adversarial approach.
Interestingly, for small problem sizes the RC method outpreforms the adversarial approach. But as the size of the problem increases the adversarial approach is preforming much better. This is due to the fact that solving an SOCP, with a lot of (similar) constraints, is always faster than solving a large scale SDP.
In Figure 5.2 the result of the second set of experiments is given. Combined with Experiment 2, the second experiment of Figure 5.1, we conclude that the RC method starts to break down as the dimension of uncertainty increases.
The results of the last set of experiments are visible in Figure 5.3. Combined with the Experiment 1, the first experiment of Figure 5.1, we can conclude that the average running time increases if the number of constraints increases for both methods. This is not unexpected, more constraints in the original problem resulting in more RC's and more optimization- and pessimization steps.

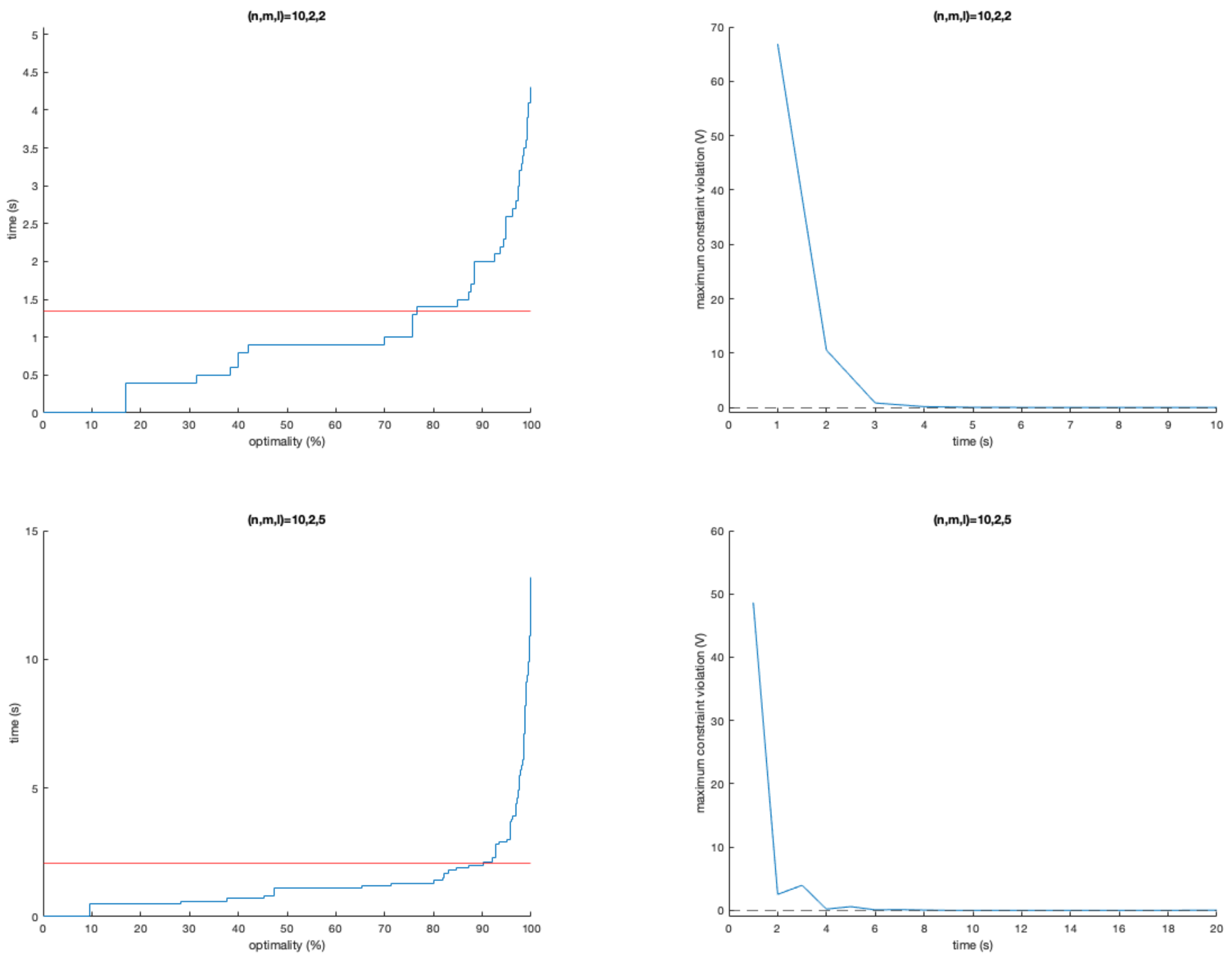

Figure 5.3: Optimality-time plots and max violation constraint plots for the second set of experiments. Red indicates the average running time of the RC method. The blue indicates the moving average running time of the adversarial approach in the left plot and the moving average of the max constraint violation in the right.

5.3 Summary

In Table 5.2 an overview of the average running times for the experiments is given. Additionally, the standard deviation has also been included in brackets. It is reasonable to assume the bug in the implementation is of minor importance, because both methods are terminating properly.
Hence, we can draw the conclusion that the adversarial approach is preferred for large scale problems.[3] Conversely, for small scale problems the RC method is much more efficient. This was not entirely unexpected, because the RC method takes care of all these pessimistic scenarios in one big transformation instead of adding more and more scenarios like the adversarial approach.
Additionally, it is interesting to notice that the standard deviation of the average running time is relatively small.
A lot of methods for large scale optimization problems are based on solving many small problems.[4] Indeed, solving a lot of small problems is much faster than solving one big problem. Hence, this could make the RC method the ideal candidate for solving sub-problems which are focused on finding a robustly optimal solution for SOCP-constriants perturbed by ellipsoidal uncertainty. However,

[3]There is an argument to be made considering quantum optimization algorithms which provide a considerable speed up for solving SDP's [?]. But that is for the future to decide.

[4]Exactly like the adversarial approach does.

to answer this question it is clear that more thorough research is necessary. Perhaps there are even pessimization problems, which require solving this problem, but this remains to be seen.

Table 5.2: Summarizing table for the different experiments (of Table 5.1). The Running times are averages over 50 instances. For the adversarial approach this indicates the running time of the entire algorithm up until termination. For completeness, the standard deviations are added in brackets.

Experiment	Parameters(n,m,l)	SDP Running Time (sec)	Aversarial Approach Running Time (sec)
1	(10,2,1)	1.0613 (0.3102)	2.5053 (0.9009)
2	(20,2,1)	1.5247 (0.4754)	3.1059 (2.2561)
3	(50,2,1)	5.4077 (0.8729)	3.2089 (1.2123)
4	(20,5,1)	8.9920 (0.9775)	4.9957 (6.1830)
5	(20,10,1)	230.6780 (32.7145)	5.0665 (2.7425)
6	(10,2,2)	1.3432 (0.4225)	4. 5.0080 (3.4422)
7	(10,2,5)	2.0620 (0.4661)	13.7591 (13.1923)

6 | Conclusion

In this thesis we have presented two methods for robustifying a Second Order Cone Programming perturbed by an ellipsoidal uncertainty set. Both methods work in completely different ways.

The adversarial approach is based on the cutting-set method, which iteratively optimizes the problem for a finite set of uncertainties and then finds a scenario, i.e. realization of uncertainty, for every constraint, which pessimizes this optimal solution the most. The algorithm terminates once no more pessimizations, which violate the original constraints, can be found. The final solution found in the optimization step is the robustly optimal solution for the original problem. The pessimization step turned out to have a hidden convexity structure, which makes it equivalent to minimizing a convex optimization problem, the so called Trust Region Problem. Instead of maximizing a convex function, which is inherently difficult, a pessimization is found using a 'simple' problem. In the optimization step a basic SOCP is solved. Thus, both steps are efficiently solvable, which makes the adversarial approach efficient. However, the major downside is the growing number of constraints per iteration.
The RC method derives a deterministically equivalent problem. In general, finding a RC is computationally intractable. But for a SOCP perturbed by an ellipsoidal uncertainty set, this RC can be reformulated exactly to an explicit SDP reformulation. Hence, it turned out to be computationally tractable, in a theoretical sense. However, the size of the resulting SDP is a lot larger than the size of the original problem. Combined with the fact that solving a SDP is far more time consuming than solving a SOCP this makes the RC method impractical for large scale problems.

Modest numerical experiments supported the fact that the SDP-reformulation breaks down for large scale problems. Interestingly, based on the experiments, the SDP-reformulation out-preforms the adversarial approach on small problems, which could potentially make it a prevalent method for solving small problems. However, this is a preliminary conclusion, which is largely unsubstantiated and needs to be researched more thoroughly.

A Relevant Mathematical Concepts

A.1 The Subspaces of Skew-Symmetric Matrices

In this section one of the crucial concepts for deriving the RC is set out. We will start with two definitions,

- A matrix is called **symmetric** if is square and it equals its transpose, i.e. $A = A^\top$.
- A matrix is called **skew-symmetric** if it is square whose transpose is its negative, i.e.e $A^\top = -A$,

and some generic examples,

Example A.1.1:

$$X_2 = \begin{pmatrix} 0 & X_{12} \\ -X_{12} & 0 \end{pmatrix}, \quad X_3 = \begin{pmatrix} 0 & X_{12} & X_{13} \\ -X_{12} & 0 & X_{23} \\ -X_{13} & -X_{23} & 0 \end{pmatrix}, \quad X_4 = \begin{pmatrix} 0 & X_{12} & X_{13} & X_{14} \\ -X_{12} & 0 & X_{23} & X_{24} \\ -X_{13} & -X_{23} & 0 & X_{34} \\ -X_{14} & -X_{24} & -X_{34} & 0 \end{pmatrix}.$$

Notice that every entry on the diagonal is zero, this is true for all skew-symmetric matrices. More interestingly, a skew-symmetric matrix only consists of $\frac{n^2-n}{2}$ distinct values. Before we can move on we need to define the kronecker-product of two matrices A and B, as follows

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}.$$

Note in general $A \otimes B$ and $B \otimes A$ are different matrices. We will show one example of how to compute a kronecker product.

Example A.1.2:

$$\begin{pmatrix} 1 & 2 & 0 \\ 4 & 3 & -1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} & 2\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} & 0\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \\ 4\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} & 3\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} & -\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 2 & 0 & 0 \\ 2 & 3 & 4 & 6 & 0 & 0 \\ 0 & 4 & 0 & 3 & 0 & -1 \\ 8 & 12 & 6 & 9 & -2 & -3 \end{pmatrix}$$

Linear Subspace

We need the kronecker product to construct the linear subspace of skew-symmetric matrices. This space is constructed by setting generic matrices, like in Example A.1, as basis matrices, which span the subspace. We will define this subspace as follows

Definition 4. Let $\mathcal{A}(n) \otimes \mathcal{A}(m) \subset \mathbb{S}^N$, where $N = nm$ be a linear subspace of symmetric matrices, defined by the linear span of kronecker products $S \otimes T$ of all skew symmetric $[n \times n]$-matrices S and $[m \times m]$-matrices T.

We will discuss some specific examples, before explaining some general properties.

Example A.1.3:

- Let $X \in \mathcal{A}(2) \otimes \mathcal{A}(2)$, then

$$X = \begin{pmatrix} 0 & 0 & 0 & X_{14} \\ 0 & 0 & X_{14} & 0 \\ 0 & -X_{14} & 0 & 0 \\ -X_{14} & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & A \\ A^\top & 0 \end{pmatrix}$$

  This matrix consists of two-by-two blocks of two-by-two. Notice, X has only 1 disctinct value.

- Let $X \in \mathcal{A}(2) \otimes \mathcal{A}(3)$, then

$$X = \begin{pmatrix} 0 & 0 & 0 & 0 & X_{15} & X_{16} \\ 0 & 0 & 0 & -X_{15} & 0 & X_{26} \\ 0 & 0 & 0 & -X_{16} & -X_{26} & 0 \\ 0 & -X_{15} & -X_{16} & 0 & 0 & 0 \\ X_{15} & 0 & -X_{26} & 0 & 0 & 0 \\ X_{16} & X_{26} & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & A \\ A^\top & 0 \end{pmatrix},$$

  consists of three-by-three matrices in blocks of two-by-two, with the upper-triangle block made out of a 3-by-3 skew symmetric matrix. Thus, only 3 distinct values.

- Let $X \in \mathcal{A}(3) \otimes \mathcal{A}(2)$, then

$$X = \begin{pmatrix} 0 & 0 & 0 & X_{14} & 0 & X_{16} \\ 0 & 0 & -X_{14} & 0 & X_{-16} & 0 \\ 0 & X_{14} & 0 & 0 & 0 & X_{36} \\ X_{14} & 0 & 0 & 0 & -X_{36} & 0 \\ 0 & -X_{16} & 0 & -X_{36} & 0 & 0 \\ X_{16} & 0 & X_{36} & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & A_1 & A_2 \\ A_1^\top & 0 & A_3 \\ A_2^\top & A_3^\top & 0 \end{pmatrix},$$

  consists of two-by-two matrices in blocks of three-by-three, with an upper-triangle block made out of three 3-by-3 skew symmetric matrices. Thus, only 3 distinct values.

In general a $[N \times N]$-matrix $X \in \mathcal{A}(n) \otimes \mathcal{A}(m)$, will consists of $M = \frac{m^2-m}{2}$ different $[n \times n]$-skew-symmetric matrices. And it will have the following form

$$X = \begin{pmatrix} 0 & A_1 & \cdots & A_{m-1} \\ A_1^\top & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & A_M \\ A_{m-1}^\top & \cdots & A_M^\top & 0 \end{pmatrix},$$

concluding the symmetric matrix X has

$$\left(\frac{n^2-n}{2}\right)\left(\frac{m^2-m}{2}\right)$$

distinct values.

Bibliography

[1] Roland Hildebrand. An lmi description for the cone of lorentz-positive maps ii. *Linear Multilinear Algebra - LINEAR MULTILINEAR ALGEBRA*, 55, 02 2006.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[3] Erling D. Andersen. Which cones are needed to represent almost all convex optimization problems?, Nov 2010.

[4] Aharon Ben-Tal, Laurent Ghaoui, and Arkadi Nemirovski. *Robust Optimization.* 08 2009.

[5] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of uncertain linear programs. *Operations research letters*, 25(1):1–13, 1999.

[6] Aharon Ben-Tal and Marc Teboulle. Hidden convexity in some nonconvex quadratically constrained quadratic programming. *Math. Program.*, 72:51–63, 01 1996.

[7] Jorge J Moré and Danny C Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

[8] Almir Mutapcic and Stephen Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods and Software*, 24(3):381–406, 2009.

[9] Yongbin Zheng, Yuzhuang Yan, Sheng Liu, Xinsheng Huang, and Wanying Xu. An efficient approach to solve the large-scale semidefinite programming problems. *Mathematical Problems in Engineering*, 2012, 2012.

[10] Almir Mutapcic and Stephen Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optimization Methods And Software*, 24:381–406, 07 2009.