Integer programming techniques for Polynomial Optimization

Gonzalo Muñoz

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

ABSTRACT

Integer programming techniques for Polynomial Optimization

Gonzalo Muñoz

Modern problems arising in many domains are driving a need for more capable, state-of-the-art optimization tools. A sharp focus on performance and accuracy has appeared, for example, in science and engineering applications. In particular, we have seen a growth in studies related to Polynomial Optimization: a field with beautiful and deep theory, offering flexibility for modeling and high impact in diverse areas.

The understanding of structural aspects of the feasible sets in Polynomial Optimization, mainly studied in Real Algebraic Geometry, has a long tradition in Mathematics and it has recently acquired increased computational maturity, opening the gate for new Optimization methodologies to be developed. The celebrated hierarchies due to Lasserre, for example, emerged as good algorithmic templates. They allow the representation of semi-algebraic sets, possibly non-convex, through convex sets in lifted spaces, thus enabling the use of long-studied Convex Optimization methods. Nonetheless, there are some computational drawbacks for these approaches: they often rely on possibly large semidefinite programs, and due to scalability and numerical issues associated with SDPs, alternatives and complements are arising.

In this dissertation, we will explore theoretical and practical Integer-Programming-based techniques for Polynomial Optimization problems. We first present a Linear Programming relaxation for the AC-OPF problem in Power Systems, a non-convex quadratic problem, and show how such relaxation can be used to develop a tractable MIP-based algorithm for the AC Transmission Switching problem. From a more theoretical perspective, and motivated by the AC-OPF problem, we study how sparsity can be exploited as a tool for analysis of the fundamental complexity of a Polynomial Optimization problem, by showing LP formulations that

can efficiently approximate sparse polynomial problems. Finally, we show a computationally practical approach for constructing strong LP approximations on-the-fly, using cutting plane approaches. We will show two different frameworks that can generate cutting planes, which are based on classical methods used in Mixed-Integer Programming.

Our methods mainly rely on the maturity of current MIP technology; we believe these contributions are important for the development of manageable approaches to general Polynomial Optimization problems.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First of all, I would like to thank my advisor, Professor Daniel Bienstock. I came to Columbia hoping to work with him, and everything turned out the best possible way. I encountered an extremely smart and dedicated person, an important figure in development of the Optimization field, and an encouraging and friendly advisor. I feel privileged to have been able to share so much with him. Working with Dan these 5 years has been a truly enjoyable experience; he always kept a wise balance between hard work and having fun while doing research.

I also want to thank the IEOR department for providing such a warm environment. The faculty members were always available to give us a hand and support our research. They constantly spread their enthusiasm and helped us mature as researchers. As for the IEOR staff (past and current), they continuously provided help with all our extra-academic issues, they pushed the social side of our PhD student life, and they did everything with a contagious smile.

To my fellow PhD students: I am extremely grateful for the good friends I made here at IEOR. To all of those who were interested in creating a strong bond, not constrained to the boundaries of the Mudd building, thank you very much. A special mention for my friend Antoine: I was very lucky that in such a small cohort I was able to find such a good friend. I was lucky to see him form a beautiful family, lucky to see him start a successful academic career and lucky to have been part of this process.

I also want to express my deepest gratitude to my family. They are my most faithful supporters, and they have always made me believe I could do whatever I set my mind to. I want to thank them for being next to me this whole time, even if we were 10.000 kms apart.

I would like to thank my old friends from Chile as well, for confirming that time and distance does not wear out good friendships. And to all the new friends I made in New York, I would like to thank them for sharing with me during this part of my life. I feel immense gratitude to everyone who crossed paths with me at some point in this incredible city.

Last, but most importantly, to Fernanda for being next to me the whole time. Having a partner capable of everything was the biggest support I had during these last years. She is one of the strongest and most resourceful woman I know, and being by her side filled this adventure with pure joy.

Chapter 1

Introduction

1.1 The main motivation: AC-OPF problem

The main motivation for this work, which involves the development of efficient and accurate Polynomial Optimization techniques, is drawn from a Power System problem: the so-called AC-OPF problem.

The AC-OPF problem was introduced in [33], and it is a fundamental software component in the operation of electrical power transmission systems. According to the Federal Energy Regulatory Commission[1]:

> *Finding a good solution technique for the AC-OPF could potentially save tens of billions of dollars annually.*

The goal of this problem is to determine where and how much power should be generated in order to satisfy given demands in the transmission grid, in a way that optimizes generation costs. The AC-OPF problem is solved every day, sometimes in time intervals of not more than 10 minutes, thus efficient solution techniques are needed. Moreover, given its important role, solutions should be as accurate as possible.

The problem can be formulated as a non-convex, continuous optimization problem. Non-convexities arise naturally from the physics laws governing the power flows in the transmission grid. Mainly Ohm's Law and Kirchhoff's current law. Any reliable model or solution method

[1] `https://www.ferc.gov/industries/electric/indus-act/market-planning/opf-papers.asp`

accounts for these somehow, as they are not a simple modeling choice, but a physical phenomenon.

Due to its intrinsic non-convex nature, the problem is NP-Hard (see [107, 74]). In routine problem instances, solutions of excellent quality can be quickly obtained using a variety of methodologies, including sequential linearization and interior point methods (e.g. MATPOWER [111]). Instances involving grids under stress or extreme conditions, however, can prove significantly more difficult, as the quality of the usual approximations and simplifications becomes modest. This, along with the recent advances in computation technology and optimization methodologies, has motivated considerable efforts in solving the problem more accurately and efficiently. For more background on this problem, see [16].

Recently, in [73], a semidefinite programming (SDP) approach is explored, where relaxations of this type are used to obtain valid bounds on the AC-OPF problem, and thus certify when a given feasible solution is close to optimal. These relaxations can provide good bounds, nevertheless, SDP solvers in general can be numerically unstable and slow. It has also been argued that, in general, the semidefinite approach can be inexact [62], even in small networks. Other approaches typically involve convex quadratic relaxations to provide good bounds, relying in more mature and stable optimization routines. See [73, 86, 59, 80, 87, 57, 28, 93, 61] and the citations therein for these and other cutting edge approaches.

In this dissertation we will present fast methods to obtain good bounds without having to rely on SDP tools. We will exploit linear programming tools as much as possible, providing elaborate linear inequalities that can yield valid and good quality bounds in short time. These can be useful on their own, or to work as a complement of other state-of-the-art relaxations. In addition, we will show these tools can be used to tackle the AC *transmission switching problem*, an extension of the AC-OPF problem where a planner can decide to turn off some lines of the grid.

1.2 Network Polynomial Optimization Problems

Building on the aforementioned motivation, the following question arises naturally: to what extent can one exploit linear programming (LP) techniques in non-convex problems?. Since, in general, the AC-OPF problem is *strongly* NP-Hard [107], for the question to have a non-trivial answer further characteristics need to be considered. In this case, we will rely on the fact that transmission grids are typically sparse networks in a very specific way; they present a *tree-like* structure (see [81]). More precisely, they are graphs of low *tree-width*, a graph-theoretical parameter that measures, roughly, how tree-like a given graph is. We will study what is the effect of this structure in the tractability of the problem.

In a more abstract level, we will introduce the concept of *Network Polynomial Optimization* (NPO) problems. These are optimization problems defined over networks where the decision variables are associated to the *nodes* of the network. Additionally, there are "flows" associated to each edge, which depend on the variables of the end-nodes. The expressions defining these flows will be allowed to be non-linear. In the AC-OPF problem, for example, the network is given by the transmission grid itself, the decision variables represent *voltages* in each node of the network, and the physics laws governing the power flows yield non-linear expressions for the power flowing through any line.

Mathematically, NPOs are defined as follows:

- There is a network $\mathcal{G}$, with vertices $V(\mathcal{G})$ and edges $E(\mathcal{G})$.
- For each $u \in V(\mathcal{G})$ there is a set of variables $\mathcal{X}_u$ and a set $\mathcal{K}_u$ of constraints associated with u.

And the optimization problem is of the following form:

$$\begin{aligned} & \min \ c^T x \\ \text{subject to:} \quad & \sum_{\{u,v\}\in\delta_{\mathcal{G}}(u)} p^{(k)}_{(u,v)}(\mathcal{X}_u \cup \mathcal{X}_v) \ \geq \ 0, \quad k \in \mathcal{K}_u, \quad u \in V(\mathcal{G}) \\ & x \in \{0,1\}^p \times [0,1]^{n-p}, \end{aligned}$$

where $\delta_{\mathcal{G}}(u)$ is the set of edges of $\mathcal{G}$ incident with u and each $p^{(k)}_{(u,v)}$ is a polynomial that only involves variables in $\mathcal{X}_u \cup \mathcal{X}_v$.

This class of problems not only generalizes the AC-OPF problem, but also other important optimization problems. Standard network flow problems fit in category [1], capacitated fixed-charge network flow models [56], the unit-commitment problem [91] and some optimization problems on gas networks [60]. The well-studied pooling problem [84] can also be cast as an NPO.

In this dissertation we will study the effect of low tree-width of the underlying network $\mathcal{G}$ defining an NPO. We will provide a way of obtaining LP approximations to it that can achieve any desired tolerance, moreover, if $\mathcal{G}$ possesses bounded tree-width, our LP approximation will be of moderate size. As a consequence, we will derive polynomial-size linear programs that approximate accurately the AC-OPF problem on networks that have bounded tree-width.

1.3 Structured Sparsity in General Polynomial Optimization

We will also study sparse *mixed-integer polynomial optimization problems*, that are not necessarily defined using a network as before. Here, sparsity will be measured using the tree-width parameter as well, however, the graph we assume has low tree-width will be constructed *from* a given formulation. More precisely, we will study problems of the form

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to:} \quad & f_i(x) \ \geq \ 0 \qquad 1 \leq i \leq m \\ & x \in \{0,1\}^p \times [0,1]^{n-p}. \end{aligned}$$

where each f_i is a polynomial. These are general Polynomial Optimization (PO) problems. We will develop a reformulation operator which relies on the combinatorial structure of the constraints to produce linear programming approximations which attain provable bounds. In order to include the graph-theoretical parameter *tree-width* in this optimization context, we will use the concept of *intersection graph*: a graph with vertices corresponding to variables, and that has an edge whenever two variables appear together in a constraint. We will show that a polynomial optimization problem whose intersection graph possesses low tree-width has a "small" extended linear formulation that approximates it. We will also argue on the difference between assuming sparsity of the intersection graph of a PO problem and assuming

sparsity in the underlying network that defines an NPO.

Polynomial optimization has recently seen a considerable growth in studies related to it. This is a long-studied field with beautiful theory, offering an appealing flexibility for modeling, and that recently has acquired much more maturity on the computational aspect. The celebrated hierarchies by Lasserre [66], for example, emerged as a compelling approach for solving polynomial optimization problems. Nevertheless, since they rely on SDP, scalability issues and numerical instability associated to current SDP solvers are frequent. For these reasons, some alternatives based on Linear Programming, or Second-order Cone Programming are arising.

From the tree-width-based sparsity perspective, there is broad literature dating from the 1980s on polynomial-time algorithms for combinatorial problems on graphs with bounded tree-width. An early reference is [6]. Also see [4, 5, 30, 17, 24, 19] and from a very general perspective, [27]. These algorithms rely on "nonserial dynamic programming", i.e., dynamic-programming on trees. See [3], [88], [18].

A parallel research stream concerns "constraint satisfaction problems", or CSPs. One can obtain efficient algorithms for CSPs, whenever the constraints present a sparse pattern given by low tree-width. These algorithms rely on similar dynamic programming ideas as the algorithms above, from the perspective of *belief propagation* on an appropriately defined *graphical model*. Another central technique is the tree-junction theorem of [72], which shows how a set of marginal probability distributions on the edges of a hypertree can be extended to a joint distribution over the entire vertex set. Early references are [92, 47, 41]. Also see [108, 34, 109] (and references therein).

In the integer programming context, extended formulations for binary linear programs whose constraints present intersection graphs of small tree-width have been developed in [20, 108, 70]. A different use of tree-width in integer programming is given in [38]. An alternative perspective on structural sparsity in optimization is taken in [29].

In this context, [21] (also see the PhD thesis [112]) develop extended formulations for

binary linear programs by considering the subset algebra of feasible solutions for individual constraints or small groups of constraints; this entails a refinement of the cone of set-functions approach of [77]. The method in [21] is similar to the one used here, in that here we rely on a similar algebra and on extended reformulations for 0/1 integer programs. The classical examples in this vein are the reformulation-linearization technique of [99], the cones of matrices method [77], the lift-and-project method of [10], and the moment relaxation methodology of [66]. See [69] for a unifying analysis; another comparison is provided in [8].

In [108], binary polynomial optimization problems are considered, i.e problems as

$$\min\{\, c^T x \,:\, x \in \{0,1\}^n,\ f_i(x) \geq 0,\ 1 \leq i \leq m\}$$

where each $f_i(x)$ is a polynomial. They show that if the tree-width of the intersection graph of the constraints is $\leq \omega$, then the level-ω Sherali-Adams or Lasserre reformulation of the problem is exact. Hence there is an LP formulation with $O(n^{\omega+2})$ variables and $O(n^{\omega+2}m)$ constraints.

A comprehensive survey of results on polynomial optimization and related topics is provided in [70]. Section 8 of [70] builds on the work in [69], which provides a common framework for the Sherali-Adams, Lovász-Schrijver and Lasserre reformulation operators. In addition to the aforementioned results, [70] explicitly shows that the special case of the vertex-packing problem on a graph with n vertices and tree-width $\leq \omega$ has a formulation of size $O(2^{\omega}n)$; this is stronger than the implication from [20] discussed above. Similarly, it is shown in [70] that the max-cut problem on a graph with n vertices and tree-width $\leq \omega$ has a formulation of size $O(2^{\omega}n)$.

In the continuous variable polynomial optimization setting, [63, 110] present methods for exploiting low tree-width of the intersection graph e.g. to speed-up the sum-of-squares or moment relaxations of a problem. Also see [53] and Section 8 of [70]. [65] shows that where ω is the tree-width of the intersection graph of a polynomial optimization problem, there is a hierarchy of semidefinite relaxations where the r^{th} relaxation ($r = 1, 2, \ldots...$) has $O(n\omega^{2r})$ variables and $O(n + m)$ LMI constraints; further, as $r \to +\infty$ the value of the relaxation converges to the optimum. Also see [89, 68].

Finally, there are a number of results on using lifted formulations for polynomial optimization problems, along the lines of the RLT methodology of [99]. See [101, 100] and references therein.

1.4 Cutting Plane Approaches to Polynomial Optimization

Besides showing the existence of moderate-sized LPs that can approximate sparse polynomial problems, and with a more pragmatic focus, we also aim to the development of computationally tractable techniques for Polynomial Optimization problems that can have an empirically good performance. For this purpose, we will make use of cutting planes algorithms, widely used in Mixed-Integer Programming, in a Polynomial Optimization context.

Cutting planes applied to non-convex problems are typically derived using problem-specific structures; either using particular assumptions on the data that drives a given optimization model, or tackling single non-linear terms or single constraints separately. In this work we pursue the development of general cutting plane techniques that rely on minimal assumptions on the problem structure.

We will explore two different families of cutting planes applicable to Polynomial Optimization: *Digitization cuts* and *Intersection cuts*. Digitization cuts make use of a discretization technique that can be traced back to [51]; also see [22, 39, 54]. Using this discretization, polynomials can be approximated accurately with a linear expression over binary variables, thus allowing MIP technology to be of use in generating cuts. We will present theoretical arguments for the validity of the inequalities and the quality of the cuts, and show heuristics that can speed-up the cut-finding algorithms.

As for Intersection cuts, we will make use of a reformulation of Polynomial Optimization problems that represents the feasible solutions (semi-algebraic sets) as $P \cap S$, where P is a polyhedron and S is a closed set. These cuts will be generated from convex forbidden zones, or S-free sets, according to the Intersection cuts introduced by Balas [9]. We review the work developed in [23], showing different families of S-free sets where cuts can be derived from. We will also provide details on the computational efficacy of these cuts.

There is a considerable amount of literature concerning cuts and strong linear relaxations that can be applicable to Polynomial Optimization problems. These cuts typically tackle single non-linear terms or single constraints, making use of substructures like edge-concavity [85, 103], multilinearity [79, 12, 94], or other special characteristics to derive convex envelopes and cuts. Also see [76, 105, 104]. All cuts we will present can account for several (or all) variables and functions in the problem simultaneously. To the best of our knowledge, two papers are similar in this regard. The disjunctive cuts of Saxena, Bonami, and Lee [97, 98] and the lift-and-project-based cuts, using moment relaxations, proposed Ghaddar, Vera, and Anjos [50].

1.5 Overview

This dissertation is organized as follows. In Chapter 2 we discuss the AC-OPF problem and our contributions towards fast bounding procedures. We begin by showing the commonly used models for Power Flows in Section 2.1, which are used in AC-OPF problem formulation presented in Section 2.2. In Section 2.3 we describe our approach in general terms as an introduction to Section 2.4, where we present in full detail the inequalities we developed for this problem. A tightening procedure of these inequalities is presented in Section 2.5, and in Section 2.6 we present experiments testing the performance of them.

In Section 2.7 we describe the AC transmission switching problem (ACTS), to then specify our proposed algorithm for it in Section 2.8, which makes use of our linear relaxation of AC-OPF. Finally, in Section 2.9 we provide computational experiments on the ACTS problem.

In the subsequent chapters, and motivated by the sparse structure present in transmission grids, we will provide a thorough study on tree-width-based sparsity in optimization problems. In Chapter 3 we discuss a general class of optimization problems over binary variables and analyze the effect of a low tree-width intersection graph. This will serve as the basic building block for all subsequent results. Here, we show how to obtain a *reformulation* of these binary

problems, where the size of the reformulation will be parametrized by the tree-width of the corresponding intersection graph. In Section 3.1 we provide a brief tutorial on the tree-width concept, along with the key results we will use. In Sections 3.2 through 3.5 we provide a preliminary discussion on the results we will prove, the necessary definitions and their consequences. Finally, in Sections 3.6 and 3.7 we provide two different LP reformulations that attain the desired effect.

Chapter 4 concerns Mixed-integer Polynomial Optimization problems that present an intersection graph with small tree-width. In Section 4.1 we state the family of problems we will tackle, as well as the main results of the chapter. In Section 4.2 we present the digitization technique that will allow us to approximate polynomial problems with pure binary problems. In Section 4.3 we show how to reformulate this pure binary approximation with an LP, analyzing what is the effect of low tree-width in this process. We conclude the chapter with Section 4.4, where we further discuss some aspects of our results and give a full detailed example of the LP approximation we obtain on a concrete problem.

In Chapter 5 we move to approximability results for Network Polynomial Optimization problems, which serve as a generalization of the AC-OPF problem since they have an underlying graph in the problem description, as opposed to the problems studied in Chapter 4, where a graph is drawn *from* a formulation. In Section 5.1 we formally define Network Polynomial Optimization problems and argue on the difference between assuming a sparse structure in the underlying graph and assuming the same structure on the intersection graph of a formulation. In Section 5.2 we outline the technique we will rely on to obtain tractability for these problems. We provide a complete example of this proof technique in Section 5.3, and then in Section 5.4 we provide the proof of the general case.

On a more pragmatic spirit, in Chapter 6 we discuss the two families of cutting planes mentioned above, meant for generating strong LP relaxations of Polynomial Optimization in a computationally effective manner. The first family, *Digitization cuts*, is presented in Section 6.1. These are cuts based on a digitization technique used for theoretical purposes in Chapter 4, and used here for generating cuts efficiently. The second family, *Intersection cuts*, which were were introduced in [23] in the polynomial optimization context, is discussed in Section

6.2. We review the key concepts behind them and show details on the computational aspects.

Finally, in Chapter 7 we present our concluding remarks, providing a general view on the proposed techniques of this dissertation and their impact, as well as future extensions of them.

1.6 Notation

We will denote as $\mathbb{R}$ and $\mathbb{Z}$ the set of real and integer numbers, respectively. We denote as $\mathbb{R}^n$ and $\mathbb{Z}^n$ the sets of n-dimensional vectors with coordinates in $\mathbb{R}$ and $\mathbb{Z}$, respectively. We use $\mathbb{S}^{n\times n}$ to denote the space of symmetric $n \times n$ matrices, and $\mathbb{S}^{n\times n}_+$ for the symmetric $n \times n$ matrices that are positive semidefinite. In some occasions we will use $[n]$ to denote the set $\{1, \ldots, n\}$.

Given a set S, we let $\text{int}(S)$ be its interior, $\text{bd}(S)$ its boundary, $\text{conv}(S)$ its convex hull, $\text{clconv}(S)$ the closure of its convex hull, $\text{cone}(S)$ its conic hull and $\text{clcone}(S)$ the closure of its conic hull. We use $\text{proj}(S)$ to denote the projection of S onto a lower dimensional space, which will be specified using a subscript. For example, if $\mathbb{V}$ is a subspace, then $\text{proj}_{\mathbb{V}}(S)$ will be the projection of S onto $\mathbb{V}$. In the case we distinguish different coordinates in S, for example, if we use (x, y) to refer to vectors in S, we use $\text{proj}_x(S)$ to denote the projection of S onto the x coordinates.

We use $|\cdot|$ to denote the absolute value, or magnitude, of the argument. When applied to a complex number (in the AC-OPF context), it denotes the complex number magnitude. $\|\cdot\|$ is used to denote the euclidean norm in $\mathbb{R}^n$ and $\|\cdot\|_1$ for the 1-norm. The inner product between two vectors u and v will be denoted as $u^T v$, where $(\cdot)^T$ is the transpose operator. We reserve $\leq$ ($<$) for component-wise (strict) inequality between vectors.

We let $\langle \cdot, \cdot \rangle$ be the Frobenius inner product of matrices

$$\langle A, B \rangle = \text{trace}(A^T B) = \sum_{i,j} A_{ij} B_{ij},$$

and $\|\cdot\|_F$ be the corresponding norm. We use $X \succ 0$ ($X \succeq 0$) whenever a matrix X is positive (semi) definite, and we let $\text{rank}(X)$ be its rank, i.e, the number of linearly independent rows

(or columns) of X. $X_{[i,j]}$ represents the principal submatrix of X induced by indices i and j, i.e,

$$X_{[i,j]} = \begin{bmatrix} X_{ii} & X_{ij} \\ X_{ij} & X_{jj} \end{bmatrix}.$$

For a polynomial $f(x)$, we will use the following representation. Given $\alpha \in \mathbb{Z}_+^n$ we write

$$x^\alpha \doteq \prod_{j=1}^{n} x_j^{\alpha_j}.$$

Then, we represent $f(x)$ as a sum of weighted monomials:

$$f(x) = \sum_{\alpha \in I(f)} f_\alpha x^\alpha,$$

where each f_α is rational and $I(f) \subseteq \mathbb{Z}_+^n$ is a finite set. We write $\|f\|_1 = \sum_{\alpha \in I(f)} |f_\alpha|$. The *degree* of f is defined as $\max_{\alpha \in I(f)} \sum_j \alpha_j$ and the *support* of f, denoted $\mathrm{supp}(f)$, is defined as the set of variables that appear explicitly in $f(x)$.

Chapter 2

AC Optimal Power Flow Problem

In this chapter we formally describe the AC-OPF problem, a key component in the operations of the power grid. This problem can be modeled using a non-convex optimization problem, and needs to be solved, in some form, every day, even every 10 minutes. Thus, it yields challenging optimization instances that are typically approximated in order to achieve tractability. We present details on the formulation of the problem, along with novel linear relaxations that can provide valid strong bounds quickly. We will also discuss how this efficient relaxation can be used on the AC Transmission Switching problem.

2.1 Modeling Power Flows on the Grid

The power grid can be represented as a network. We call the nodes of the network *buses* (that can be generators, loads or nodes where the power is redistributed) and the edges *lines*. Some buses will have a demand of power that must be satisfied, some buses will be able to generate power, and the power will flow through lines. Even though this setting is reminiscent of classical network-flow problems, extra difficulties arise from the way current flows in the lines.

Each bus k will have a *voltage* associated to it, given by a complex number V_k. Two variables e_k and f_k are used to represent the real and imaginary part of V_k, i.e,

$$V_k = e_k + jf_k$$

where $j = \sqrt{-1}$. Alternatively, one can use a polar representation of V_k given by

$$V_k = |V_k| e^{j\theta_k}$$

where θ_k is known as the *phase angle*. We will use both representations, depending on which is better suited for arguing a given statement.

In order to represent the lines of the network, we use the traditional π-model (see Figure 2.1). Consider a line $\{k, m\}$ between buses k and m; its *series impedance* $z_{\{k,m\}}$ is defined as

$$z_{\{k,m\}} = r_{\{k,m\}} + j x_{\{k,m\}}$$

where $r_{\{k,m\}}$ and $x_{\{k,m\}}$ are constants representing the line's *resistance* and *reactance*, respectively. A line's series admittance is given by

$$y_{\{k,m\}} \doteq z_{\{k,m\}}^{-1} = g_{\{k,m\}} + j b_{\{k,m\}}, \quad \text{where} \tag{2.1}$$

$$g_{\{k,m\}} = \frac{r_{\{k,m\}}}{r_{\{k,m\}}^2 + x_{\{k,m\}}^2} \quad \text{and} \quad b_{\{k,m\}} = -\frac{x_{\{k,m\}}}{r_{\{k,m\}}^2 + x_{\{k,m\}}^2}. \tag{2.2}$$

In addition, there will be a *shunt admittance* $y_{\{k,m\}}^{sh} = g_{\{k,m\}}^{sh} + j b_{\{k,m\}}^{sh}$, and a transformer (assuming it is located at the k side of the line) with ratio

$$\mathcal{N}_k \doteq \tau_k e^{j\sigma_k},$$

where τ_k and σ_k are constants representing the transformer's *magnitude* and *phase shift angle*, respectively.

We let I_{km} be the complex *current injection* of k in line $\{k, m\}$. P_{km} will represent the *active power* injected by k in line $\{k, m\}$ and Q_{km} will represent the *reactive power* injected by k in line $\{k, m\}$. Similarly, we define I_{mk}, P_{mk} and Q_{mk}. Defining

$$S_{km} = P_{km} + j Q_{km}$$

and using Ohm's Law, we have

$$I_{km} = \frac{1}{\tau_k} y_{\{k,m\}} \left[\frac{1}{\tau_k} V_k - e^{j\sigma_k} V_m \right] + \frac{1}{2\tau_k^2} y_{\{k,m\}}^{sh} V_k \tag{2.3}$$

$$I_{mk} = y_{\{k,m\}} \left[-\frac{1}{\tau_k e^{j\sigma_k}} V_k + V_m \right] + \frac{1}{2} y_{\{k,m\}}^{sh} V_m \tag{2.4}$$

$$S_{km} = V_k I_{km}^* \tag{2.5}$$

$$S_{mk} = V_m I_{mk}^*. \tag{2.6}$$

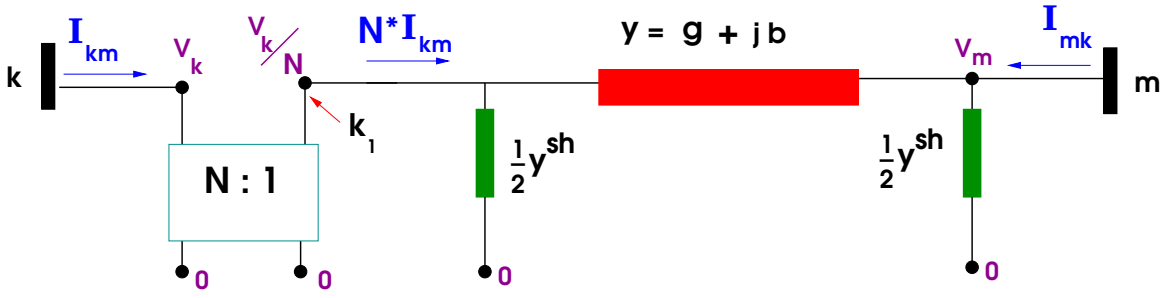


Figure 2.1: π-model of a line $\{k, m\}$, including transformer on the k side, and shunt admittance.

These are the so-called *power flow equations*. If we denote I the vector with components I_{km}, and V the vector with components V_k, equations (2.3) and (2.4) are typically written in summarized form as

$$I \;=\; \mathbb{Y}V,$$

where $\mathbb{Y}$ is referred to as the *branch admittance matrix*. This matrix is defined as follows: let $\mathbb{Y}_{\{k,m\}}$ be the submatrix of $\mathbb{Y}$ given by rows km and mk, and columns k and m, then

$$\mathbb{Y}_{\{k,m\}} \;=\; \begin{pmatrix} \left(y_{\{k,m\}} + \frac{y^{sh}_{\{k,m\}}}{2}\right)\frac{1}{\tau_k^2} & -y\frac{1}{\tau_k e^{-j\sigma_k}} \\ -y\frac{1}{\tau_k e^{j\sigma_k}} & y_{\{k,m\}} + \frac{y^{sh}_{\{k,m\}}}{2} \end{pmatrix}. \tag{2.7}$$

All other components of $\mathbb{Y}$ (i.e, entries that do not appear in $\mathbb{Y}_{\{k,m\}}$ for some line $\{k, m\}$) are 0. Now that we have defined the formulas for the power flow equations, we can move forward in properly defining the AC-OPF problem.

2.2 The AC-OPF Problem

The AC-OPF problem consists on determining the power to be generated in the generator buses and the appropriate voltages in each bus, in order to meet the demands of power throughout the grid and in a way that minimizes the generating costs and satisfies engineering and operational constraints.

The parameters of the model are given by

- The branch admittance matrix $\mathbb{Y}$ defined above.
- P_k^d (resp. Q_k^d) the active (reactive) load, or demand, at bus k.
- $P_k^{\max}$, $P_k^{\min}$, $Q_k^{\max}$, $Q_k^{\min}$ the active and reactive generator output limits in bus k. If bus k is not connected to a generator, then when we set

$$P_k^{\max} \;=\; P_k^{\min} \;=\; Q_k^{\max} \;=\; Q_k^{\min} \;=\; 0.$$

- $V_k^{\min}$ and $V_k^{\max}$ the voltage magnitude limits in bus k.

And the variables in the model are:

- For each line $\{k, m\}$ two (complex) variables associated to *current* I_{km} and I_{mk}, two (complex) variables associated to *active power* P_{km} and P_{mk}, and two (complex) variables associated to *reactive power* Q_{km} and Q_{mk}.
- For each bus k a (complex) variable for the voltage V_k, and additionally P_k^g (resp., Q_k^g) the active (or reactive) generation at k (which would be fixed at zero by the limits above, if no generator is connected to bus k).

For a bus k, we denote as δ_k the set of buses connected to k through a line. The AC-OPF problem can be represented using the following form (see [73], equations (2)):

$$\min \;\; F(P^g, Q^g) \tag{2.8a}$$

subject to:

$$\forall \,\text{bus}\, k : P_k^{\min} \le P_k^g \le P_k^{\max} \tag{2.8b}$$

$$Q_k^{\min} \le Q_k^g \le Q_k^{\max} \tag{2.8c}$$

$$V_k^{\min} \le \;\|V_k\| \le V_k^{\max} \tag{2.8d}$$

$$\sum_{m \in \delta_k} (P_{km} + jQ_{km}) \;=\; (P_k^g - P_k^d) + j(Q_k^g - Q_k^d) \tag{2.8e}$$

$$\forall \,\text{line}\, \{k, m\} : P_{km} + jQ_{km} = V_k I_{km}^* \tag{2.8f}$$

$$P_{mk} + jQ_{mk} = V_m I_{mk}^* \tag{2.8g}$$

$$I \;=\; \mathbb{Y}V. \tag{2.8h}$$

In (2.8a) F is a cost function, usually a sum of quadratics depending on the active power generation. Constraints (2.8b) and (2.8c) set the generation limits in each bus. Constraints (2.8d) indicate line voltage magnitude limits in each bus. Constraints (2.8e) are active and reactive balance constraints, i.e, they account for the power demanded on a bus and/or power generation limits on a bus. Constraints (2.8f)-(2.8h) capture the Power Flow equations described before. Note that (2.8) can be formulated using only the voltage variables V_k, nonetheless, we keep the other variables to simplify expressions.

Additionally, one can include limits on the flow over a line (given by the material properties of the line). These are typically convex constraints of the type $\|P_{km} + jQ_{km}\| \leq U_{\{k,m\}}$. We omit these to simplify the discussion, but they can easily be added to all results.

The non-convexity of this problem comes mainly from constraints (2.8f), (2.8g) and (2.8h) (also from (2.8d)), which we will see can be very complicated quadratic expressions. In the following section we will describe different ways of obtaining convex relaxations to these non-convex constraints.

2.3 Description of our approach

Here we focus on developing *linear relaxations* to AC-OPF problems, in *lifted spaces*, with the primary goal of quickly proving lower bounds and enabling fast, standard optimization methodologies, such as branching, to be used in this context. To motivate our approach, let $(P, Q, V^{(2)})$ be a vector that includes, for each line $\{k, m\}$, the real and reactive power injections P_{km}, P_{mk}, Q_{km} and Q_{mk}, and for each bus k the squared bus voltage magnitude $|V_k|^2$, denoted by $V_k^{(2)}$. Using these variables, we first write the AC-OPF (2.8) problem in the following summarized form

$$\min \;\; F(P, Q) \tag{2.9a}$$

subject to:

$$d^L \;\leq\; AP + BQ + CV^{(2)} \;\leq\; d^U \tag{2.9b}$$

$$(P, Q, V^{(2)}) \;\in\; \Omega. \tag{2.9c}$$

Here,

- In constraints (2.9b), A, B and C are matrices and d^L and d^U are vectors, all of appropriate dimension. These constraints describe basic relationships such as generator output limits, voltage limits, and active and reactive flow balance constraints. Clearly constraints (2.8d) fit in this category. Constraints (2.8b), (2.8c) and (2.8e) can be included here as well. Note that (2.8b) and (2.8e) can be combined to obtain a constraint of the form
$$P_k^L \;\leq\; \sum_{m \in \delta_k} P_{km} \;\leq\; P_k^U,$$
where P_k^L and P_k^U are given values. We can argue the same for constraints (2.8c).

- Constraints (2.9c) describe the underlying physics, e.g. Ohm's law. In formulation (2.8) these correspond to constraints (2.8f), (2.8g) and (2.8h). Note that in (2.9c) we do not explicitly state the dependency on I, since we can either fully replace these variables using (2.8h), or alternatively include them in the description of the set Ω.

- In standard OPF problem formulations, the objective $F(P, Q)$ is typically the sum of active power generation costs (summed over the generators); a separable convex quadratic function of the generator outputs.

Our basic approach will approximate (2.9c) with linear inequalities obtained by lifting formulation (2.9) to a higher-dimensional space. By 'lifting' we mean a procedure that adds new variables (with specific interpretations) and then writes inequalities that such variables, together with $(P, Q, V^{(2)})$, must satisfy in any feasible solution to the AC-OPF problem.

To fix our language, we view the quantities $P_{km}, P_{mk}, Q_{km}, Q_{mk}$ (for each line $\{k, m\}$) and $|V_k|^2$ (for each bus k) as *foundational*. All other variables, including those that arise naturally from constraint (2.9c) as well as those that we introduce, will be called *lifted*[1].

[1] Occasionally we may view the rectangular voltage coordinates as foundational.

In Section 2.4 we introduce our lifted variables, as well as the inequalities that we derive so as to obtain a convex relaxation of (2.9c). The inequalities described here will be of the following types:

1. Δ inequalities, (basic form given by (2.18))
2. (active power) loss inequalities, (basic form given by (2.22))
3. Circle inequalities, (basic form given by (2.28))

All these inequalities are convex; some linear and some conic. In the case of conic inequalities we rely on outer approximation through tangent cutting planes so as to ultimately obtain *linear* formulations as desired.

2.4 Valid inequalities for AC-OPF

In this section we derive valid inequalities, first for the simplest case (no shunt, no transformer) then for the case with shunts but no transformers, and finally for the most general case[2].

For ease of notation, we will omit the $\{k, m\}$ subscript in the line-related constants when the dependency is evident.

2.4.1 $y^{sh} = 0$ and $\mathcal{N} = 1$

In this case the equations in (2.8h) take the form

$$I_{km} \;=\; y(V_k - V_m). \tag{2.10}$$

Using rectangular coordinates this can be expressed as

$$I_{km} \;=\; g(e_k - e_m) - b(f_k - f_m) \,+\, j[\, b(e_k - e_m) + g(f_k - f_m)\,] \tag{2.11}$$

with a symmetric expression for I_{mk}. This implies

$$\begin{aligned} P_{km} \;&=\; e_k g(e_k - e_m) - e_k b(f_k - f_m) + f_k g(f_k - f_m) + f_k b(e_k - e_m) \\ &=\; (e_k - e_m)(g\,,\, b)\tbinom{e_k}{f_k} \;+\; (f_k - f_m)(-b\,,\, g)\tbinom{e_k}{f_k} \end{aligned} \tag{2.12}$$

[2]Bus shunt admittances are omitted, but can be easily incorporated into our inequalities.

with a symmetric expression for P_{mk}. Similarly,

$$\begin{aligned} Q_{km} &= f_k g(e_k - e_m) - f_k b(f_k - f_m) - e_k g(f_k - f_m) - e_k b(e_k - e_m) \\ &= (e_k - e_m)(-b\,,\, g)\tbinom{e_k}{f_k} + (f_k - f_m)(-g\,,\, -b)\tbinom{e_k}{f_k}. \end{aligned} \tag{2.13}$$

To obtain expressions in polar coordinates we write the impedance and admittance in polar coordinates:

$$z = |z|e^{j\angle z}, \quad y = \frac{1}{|z|}e^{-j\angle z}.$$

Then (see e.g. Bergen and Vittal [16], p. 104)

$$S_{km} = V_k I_{km}^* = V_k(V_k^* - V_m^*)y^* = \frac{|V_k|^2}{|z|}e^{j\angle z} - \frac{|V_k||V_m|}{|z|}e^{j\angle z}e^{j\theta_{km}}, \tag{2.14}$$

where

$$\theta_{km} \doteq \theta_k - \theta_m.$$

We also can rewrite (2.14) as

$$\begin{aligned} S_{km} &= |V_k|^2(g - jb) - |V_k||V_m|(g - jb)(\cos\theta_{km} + j\sin\theta_{km}) \\ &= |V_k|^2 g - |V_k||V_m| g\cos\theta_{km} - |V_k||V_m| b\sin\theta_{km} \\ &\quad + j\left[-|V_k|^2 b + |V_k||V_m| b\cos\theta_{km} - |V_k||V_m| b\sin\theta_{km}\right]. \end{aligned} \tag{2.15}$$

Likewise, the power received at m (rather than injected), $-S_{mk}$, satisfies

$$-S_{mk} = -\frac{|V_m|^2}{|z|}e^{j\angle z} + \frac{|V_k||V_m|}{|z|}e^{j\angle z}e^{-j\theta_{km}}. \tag{2.16}$$

We can also obtain an expression for S_{mk} similar to (2.15) by switching the k and m symbols. For these expressions in polar coordinates in the most general case, see the Appendix A.1.

2.4.1.1 Δ and loss inequalities, 1

Let μ_{km} and ν_{km} denote known upper bounds on

$$|(g\,,\, b)\tbinom{e_k}{f_k}| \quad \text{and} \quad |(-b\,,\, g)\tbinom{e_k}{f_k}|,$$

respectively. For example, from the Cauchy-Schwarz inequality, both quantities are upper-bounded by $\|(g,b)\|V_k^{\max}$, where $V_k^{\max}$ is an upper bound on $|V_k|$. Then, using (2.12) we obtain:

$$|P_{km}| \leq \mu_{km}|e_k - e_m| + \nu_{km}|f_k - f_m|. \tag{2.17}$$

Variables $|e_k - e_m|$ and $|f_k - f_m|$ will be represented using (nonnegative) lifted variables $d_{e,km}$, and $d_{f,km}$, thus obtaining the inequality

$$|P_{km}| \;\leq\; \mu_{km} d_{e,km} \,+\, \nu_{km} d_{f,km}. \tag{2.18}$$

This is the basic Δ inequality. Note that the vectors $\binom{g}{b}$ and $\binom{-b}{g}$ are of equal norm and orthogonal, so further elaborations of the Δ inequalities are possible. The upper bound $\|(g,b)\|V_k^{\max}$ on μ_{km} and ν_{km} appears loose because, when $(g,b) \neq 0$ and $\|V_k\| > 0$, it could not be the case that both bounds are simultaneously tight. However,

Lemma 2.4.1 *The Δ inequality*

$$|P_{km}| \;\leq\; \|(g,b)\|V_k^{\max} d_{e,km} \,+\, \|(g,b)\|V_k^{\max} d_{f,km} \tag{2.19}$$

is the best possible.

Here, by "best possible" what is meant is that one can produce examples such that if the coefficient of either $d_{e,km}$ or $d_{f,km}$ is tightened (i.e. decreased from $\|(g,b)\|V_k^{\max}$ to a smaller value) the resulting inequality becomes invalid – it cuts-off a feasible solution. However, the result should not be interpreted as saying that the inequality can *never* be tightened. In particular, if we consider a set of lines L and apply inequality (2.19) to each line $\{k,m\} \in L$, obtaining a system of linear inequalities, it may well be the case that the system itself can be tightened, that is to say, not all inequalities (2.19) for $\{k,m\} \in L$ can simultaneously hold as equations.

By adding the expression for P_{km} in (2.15) and the corresponding expression for P_{mk} we obtain

$$P_{km} + P_{mk} \;=\; g(|V_k|^2 + |V_m|^2) - 2g|V_k||V_m|\cos\theta_{km} \;=\; g|V_k - V_m|^2, \tag{2.20}$$

which can be *relaxed* as

$$g(e_k - e_m)^2 + g(f_k - f_m)^2 \;\leq\; P_{km} + P_{mk}, \tag{2.21}$$

or equivalently, using lifted variables,

$$g d_{e,km}^2 + g d_{f,km}^2 \;\leq\; P_{km} + P_{mk}, \tag{2.22}$$

We term (2.22) the *loss* inequality. Note that by definition $g \geq 0$ (unless by a modeling artifact we have $r < 0$), thus (2.22) is convex and a linear outer-approximation can be used for it.

The quantity $P_{km} + P_{mk}$ represents the *active power loss* on line $\{k, m\}$. When $g \geq 0$ (i.e. $r \geq 0$, the usual setting) it implies that losses are nonnegative. It is important to understand the connection between the Δ and the loss inequalities, which is highlighted by Theorem 2.4.4 given below, which may be stated in simplified form as "total active power generation equals total active power loads plus total losses." However, in order to obtain a precise statement (which is also valid in the cases where negative resistances occur) we proceed as follows.

Definition 2.4.2 *Let G be an undirected graph. A pseudo-flow is a vector P that assigns to each edge $\{k, m\}$ of G two reals, P_{km} and P_{mk}. For any node k of G we write*

$$\delta_k \doteq \text{set of nodes of } G \text{ adjacent to } k$$

and

$$o_k(P) \doteq \sum_{m \in \delta_k} P_{km}.$$

We call $o_k(P)$ the net output of k. We say that k is a **source** *if $o_k(P) > 0$ and is a* **sink** *if $o_k(P) < 0$. Likewise, an edge $\{k, m\}$ is termed a* **sink-edge** *(or* **source-edge***) if*

$$P_{km} + P_{mk} > 0 \quad \text{or, respectively} \quad P_{km} + P_{mk} < 0.$$

Remark 2.4.3

1. *When P denotes the (standard) vector of active power flows in a transmission system, $o_k(P) = G_k - D_k$ where G_k is generation at k and D_k is load at k. In general, we expect that generators will be sources and that sinks will be pure loads, though in principle one could have a generator bus k with $o_k(P) < 0$.*

2. *In the normal case of a transmission line we will have $r \geq 0$ and as discussed above line losses are non-negative. This means that no edge will ever be a source edge, but could be if negative resistances are used.*

Theorem 2.4.4 *Let G be a graph and P be a pseudo-flow on G. Then*

$$\sum_{k\,:\,o_k(P)>0} o_k(P) \;=\; \sum_{k\,:\,o_k(P)<0} (-o_k(P)) \;+\; \sum_{\{k,m\}} (P_{km}+P_{mk}).$$

Furthermore, P can be decomposed into directed flow paths, each originating at a source or source-edge and terminating at a sink or sink-edge.

A proof following standard flow conservation concepts is given in the Appendix A.2.

We can use this result to understand how the Δ inequalities work. If a vector P satisfies flow balance constraints, i.e. constraints at each bus k of the form

$$\sum_{km} P_{km} \;=\; G_k - D_k$$

some of the individual values P_{km} must be large enough (if the active power loads are nonzero). In the language of Definition 2.4.2, if some values $o_k(P)$ are nonzero we will necessarily have that some values P_{km} are also nonzero. If inequality (2.18) is enforced, it will cause the lifted variables $\delta_{e,km}$ and $\delta_{f,km}$ to be large enough. And in that case (2.22) implies that losses are appropriately large; hence by Theorem 2.4.4 total generation will (typically) have to be larger than the sum of loads.

2.4.1.2 Circle inequalities, 1

We can rewrite equation (2.14) as

$$S_{km} \;=\; C_{km} - B_{km}e^{j\theta_{km}} \quad \text{where} \tag{2.23}$$

$$C_{km} \;\doteq\; \frac{|V_k|^2}{|z|}e^{j\angle z} \text{ and} \tag{2.24}$$

$$B_{km} \;\doteq\; \frac{|V_k||V_m|}{|z|}e^{j\angle z}. \tag{2.25}$$

Note that C_{km} and B_{km} are obtained in the complex plane by rotating the real numbers $\frac{|V_m|^2}{|z|}$ and $\frac{|V_k||V_m|}{|z|}$ (respectively) by the same angle $\angle z$. As θ_{km} varies, (2.14) indicates that S_{km} describes a circle (the "sending circle") with center C_{km} and radius

$$\rho \doteq \frac{|V_k||V_m|}{|z|}.$$

Likewise, $-S_{mk}$ describes a circle (the "receiving circle") with center $-\frac{|V_m|^2}{|z|}$ and radius ρ. Refer to Bergen and Vittal [16] for more details. Using either circle we can obtain valid convex inequalities. For example, clearly we have

$$[\mathrm{Re}(S_{km} - C_{km})]^2 \;+\; [\mathrm{Im}(S_{km} - C_{km})]^2 \;\leq\; \rho^2, \tag{2.26}$$

or in other words,

$$\left(P_{km} - \frac{r|V_k|^2}{r^2+x^2}\right)^2 \;+\; \left(Q_{km} - \frac{x|V_k|^2}{r^2+x^2}\right)^2 \;\leq\; \frac{|V_k|^2|V_m|^2}{r^2+x^2}. \tag{2.27}$$

As discussed in Section 2.3, our formulation has variables used to represent $P_{km}, Q_{km}, |V_k|^2$ and $|V_m|^2$. Using these variables, from (2.27) we obtain a convex system by adding two lifted variables α_{km}, β_{km} and the constraints

$$P_{km} - \frac{rV_k^{(2)}}{r^2+x^2} \;=\; \alpha_{km} \tag{2.28a}$$

$$Q_{km} - \frac{xV_k^{(2)}}{r^2+x^2} \;=\; \beta_{km} \tag{2.28b}$$

$$\alpha_{km}^2 + \beta_{km}^2 \;\leq\; \frac{V_k^{(2)}V_m^{(2)}}{r^2+x^2}. \tag{2.28c}$$

We term (2.28) the *circle inequalities.* Constraints (2.28a) and (2.28b) are linear, and constraint (2.28c) is a rotated cone constraint.

Remark 2.4.5 *Many other inequalities can be obtained, in particular so as to bound the ratio P_{km}/Q_{km}, using the geometry of sending- and receiving-circles.*

2.4.2 General y^{sh} but $\mathcal{N} = 1$

In this case we have

$$I_{km} \;=\; y(V_k - V_m) \;+\; \frac{1}{2}y^{sh}V_k, \tag{2.29}$$

and so in rectangular coordinates

$$\begin{aligned} I_{km} \;=\;& g(e_k - e_m) - b(f_k - f_m) + \frac{1}{2}(g^{sh}e_k - b^{sh}f_k) \;+ \\ & j[\,b(e_k - e_m) + g(f_k - f_m) + \frac{1}{2}(b^{sh}e_k + g^{sh}f_k)\,]. \end{aligned} \tag{2.30}$$

This implies:

$$P_{km} = (e_k - e_m)(g\,,\,b)\binom{e_k}{f_k} + (f_k - f_m)(-b\,,\,g)\binom{e_k}{f_k} + \frac{g^{sh}}{2}(e_k^2 + f_k^2) \quad (2.31)$$

$$Q_{km} = (e_k - e_m)(-b\,,\,g)\binom{e_k}{f_k} + (f_k - f_m)(-g\,,\,-b)\binom{e_k}{f_k} - \frac{b^{sh}}{2}(e_k^2 + f_k^2). \quad (2.32)$$

Note that expressions in (2.31) and (2.32) are obtained from (2.12) and (2.13) by adding the terms $\frac{g^{sh}}{2}(e_k^2 + f_k^2)$ and $-\frac{b^{sh}}{2}(e_k^2 + f_k^2)$, respectively. In polar coordinates we will get

$$S_{km} = V_k I_{km}^* = V_k(V_k^* - V_m^*)y^* + \frac{1}{2}(g^{sh} - jb^{sh})V_k V_k^*. \quad (2.33)$$

2.4.2.1 Δ and loss inequalities, 2

Using (2.31), we obtain

$$|P_{km}| - \frac{g^{sh}}{2}V_k^{(2)} \leq \mu_{km}|e_k - e_m| + \nu_{km}|f_k - f_m|. \quad (2.34)$$

This is the second version of the Δ inequality. Since the right-hand side of (2.31) is obtained by adding $\frac{g^{sh}}{2}(e_k^2 + f_k^2)$ to the right-hand side of (2.12), we have the following analogue of (2.21):

$$g(e_k - e_m)^2 + g(f_k - f_m)^2 \leq P_{km} + P_{mk} - \frac{g^{sh}}{2}(V_k^{(2)} + V_m^{(2)}), \quad (2.35)$$

the second version of our loss inequality, which again is a conic constraint in the space of lifted variables.

2.4.2.2 Circle inequalities, 2

From (2.33) we get

$$S_{km} = |V_k|^2\left(\frac{e^{j\angle z}}{|z|} + \frac{1}{2}(g^{sh} - jb^{sh})\right) - \frac{|V_k||V_m|}{|z|}e^{j\angle z}e^{j\theta_{km}}, \quad (2.36)$$

which again describes a circle, with center and radius, respectively,

$$|V_k|^2\left(\frac{e^{j\angle z}}{|z|} + \frac{1}{2}(g^{sh} - jb^{sh})\right) \quad \text{and} \quad \frac{|V_k||V_m|}{|z|}. \quad (2.37)$$

Using (2.36), (2.37), and since

$$\mathrm{Re}\left((|V_k|^2\left(\frac{e^{j\angle z}}{|z|} + \frac{1}{2}(g^{sh} - jb^{sh})\right)\right) = |V_k|^2\left(\frac{r}{r^2 + x^2} + \frac{g^{sh}}{2}\right) \text{ and} \quad (2.38)$$

$$\mathrm{Im}\left((|V_k|^2\left(\frac{e^{j\angle z}}{|z|} + +\frac{1}{2}(g^{sh} - jb^{sh})\right)\right) = |V_k|^2\left(\frac{x}{r^2 + x^2} - \frac{b^{sh}}{2}\right), \quad (2.39)$$

we obtain the following generalization of the convex lifted system (2.28):

$$P_{km} - \left(\frac{r}{r^2+x^2} + \frac{g^{sh}}{2}\right) V_k^{(2)} = \alpha_{km} \tag{2.40a}$$

$$Q_{km} - \left(\frac{x}{r^2+x^2} - \frac{b^{sh}}{2}\right) V_k^{(2)} = \beta_{km} \tag{2.40b}$$

$$\alpha_{km}^2 + \beta_{km}^2 \leq \frac{V_k^{(2)} V_m^{(2)}}{r^2+x^2}. \tag{2.40c}$$

2.4.3 General y^{sh} and $\mathcal{N}$

In this case two approaches are possible.

- Indirect approach. Here we break up line $\{k, m\}$ into two separate lines, i.e. line $\{k, k_1\}$ and line $\{k_1, m\}$ (see Figure 2.1). We have that

$$V_{k_1} = \frac{1}{\tau}(e_k \cos\sigma + f_k \sin\sigma) + j\frac{1}{\tau}(f_k \cos\sigma - e_k \sin\sigma). \tag{2.41}$$

  which is an explicit linear inequality. Line $\{k_1, m\}$ can be separately handled using the approach in Section 2.4.2 so as to obtain Δ, loss and circle inequalities.

- Direct approach. This is the approach followed next.

From the general formula (2.5) we have (see the Appendix A.1 for details)

$$\begin{aligned} P_{km} = {} & \frac{1}{\tau}\left[\frac{e_k}{\tau} - e_m \cos\sigma + f_m \sin\sigma\right](g\,,\, b)\binom{e_k}{f_k} \\ & + \frac{1}{\tau}\left[\frac{1}{\tau} f_k - f_m \cos\sigma - e_m \sin\sigma\right](-b\,,\, g)\binom{e_k}{f_k} \\ & + \frac{g^{sh}}{2\tau^2}(e_k^2 + f_k^2). \end{aligned} \tag{2.42}$$

Similarly,

$$\begin{aligned} P_{mk} = {} & \left[e_m - \frac{1}{\tau} e_k \cos\sigma - \frac{1}{\tau} f_k \sin\sigma\right](g\,,\, b)\binom{e_m}{f_m} \\ & + \left[f_m - \frac{1}{\tau} f_k \cos\sigma + \frac{1}{\tau} e_k \sin\sigma\right](-b\,,\, g)\binom{e_m}{f_m} \\ & + \frac{g^{sh}}{2}(e_m^2 + f_m^2). \end{aligned} \tag{2.43}$$

2.4.3.1 Δ and loss inequalities, 3

In the transformer case there will be two Δ inequalities. The first is obtained by from (2.42) by taking absolute values:

$$|P_{km}| - \frac{g^{sh}}{2\tau^2}|V_k|^2 \ \leq$$
$$\frac{\mu_{km}}{\tau}\left|\frac{1}{\tau}e_k - e_m\cos\sigma + f_m\sin\sigma\right| \ + \ \frac{\nu_{km}}{\tau}\left|\frac{1}{\tau}f_k - f_m\cos\sigma - e_m\sin\sigma\right|. \qquad (2.44)$$

Here as before μ_{km} and ν_{km} are known upper bounds on $|(g\,,\,b)\binom{e_k}{f_k}|$ and $|(-b\,,\,g)\binom{e_k}{f_k}|$, respectively. Similarly, we obtain a second Δ inequality from (2.43):

$$|P_{mk}| - \frac{g^{sh}}{2\tau^2}|V_m|^2 \ \leq$$
$$\mu_{mk}\left|e_m - \frac{1}{\tau}e_k\cos\sigma - \frac{1}{\tau}f_k\sin\sigma\right| \ + \ \nu_{mk}\left|f_m - \frac{1}{\tau}f_k\cos\sigma + \frac{1}{\tau}e_k\sin\sigma\right|. \qquad (2.45)$$

Thus in order to represent these inequalities we need to introduce additional lifted variables used to model the expressions

$$\left|\frac{1}{\tau}e_k - e_m\cos\sigma + f_m\sin\sigma\right| \quad , \quad \left|e_m - \frac{1}{\tau}e_k\cos\sigma - \frac{1}{\tau}f_k\sin\sigma\right|$$
$$\left|\frac{1}{\tau}f_k - f_m\cos\sigma - e_m\sin\sigma\right| \quad , \quad \left|f_m - \frac{1}{\tau}f_k\cos\sigma + \frac{1}{\tau}e_k\sin\sigma\right|.$$

In the no-transformer case the first two variables are equal to $|e_m - e_k|$ and the last two are equal to $|f_m - f_k|$. Replacing, in (2.44) and (2.45), $|V_k|^2$ and $|V_m|^2$ with $V_k^{(2)}$ and $V_m^{(2)}$ respectively, we obtain the most general form of the Δ inequalities.

Since all losses are incurred in the section of the line between k_1 and m, applying (2.35) and (2.41) we obtain:

$$g\left(e_m - \frac{1}{\tau}e_k\cos\sigma - \frac{1}{\tau}f_k\sin\sigma\right)^2 + g\left(f_m - \frac{1}{\tau}f_k\cos\sigma + \frac{1}{\tau}e_k\sin\sigma\right)^2$$
$$\leq \ P_{km} + P_{mk} - \frac{g^{sh}}{2}\left(\frac{V_k^{(2)}}{\tau^2} + V_m^{(2)}\right). \qquad (2.46)$$

In this form we obtain a convex inequality that employs the lifted variables introduced in (2.45). A similar construction yields an inequality using the lifted variables in (2.44), since

$$\left(e_m - \frac{1}{\tau}e_k\cos\sigma - \frac{1}{\tau}f_k\sin\sigma\right)^2 + \left(f_m - \frac{1}{\tau}f_k\cos\sigma + \frac{1}{\tau}e_k\sin\sigma\right)^2$$
$$= \ \left(\frac{1}{\tau}e_k - e_m\cos\sigma + f_m\sin\sigma\right)^2 + \left(\frac{1}{\tau}f_k - f_m\cos\sigma - e_m\sin\sigma\right)^2.$$

These inequalities represent a generalized version of the *loss inequalities*. For a complete and detailed derivation of (2.46) see Appendix A.1.

2.4.3.2 Circle inequalities, 3

In the transformer case the structure of the circle inequalities differs due to the asymmetry caused by the transformer. First, the system (2.40) applied at m is unchanged (i.e. system (2.40) with k and m interchanged). To obtain a system at k we again consider point k_1 in Figure 2.1 and we now obtain:

$$P_{km} - \left(\frac{r}{r^2+x^2} + \frac{g^{sh}}{2}\right)\frac{V_k^{(2)}}{\tau^2} = \alpha_{km} \tag{2.47a}$$

$$Q_{km} - \left(\frac{x}{r^2+x^2} - \frac{b^{sh}}{2}\right)\frac{V_k^{(2)}}{\tau^2} = \beta_{km} \tag{2.47b}$$

$$\alpha_{km}^2 + \beta_{km}^2 \leq \frac{V_k^{(2)} V_m^{(2)}}{\tau^2(r^2+x^2)}. \tag{2.47c}$$

2.5 Tightening inequalities through reference angle fixings

Above we introduced a family of inequalities for each line of the underlying network. Here we will describe a tightening procedure that can render significant improvements.

Recall the discussion in Section 2.4 regarding foundational and lifted variables. The lifted variables include e.g. a variable used to represent $\left|e_m - \frac{1}{\tau}e_k\cos\sigma - \frac{1}{\tau}f_k\sin\sigma\right|$ introduced in equation (2.45).

We can express these facts in compact form as follows. As in Section 2.4, let $(P, Q, V^{(2)})$ indicate the vector of all foundational variables. Here, for each bus k variable $V_k^{(2)}$ is used to represent the quantity $|V_k|^2$. If N and M indicate the number of buses and lines, respectively, then $(P, Q, V^2) \in \mathbb{R}^{2M+N}$. Let W indicate the vector of all lifted variables, say with H components, and let $K \subseteq \mathbb{R}^{2M+N+H}$ indicate the convex set described by all inequalities introduced in Section 2.4. Then we can represent $(P, Q, V^{(2)}, W) \in K$ more compactly by stating that

$$(P, Q, V^{(2)}) \in \hat{K} \doteq \text{proj}_{\mathbb{R}^{2M+N}} K \tag{2.48}$$

where $\text{proj}_{\mathbb{R}^{2M+N}} K$ is the projection of K to the subspace of the first $2M + N$ variables.

We now describe a procedure for tightening (2.48). As is well known, fixing an arbitrary bus at an arbitrary angle does not change the set of feasible solutions to a standard AC-OPF problem. Thus, let $\hat{k}$ be a particular bus, and let $\hat{\theta}_{\hat{k}}$ be a particular angle; we can therefore without loss of generality fix $\theta_{\hat{k}} = \hat{\theta}_{\hat{k}}$. How can we take advantage of this fact so as to obtain stronger constraints? Trivially, we can of course enforce $f_{\hat{k}} = \tan \hat{\theta}_{\hat{k}} e_{\hat{k}}$.

Moreover, consider for example the Δ inequality (2.18) for a line $\hat{k}m$ (for simplicity we assume the line has zero shunt admittance and no transformer). We repeat the constraint here for convenience:

$$|P_{\hat{k}m}| \;\; \leq \;\; \mu_{\hat{k}m}|e_{\hat{k}} - e_m| \, + \, \nu_{\hat{k}m}|f_{\hat{k}} - f_m|, \tag{2.49}$$

where $\mu_{\hat{k}m}$ and $\nu_{\hat{k}m}$ are valid upper bounds on

$$|(g\,,\,b)\tbinom{e_{\hat{k}}}{f_{\hat{k}}}| \quad \text{and} \quad |(-b\,,\,g)\tbinom{e_{\hat{k}}}{f_{\hat{k}}}|,$$

respectively. As stated above, both b and g depend on the line but we omit the dependency for simplicity of notation. Given that we know $\theta_{\hat{k}} = \hat{\theta}_{\hat{k}}$ we can tighten the estimates on $\mu_{\hat{k}m}$ and $\nu_{\hat{k}m}$, thereby obtaining a tighter inequality from (2.49). We can likewise tighten many of the inequalities introduced above.

More generally, suppose that rather than fixing $\theta_{\hat{k}}$ to a fixed value, we insist that it is contained in a known set $\mathcal{I}(\hat{k})$ (in particular an interval), i.e.

$$\theta_{\hat{k}} \, \in \, \mathcal{I}^{(\hat{k})}$$

As just argued we can therefore *without loss of generality*, tighten the valid inequalities we described in previous section. This tightening is easiest in the case where the set is in fact an interval.

Let $K\left(\hat{k}, \mathcal{I}^{(\hat{k})}\right) \subseteq \mathbb{R}^{2M+N+H}$ denote the resulting convex body, and let

$$\Pi\left(\hat{k}, \mathcal{I}^{(\hat{k})}\right) \doteq \text{proj}_{\mathbb{R}^{2M+N}} K\left(\hat{k}, \mathcal{I}^{(\hat{k})}\right).$$

As a consequence of the above observations, we now formally have:

Lemma 2.5.1 *Suppose $(\tilde{P}, \tilde{Q}, \tilde{V}^{(2)})$ is feasible for the AC-OPF problem. Then for any bus $\hat{k}$, and any set $\mathcal{I}^{(\hat{k})}$,*

$$(\tilde{P}, \tilde{Q}, \tilde{V}^{(2)}) \;\in\; \Pi\left(\hat{k}, \mathcal{I}^{(\hat{k})}\right). \tag{2.50}$$

Of course one can simply enforce (2.50) by explicitly writing down all the lifted variables and all the constraints used to describe the set $K\left(\hat{k}, \mathcal{I}^{(\hat{k})}\right)$. Alternatively, one can *separate* from the convex set $\Pi\left(\hat{k}, \mathcal{I}^{(\hat{k})}\right)$ and use such cuts as cutting planes. From this perspective, the following result is important:

Corollary 2.5.2 *Suppose* $(\tilde{P}, \tilde{Q}, \tilde{V}^{(2)})$ *is feasible for the AC-OPF problem. Then for any family of buses* k_i *(*$i \in F$*) and sets* $\mathcal{I}^{(k_i)}$ *we have*

$$(\tilde{P}, \tilde{Q}, \tilde{V}^{(2)}) \in \bigcap_{i \in F} \Pi\left(k_i, \mathcal{I}^{(k_i)}\right). \tag{2.51}$$

In other words, in particular, we can separate a given vector $(\tilde{P}, \tilde{Q}, \tilde{V}^{(2)})$ from sets obtained from our original family of valid inequalities by e.g. fixing one arbitrary bus to an arbitrary angle, and tightening.

2.6 Computational experiments

In the experiments reported here, we implemented the Δ, loss and circle inequalities in their most general form. To solve conic and linear programs, we used Gurobi 5.6.3 [58]. To solve semidefinite programs, we used the system due to Lavaei et al. [80], which also includes a procedure for extracting a feasible rank-one solution from the SDP. All runs were performed on a current workstation with ample physical memory. All running times are in seconds unless indicated.

In Table 2.1, "SDP time" is the time taken to solve the SDP relaxation of the OPF problem, "SDP gap" is the percentage gap between the value of the SDP relaxation and the upper bound (value of feasible solution) obtained by the SDP system. "SOCP time" and "LP time", are, respectively, the time required to solve our conic relaxation and its first-order (outer) relaxation through a cutting-plane algorithm. "SOCP gap" and "LP gap" are the percentage gaps relative to the SDP upper bound.

From Table 2.1 we can see the big potential of using LP technology. We are able to obtain competitive gaps in reduced time. In the smaller instances we are not able to improve on the SOCP time; this behavior was expected, as commercial optimization software sometimes follow the same approach we used for dealing with SOCP constraints (using linear approx-

	SDP time	SDP gap	SOCP time	SOCP gap	LP time	LP gap
case9	1.04	0.0002 %	0.05	0.7899 %	0.04	0.7899 %
case30	3.40	0.0185 %	0.23	1.3808 %	0.35	1.3964 %
case57	4.23	0.0000 %	0.62	0.9954 %	1.41	0.9954 %
case118	8.73	0.0045 %	0.98	1.4645 %	5.12	1.4642 %
case300	20.29	0.0018 %	4.62	1.0585 %	49.61	1.0559 %
case2383wp	13 min	0.6836 %	2 min	3.6134 %	1.63	5.6489 %
case2746wp	16 min	0.0375 %	79.10	1.8593 %	1.88	3.1235 %

Table 2.1: Comparison of different relaxations of AC-OPF.

imations). For large instances we are able to obtain a much smaller running time, to the expense of a slightly weaker gap.

The results are nevertheless encouraging, as they provide indications that LPs can still be used in challenging non-convex settings such as the AC-OPF problem.

2.7 AC transmission switching problem

We now turn to a different power-grid-based problem, where the aforementioned relaxations can be used with promising results. This is the so-called AC transmission switching problem.

In the AC transmission switching problem (ACTS for short) a planner seeks to switch off transmission lines with the goal of reducing transmission cost, improving congestion, carrying out line maintenance, or a number of other reasons. In addition, the planner may seek to enforce additional constraints on the set of switched off lines, such as only allowing a specific subset of lines to be switched off and placing upper or lower bounds on the quantity of switched off lines. See [45], [55] for results and background. Network modeling relies on an AC power flow model; thus ACTS is a nonlinear, non-convex, mixed-integer optimization problem.

In the following sections we describe a methodology for addressing ACTS problems that borrows from ideas in traditional mixed-integer optimization methods. In designing an algorithm for ACTS along these lines, one would start with an effective convex relaxation

for ACTS that yields good (lower) bounds. However, care needs to be exercised in designing such a relaxation so that it can be leveraged by the standard set of tools for solving non-continuous optimization problems: disjunctions, branching, formulation tightening (e.g. dynamically developed cutting planes) and repeated solutions. In this context, it is important to stress that when addressing truly difficult non-continuous optimization problems one often needs iterative algorithms that require repeated solutions of progressively modified relaxations, as opposed to feeding a single, static problem formulation to a generic solver.

In what follows, we introduce an iterative method for tackling ACTS, at the core of which is the lightweight relaxation of AC-OPF presented above. In order to handle a simplified version of our relaxation to the AC-OPF problem for ease of notation, we will assume no transformer is present and that $y^{sh} = 0$. The general case follows directly.

2.8 Formulation and algorithm for ACTS

The ACTS problem has a similar structure as AC-OPF; however for each line will we have a binary variable used to model the decision to switch off that line. We model line switching using a binary variable $s_{\{k,m\}}$ for each line $\{k, m\}$. We include the constraints

$$\|P_{km} + jQ_{km}\| \leq M_{\{k,m\}}(1 - s_{\{k,m\}})$$

where $M_{\{k,m\}}$ is either the line's limit $U_{\{k,m\}}$ (if present), or a large number. We also modify the last constraint of the circle system (2.28) to state

$$-\frac{[V_k^{max}]^4 - [V_k^{min}V_m^{min}]^2}{r^2 + x^2} s_{\{k,m\}} + \alpha_{km}^2 + \beta_{km}^2 \quad \leq \quad \frac{V_k^{(2)} V_m^{(2)}}{r^2 + x^2}. \tag{2.52}$$

When $s_{\{k,m\}} = 0$ (line not switched) (2.52) coincides with (2.28c). When $s_{\{k,m\}} = 1$, equations (2.28a)-(2.28b) and $P_{km} = Q_{km} = 0$ give

$$\alpha_{km}^2 + \beta_{km}^2 = \frac{\left(V_k^{(2)}\right)^2}{r^2 + x^2}$$

which shows the validity of (2.52). The following is a valid relaxation for ACTS:

$$\min F(P,Q) \tag{2.53a}$$

subject to:

$$\forall \text{ bus } k: P_k^L \;\le\; \sum_{m\in\delta_k} P_{km} \;\le\; P_k^U \tag{2.53b}$$

$$Q_k^L \;\le\; \sum_{m\in\delta_k} Q_{km} \;\le\; Q_k^U \tag{2.53c}$$

$$\left(V_k^{\min}\right)^2 \le\; V_k^{(2)} \le (V_k^{\max})^2 \tag{2.53d}$$

$$\forall \text{ line } \{k,m\}: s_{\{k,m\}} \;\in\; \{0,1\} \tag{2.53e}$$

$$\|P_{km} + jQ_{km}\| \le M_{km}(1 - s_{\{k,m\}}) \tag{2.53f}$$

$$\Delta \text{ inequality (2.19), loss inequality (2.22)} \tag{2.53g}$$

$$\text{(2.28a), (2.28b), (2.52),} \tag{2.53h}$$

$$\text{side-constraints on the } s_{\{k,m\}} \text{ variables.} \tag{2.53i}$$

Constraint (2.53i) will be defined from the extra requirements of the line switching instance, e.g, it can represent the maximum number of lines that can be switched off, or a particular set of lines from where to choose from. Note that the Δ and loss constraints are enforced even when line $\{k,m\}$ is switched off. For convenience of the reader, we restate these constraints:

$$|P_{km}| \;\le\; \|(g,b)\|V_k^{max} d_{e,km} + \|(g,b)\|V_k^{max} d_{f,km} \tag{2.54}$$

$$g\, d_{e,km}^2 + g\, d_{f,km}^2 \;\le\; P_{km} + P_{mk}. \tag{2.55}$$

The critical observation is that, even though in our lifted relaxation for AC-OPF the variables $d_{e,km}$ and $d_{f,km}$ modeled the rectangular voltage deviations $|e_k - e_m|$ and $|f_k - f_m|$, variables $d_{e,km}$ and $d_{f,km}$ appear only in the two inequalities listed above. Hence, subject to satisfying (2.54) and (2.55), $d_{e,km}$ and $d_{f,km}$ can take arbitrary values. In particular, when $s_{\{k,m\}} = 1$ (line switched off), setting $d_{e,km} = d_{f,km} = 0$ will satisfy (2.54)-(2.55). In summary:

Theorem 2.8.1 *Formulation* (2.53) *is a valid relaxation for ACTS, and as a consequence its value is a lower bound on the value of ACTS.*

2.8.1 Non-standard branching for ACTS

Formulation (2.53) only proves a lower bound on the value of ACTS. To obtain upper bounds, we use a non-standard branch-and-bound procedure which relies on any fast algorithm for computing *upper bounds* for AC-OPF; see e.g. [32]. The root node for our branching method is given by formulation (2.53). A typical node will be endowed with an extension of formulation (2.53), and is processed by the following template ($\mathcal{L}$ is the set of all lines):

PROCESSING NODE v OF BRANCH-AND-BOUND

1. Solve the formulation at v. Let K^v the set of lines switched off in the solution.

2. Run the AC-OPF upper bounding procedure with set K^v switched off. If feasible, we obtain an upper bound for ACTS as well.

3. If not, add to the branch-and-bound tree two nodes:

$\rightarrow$ The first extends the formulation at v with:

$$\sum_{\{k,m\}\in K^v} s_{\{k,m\}} \leq |K^v| - 1.$$

$\rightarrow$ The second extends the formulation at v with:

$$\sum_{\{k,m\}\in K^v} s_{\{k,m\}} = |K^v|.$$

$$\sum_{\{k,m\}\in \mathcal{L}} s_{\{k,m\}} \geq |K^v| + 1.$$

The two nodes added to the branch-and-bound tree in Step 3 guarantee correct enumeration; the second one is justified by the observation that switching off a set K of lines may prove infeasible, but a super-set of K could prove feasible. As branch-and-bound iterates it will produce both upper bounds and rigorous lower bounds for ACTS[3]. Any feasible solution (Step 2) will furnish an upper bound for ACTS and a corresponding set of lines to switch off.

[3]Subject to round-off errors by the solver.

H	MIP time	MIP value	Nodes	Feas. value
5	192 s	1804930.39	1	1868524.14
6	265 s	1804981.21	1	1868585.69
10	190 s	1805024.07	1	1868760.04

Table 2.2: ACTS algorithm on case2383wp

2.9 Computational experiments for ACTS

In our numerical experiments with the above algorithm we consider ACTS with a side-constraint (2.53i) stating that for some given value H, a *minimum* of H lines must be switched off, indicating a maintenance or testing schedule. This constraint can be expressed as

$$\sum_{\{k,m\}\in\mathcal{L}} s_{\{k,m\}} \geq H.$$

We ran branch-and-bound until the first feasible solution was found.

Table 2.2 shows results using *case2383wp*, using Gurobi to solve the mixed-integer program in Step 1. These experiments required a single node of branch-and-bound to terminate. In the Table, "MIP time" and "MIP value" indicate the time needed to solve the problem in Step 1 and its value (i.e., K^1). "Nodes" is the number of iterations taken by the cutting-plane method. "Feas. value" is the value of the AC-OPF problem reported by MATPOWER [111], which we used to handle Step 2 of the algorithm, typically taking just a few seconds of CPU time.

Note that *case2383wp* has more than 2800 lines; and thus, even in the case $H = 5$, complete enumeration of all subsets with exactly $|H|$ lines is impractical.

Table 2.3 concerns *case2746wp* with $H = 5$. The column headed "Status" displays INF when the AC-OPF problem in Step 2 of the template above is infeasible; and otherwise it shows the solution value.

In this case two nodes of branch-and-bound are needed. Some round-off error from the MIP solver is noted in the sixth digit of the solution obtained in the second node.

Node 1 (the root)			
MIP time	**MIP value**	**Lines**	**Status**
114 s	1601139.20	270, 1246, 1262 1517, 3016	**INF**
Node 2			
MIP time	**MIP value**	**Lines**	**Status**
135 s	1601085.68	3163, 3374, 3439 3492, 3500	**1631760.80**

Table 2.3: ACTS algorithm on case2746wp with ≥ 5 switched lines

Chapter 3

Binary Optimization with small tree-width

In the following chapters we will focus on obtaining tractable methods to structurally sparse optimization problems. Motivated by the AC-OPF problem discussed in the previous chapter and the *tree-like* topology typically observed in transmission networks, tractability will be measured by *tree-width*; a graph theoretical parameter used to roughly measure how "tree-like" a given graph is.

In this chapter we begin by introducing the *tree-width* concept to the reader, along with key notions surrounding it in order to cement the foundations for this chapter. Next, we will study a general class of binary optimization problems and prove how tree-width-based sparsity can be used to tackle them effectively. The pure binary case will be the building block toward the other more general results in the subsequent chapters.

3.1 A brief tutorial on tree-width

In what follows, given an undirected graph H, we will use $V(H)$ and $E(H)$ to denote the vertex-set and edge-set of H, respectively, and $\delta_H(u)$ will be the set of edges incident with vertex u.

Definition 3.1.1 *Let G be an undirected graph. A* tree-decomposition *[95, 96] of G is a pair (T, Q) where T is a tree and $Q = \{Q_t : t \in V(T)\}$ is a family of subsets of $V(G)$ (the*

vertices of G) such that

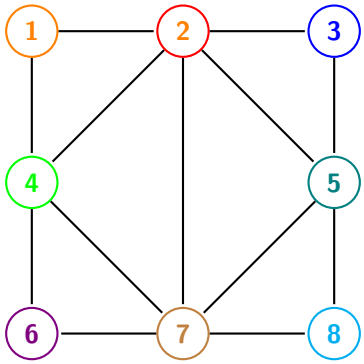
(i) For all $v \in V(G)$, the set $\{t \in V(T) : v \in Q_t\}$ forms a subtree T_v of T, and

(ii) For each $\{u, v\} \in E(G)$ there is a $t \in V(T)$ such that $\{u, v\} \subseteq Q_t$, i.e. $t \in T_u \cap T_v$.

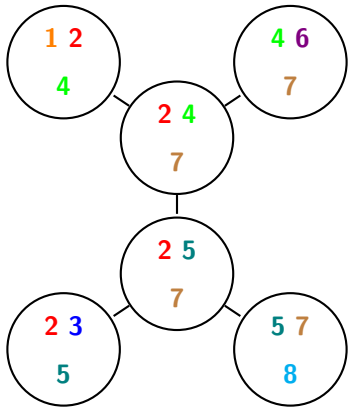
(iii) $\bigcup_{t \in V(T)} Q_t = V(G)$.

The width *of the decomposition is defined as* $\max\{|Q_t| : t \in V(T)\} - 1$. *The* tree-width *of G is the minimum width over all tree-decompositions of G.*

Example 3.1.2 *Let G be defined as*



A valid tree-decomposition, with the sets Q_t indicated inside each node of the tree, is as follows:



The width of this tree-decomposition is 2.

In some cases we will refer to the sets Q_t as "bags", as they consists on sets of nodes of the original graph.

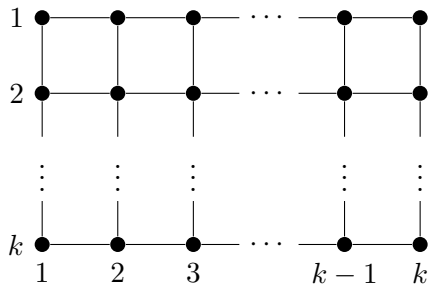


Figure 3.1: Square grid formed by k colums and k rows of nodes.

Tree-width, roughly speaking, indicates how "tree-like" a graph is. Trees are the graphs with tree-width 1, cycles have tree-width 2 and cliques on n nodes have tree-width $n-1$, to mention a few. It can be shown that a graph with tree-width ω and n vertices has $O(\omega n)$ edges, and so graphs of small tree-width are sparse. However, not all sparse graphs have small tree-width. One well known example of this is the $k \times k$ grid given in Figure 3.1. This graph is sparse in the usual sense, i.e, it posseses a low number of edges (they scale linearly with the number of nodes), however its tree-width is k.

An alternative definition of tree-width the reader might find useful is the following:

Definition 3.1.3 *A graph G is said to have tree-width at most ω if and only if G has a chordal super-graph with clique number $\omega + 1$.*

A chordal super-graph of G is sometimes referred to as *chordal completion* of G.

Determining if a given graph has tree-width at most ω, with ω variable, is NP-hard [7]. For ω fixed, however, tree-width ω can be recognized in linear time [25]. In terms of finding approximations to the tree-width of a graph, please see the recent work [26] and references therein. We also refer the reader to [19, 24, 5, 44] for additional background.

Besides its *width*, another important feature of a tree-decomposition (T, Q) we will use is the *size* of the tree-decomposition, given by $|V(T)|$. It was recently proven that, given a graph H of width at most ω, computing a tree-decomposition (T, Q) of width ω that minimizes $|V(T)|$ is NP-hard in the class of graphs with tree-width at most ω [75]. However, for our purposes the following well known result will suffice.

Proposition 3.1.4 *Given a graph G with tree-width at most ω, then there exists a tree-decomposition (T, Q) of G of width ω such that*

$$|V(T)| = O(|V(G)|)$$

Another important tree-decomposition result we will use is given in the following Remark.

Remark 3.1.5 *Suppose (T, Q) is a tree-decomposition of some graph G, and let H be a connected subgraph of G. Then the set of vertices t of T such that Q_t intersects $V(H)$ forms a subtree of T.*

Remark 3.1.5 follows directly from property (i) in Definition 3.1.1. And finally, a key property relating cliques to the bags in a given tree-decomposition is given by:

Proposition 3.1.6 *Consider a graph G and a tree-decomposition (T, Q) of G. Then for every clique $K \in V(G)$, there exists $t \in T$ such that*

$$K \subseteq Q_t.$$

Proposition 3.1.6 is a standard result of graph theory, and not only it provides a condition bags must satisfy, it also shows a direct lower bound for the tree-width of a graph.

We now move to the optimization context, where the graph-theoretical tools we just introduced will be used to measure how sparse a problem is.

3.2 Problem description

We will study "general" binary problems, or **GB** for short, defined as follows.

$$(\mathbf{GB}): \quad \min \ c^T x \tag{3.1a}$$

$$\text{subject to:} \quad x_{K_i} \in S^i \qquad 1 \leq i \leq m \tag{3.1b}$$

$$x \in \{0,1\}^n. \tag{3.1c}$$

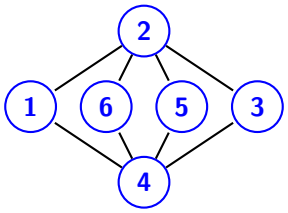


Figure 3.2: Intersection Graph for system 3.2

For $1 \leq i \leq m$, constraint i is characterized by a subset $K_i \subseteq \{1, \ldots, n\}$ and a set $S^i \subseteq \{0,1\}^{K_i}$. Set S^i is implicitly given by a *membership oracle*, that is to say a mechanism that upon input $y \in \{0,1\}^{K_i}$, truthfully reports whether $y \in S^i$ or not.

Any linear-objective, binary optimization problem whose constraints are explicitly stated can be recast in the form **GB**; e.g., each set S^i could be described by a system of algebraic equations in the variables x_j for $j \in K_i$. **GB** problems are related to classical constraint satisfaction problems, however the terminology above will prove useful later.

The link between problems of the form **GB** and the sparsity structure defined in Section 3.1, i.e *tree-width*, will be given by the concept of *intersection graph*, which provides a way of using this graph-theoretical parameter in an optimization context.

Definition 3.2.1 *The* **intersection graph** *[48] for a system of constraints is the undirected graph which has a vertex for each variable and an edge for each pair of variables that appear in any common constraint.*

Example 3.2.2 *Consider the following system of constraints on variables* $x_1, \ldots, x_6$*:*

$$3x_1^2 - x_2 \geq 0, \qquad -2x_2^2 + x_3^3 \geq 1, \qquad x_2 + x_6 = 1, \qquad x_4 - x_5^3 \leq 0, \tag{3.2a}$$

$$x_3^3 - x_4^2 \leq 2, \qquad x_1 + x_4 \leq 0, \qquad x_2 + x_5 \geq 0, \qquad x_5^2 - x_4^2 = 0. \tag{3.2b}$$

The intersection graph is shown in Figure 3.2, where vertex j *represents* x_j *for* $1 \leq j \leq 6$.

The intersection graph depicts the complexity of relationships among variables. If the intersection graph is dense then, potentially, problem **GB** could prove difficult. However, as we will see in what follows, when the intersection graph presents low tree-width (hence, it

is sparse) there exists tractable ways to approach **GB**. The main theorem we prove in this chapter is the following:

Theorem 3.2.3 *Consider a **GB** problem whose intersection graph has tree-width $\leq \omega$.*

(a) There is an exact linear programming formulation with $O(2^{\omega}n)$ variables and constraints, with $\{0, 1, -1\}$-valued constraint coefficients.

(b) The formulation can be constructed by performing $O(2^{\omega}m)$ oracle queries and with additional workload $\tilde{O}(\omega n 2^{\omega}(m + \omega))$, where the "$\tilde{O}$" notation indicates logarithmic factors in m or n.

Note that the size of the formulation is independent of the number constraints in the given instance of **GB**. And even though we use the general setting of membership oracles, this theorem gives an exact reformulation.

A proof of part (a) in Theorem 3.2.3 can be obtained using techniques in [70] (Section 8) although not explicitly stated there. We will outline this proof, which relies on the "cone of set-functions" approach of [77] and also present a new proof.

Regarding part (b) of the theorem, it can be easily seen that $2^{\omega}m$ is a *lower bound* on the number of oracle queries that any algorithm for solving **GB** must perform.

Of course, Theorem 3.2.3 also implies the existence of an *algorithm* for solving **GB** in time polynomial in $(2^{\omega}, n, m)$. However one can also derive a direct algorithm of similar complexity using well-known, prior ideas on polynomial-time methods for combinatorial problems on graphs of bounded tree-width.

In the rest of this chapter we work on the context of Theorem 3.2.3. Before proving the Theorem, we begin by showing some examples for problem **GB** in order to emphasize its potential and further analyze different angles of it.

3.3 Examples of GB

Example 3.3.1 *(Linear binary integer programming). Let A be an $m \times n$ matrix, and consider a problem $\min\{c^T x : Ax \geq b,\ x \in \{0,1\}^n\}$. To view this problem as a special case*

of ***GB****, we set for* $1 \leq i \leq m$, $K_i = \{1 \leq j \leq n \,:\, a_{ij} \neq 0\}$ *and* $S^i = \{x \in \{0,1\}^{K_i} \,:\, \sum_{j \in K_i} a_{ij} x_j \geq b_i\}$.

In this special case, problem **GB** can be addressed by a variety of methods. Of particular interest in this work are the reformulation or lifting methods of [77] and [99]. Next we consider a more complex example, chosen to highlight the general nature of the problem.

Example 3.3.2 *Let* d, r, p *be positive integers. Consider a constrained semidefinite program over binary variables of the form*

$$\min \sum_{k=1}^{r} \sum_{i=1}^{d} \sum_{j=1}^{d} c_{kij} X^k_{i,j} \tag{3.3a}$$

$$\textit{subject to:} \quad M^k \bullet X^k = b_k, \quad 1 \leq k \leq r, \tag{3.3b}$$

$$X^k \in \mathbb{S}^{d \times d}_+, \quad 1 \leq k \leq r, \tag{3.3c}$$

$$\sum_{i,j} X^k_{i,j} \equiv 0 \mod p, \quad 1 \leq k \leq r, \tag{3.3d}$$

$$X^k_{i,1} = X^{k-1}_{i,d}, \quad 1 \leq i \leq d, \quad 2 \leq k \leq r, \tag{3.3e}$$

$$X^k_{i,j} \in \{0,1\}, \ \forall i, j, k. \tag{3.3f}$$

Here $M_1, \ldots, M_r$ *are symmetric* $d \times d$ *matrices, and* b *and* c *are vectors. Constraint* (3.3e) *states that the first column of matrix* X^k *is identical to the last column of matrix* X^{k-1}.

We obtain an instance of problem ***GB*** *with* $m = 2r - 1$, *as follows. First, for each* $1 \leq k \leq r$ *we let* K_k *be the set of triples* (i, j, k) *with* $1 \leq i, j \leq r$, *and* S^k *to be the set of binary values* $X^k_{i,j}$ *that satisfy* (3.3b)-(3.3d). *Next, for each* $2 \leq k \leq r$ *we let* K_{r+k-1} *be the set of all triples* $(i, 1, k-1)$ *and all triples* (i, d, k) *and* S^{r+k-1} *to be the set of binary values (indexed by* K_{r+k-1}*) such that* (3.3e) *holds.*

In the case of this last example, a direct application of standard integer programming methods appears difficult. Moreover, we stress that the sets S^i in problem **GB** are completely generic and that the membership oracle perspective can prove useful as we discuss below.

Theorem 3.2.3 concerns the tree-width of the intersection graph of a problem of type **GB**. Recall that as per Definition 3.2.1, given a problem instance $\mathcal{I}$ of **GB**, the intersection graph for $\mathcal{I}$ has a vertex for each $1 \leq j \leq n$, and an edge $\{j, k\}$ whenever there exists $1 \leq i \leq m$

such that $\{j,k\} \subseteq K_i$, that is to say, j and k appear in a common constraint in problem **GB**.

Example 3.3.3 *(Example 3.3.2, continued). Here the set of variables is given by*

$$\{(i,j,k) : 1 \le k \le r \text{ and } 1 \le i,j \le d\}.$$

The intersection graph of the problem will have

(a) the edge $\{(i,j,k),(i',j',k)\}$ *for all* $1 \le k \le r$ *and* $1 \le i,j,i',j' \le d$*, arising from constraints* (3.3b)-(3.3d)

(b) the edge $\{(i,1,k),(i,d,k-1)\}$ *for each* $1 \le k < r$ *and* $1 \le i \le d$*, arising from constraints* (3.3e)*.*

A tree-decomposition (T,Q) *of the intersection graph, of width* $O(d^2)$*, is obtained as follows. Define* T *as a path with vertices* $v_1,u_2,v_2,u_3,\ldots,v_{r-1},u_r,v_r$*. For* $1 \le k \le r$ *we set* $Q_{v_k} = \{(i,j,k) : 1 \le i,j \le d\}$ *and for* $2 \le k \le r$ *we set* $Q_{u_k} = \{(i,1,k),(i,d,k-1) : 1 \le i \le d\}$*. Sets* Q_{v_k} *account for all edges of type (a), whereas the sets* Q_{u_k} *cover all edges of type (b). Thus Theorem 3.2.3 states that there is an LP formulation for problem* (3.3) *with* $O(2^{d^2}d^2r)$ *variables and constraints.*

3.4 Reduction to the linear case

Consider a problem instance of **GB**. An apparently simpler alternative to the general approach we follow would be to construct, for $1 \le i \le m$, the polyhedron

$$P_i \doteq \operatorname{conv}\left\{x \in \{0,1\}^{K_i} : x \in S^i\right\} \subseteq \mathbb{R}^{K_i}.$$

Thus we can write P_i as the projection onto $\mathbb{R}^{K_i}$ of a polyhedron $\{x \in [0,1]^n : A^i x \ge b^i\}$ where each row of A^i has zero entries on any column not in K_i. Thus, the **GB** problem can be restated as the equivalent linear *integer* program

$$\min \quad c^T x \tag{3.4a}$$

$$\text{subject to:} \quad A^i x \ge b^i, \quad 1 \le i \le m \tag{3.4b}$$

$$x \in \{0,1\}^n. \tag{3.4c}$$

Switching to this formulation makes it possible to apply general integer programming methods to problem **GB**. However, this analysis ignores the size of formulation (3.4). In particular, for any integer $d \geq 1$ large enough there exist examples of 0/1-polytopes in $\mathbb{R}^d$ with at least

$$\left(\frac{d}{\log d}\right)^{d/4}$$

facets (up to constants). See [13], [49], [64]. Using this observation, one can construct examples of problem **GB** where the tree-width of the intersection graph is $\omega = d - 1$ and each of the matrices A^i has more than $\omega^{\omega/4}$ rows (see Example 3.4.1, below). This dependence on ω makes any classical integer programming method more computationally expensive than using the method we will present.

Example 3.4.1 *Choose $d \geq 2$ large enough so that there is a 0/1-polyhedron $P \subseteq \mathbb{R}^d$ with more than $(cd/\log d)^{d/4}$ facets for some c. Let P be given by the system $Ax \geq b$, where A is $M \times d$ ($M \geq (cd/\log d)^{d/4}$). Choose $N \geq 1$, and consider the system of inequalities over binary variables x^i_j, for $1 \leq i \leq N$ and $1 \leq j \leq d$:*

$$Ax^i \geq b, \quad 1 \leq i \leq N, \tag{3.5a}$$

$$x^1_j = x^i_j \quad 2 \leq i \leq N,\ 1 \leq j \leq \lfloor d/2 \rfloor. \tag{3.5b}$$

$$x^i_j \in \{0,1\} \quad \forall\ i,j. \tag{3.5c}$$

Constraint (3.5a) *indicates that this system includes N copies of polyhedron P, with each copy described using a different coordinate system. Constraint* (3.5b) *states that the first $\lfloor d/2 \rfloor$ coordinates take equal value across all such systems.*

Any linear program over (3.5) *is can be viewed as an example of problem **GB** with $m = 2N - 1$; for $1 \leq i \leq N$, K_i is used to represent the d variables x^i_j ($1 \leq j \leq d$) and S^i is a copy of the set of binary points contained in P (i.e. the extreme points of P).*

*The intersection graph of this instance of **GB** will be the union of N cliques (one for each set of variables x^i) plus the set of edges $\{x^1_1, x^i_1\}$ for $2 \leq i \leq N$. A tree-decomposition (T, Q) of this graph, of width $d - 1$, is as follows:*

- *T has vertices $u(0)$, as well as $u(i)$ and $v(i)$, for $1 \leq i \leq N$.*

- *Let* $Q_{u(0)} = \left\{x_j^1 \,:\, 1 \le j \le \lfloor d/2 \rfloor\right\}$, *for* $1 \le i \le N$

$$Q_{u(i)} = Q_{u(0)} \cup \left\{x_j^i \,:\, 1 \le j \le \lfloor d/2 \rfloor\right\}$$

and $Q_{v(i)} = \{x_j^i,\, 1 \le j \le d\}$.

Thus, $\omega = d-1$ *and Theorem 3.2.3 states that any linear objective problem over constraints* (3.5) *can be solved as a continuous LP with* $O(2^d dN)$ *variables and constraints. In contrast, system* (3.5) *has more than* $(cd/\log d)^{d/4}N$ *constraints.*

As the example shows, formulation (3.4) may be exponentially larger than the linear program stated in Theorem 3.2.3.

We will now proceed with the proof of Theorem 3.2.3, which we recall for convenience of the reader:

Theorem 3.2.3 *Consider a* ***GB*** *problem whose intersection graph has tree-width* $\le \omega$.

(a) There is an exact linear programming formulation with $O(2^\omega n)$ *variables and constraints, with* $\{0, 1, -1\}$*-valued constraint coefficients.*

(b) The formulation can be constructed by performing $O(2^\omega m)$ *oracle queries and with additional workload* $\tilde{O}(\omega n 2^\omega(m+\omega))$*, where the "*$\tilde{O}$*" notation indicates logarithmic factors in* m *or* n.

We will provide 2 formulations that prove part (a) of the Theorem. The first formulation is obtained using techniques in [70]. We outline this proof in Section 3.6, which relies on the "cone of set-functions" approach of [77]. The second and new formulation is presented in Section 3.7, and it is based on techniques presented in [21]. For part (b) please see Appendix B.1.

3.5 Preliminary definitions for LP reformulations

For the next sections we consider a problem of the form **GB** and we assume that we have a tree decomposition (T, Q) of width ω of the intersection graph of constraints (3.1b). Furthermore, we recall and define useful notation:

- As per the definition of **GB**, for $1 \leq i \leq m$ we let K_i be the set of variables x_j that explicitly appear in i-th constraint.
- Whenever we have values v_j for all j in some superset of K_i, we will write v_{K_i} to indicate the sub-vector of v with indices in K_i.

Definition 3.5.1 *Let $t \in V(T)$.*

(a) We say that $v \in \{0,1\}^{Q_t}$ is Q_t-feasible if $v_{K_i} \in S^i$ for every $1 \leq i \leq m$ such that $K_i \subseteq Q_t$.

(b) Write $\mathcal{F}_t = \{v \in \{0,1\}^{Q_t} : v \text{ is } Q_t\text{-feasible}\}$.

Roughly speaking, the set $\mathcal{F}_t$ collects all vectors that are *partially* or *locally* feasible. Note that such vectors do not posses full support, and, moreover, they might not even sub-vectors of *fully* feasible vectors. We will see, however, that this local information can be merged through an LP in order to form fully feasible solutions using the tree-decomposition of the intersection graph.

3.6 Lovász-Schrijver-based reformulation

In this first formulation, which proves Theorem 3.2.3 (a), we use:

- A variable λ_v^t, for each $t \in V(T)$ and each vector $v \in \mathcal{F}_t$.
- A variable Z_S, for each S such that $S \subseteq Q_t$ for some $t \in T$.

The formulation is as follows:

$$(\mathbf{LPz}) : \min \sum_{j=1}^{n} c_j Z_{\{j\}} \tag{3.6a}$$

s.t. $\forall t \in V(T)$:

$$Z_S = \sum \{\lambda_v^t : v \in \mathcal{F}_t,\ v_j = 1\ \forall j \in S\} \qquad \forall S \subseteq Q_t \tag{3.6b}$$

$$\sum_{v \in \mathcal{F}_t} \lambda_v^t = 1, \quad \lambda^t \geq 0. \tag{3.6c}$$

Constraints (3.6b) enforce consistency across different $t \in V(T)$ in the following sense. When a set S is a subset of at least two different sets Q_t, $Q_{t'}$ then we will have two constraints (3.6b) used to define the same value Z_S but with different right-hand sides; thus imposing a relationship between the λ^t and the $\lambda^{t'}$ variables. Additionally, when S is a singleton then Z_S appears in the objective of **LPz**. But in all remaining cases for variable Z_S (i.e., $|S| \neq 1$ and S contained in a single set Q_t) the constraint (3.6b) is redundant because Z_S does not appear anywhere else in the formulation.

Formulation **LPz** was built using the given tree-decomposition (T, Q) as an input. We know that this is a tree-decomposition for a graph on n vertices and as a result, it can be assumed that T has at most $O(n)$ vertices (see Proposition 3.1.4). Using this observation, since each set Q_t has cardinality at most $\omega + 1$, it is easily derived that formulation **LPz** has $O(2^\omega n)$ variables and constraints.

3.6.1 Correctness of formulation LPz

Constraint (3.6b) can be restated in a more familiar way. Given $t \in T$, (3.6b) states:

$$Z_S \;=\; \sum_{v \in F(t)} \lambda_v^t \zeta_S^{supp(v,Q_t)}, \qquad \forall S \in 2^{Q_t}. \tag{3.7}$$

Here, given a set Y and a vector $w \in \mathbb{R}^Y$, $supp(w, Y) = \{j \in Y \,:\, w_j \neq 0\}$, and for any set Y and $p \subseteq Y$ the vector $\zeta^p \in \{0,1\}^{2^Y}$ is defined by setting, for each $q \subseteq Y$,

$$\zeta_q^p \;=\; \begin{cases} 1 & \text{if } q \subseteq p \\ 0 & \text{otherwise.} \end{cases}$$

Constraints (3.6b)-(3.6c) then describe the Lovász-Schrijver approach to lifted formulations, restricted to a given set Q_t. It is clear that **LPz** amounts to a relaxation for the given problem **GB**, in the sense that given $\hat{x}$ feasible for **GB** then there is a vector $(\hat{Z}, \hat{\lambda})$ feasible for **LPz** where $\hat{Z}_{\{j\}} = \hat{x}_j$ for $1 \leq j \leq n$. To do so, let $t \in V(T)$ and denote by $\hat{x}^t$ the restriction of $\hat{x}$ to Q_t. Then by definition we have that $\hat{x}^t \in \mathcal{F}_t$. Thus we can set $\hat{\lambda}^t_{\hat{x}^t} = 1$ and $\hat{\lambda}^t_v = 0$ for any other $v \in \mathcal{F}_t$, and for any $S \subseteq Q_t$ $\hat{Z}_S = \zeta_S^{supp(\hat{x}^t, Q_t)}$. The last equation simply states that $\hat{Z}_S = 1$ iff $S \subseteq supp(\hat{x}, [n])$, a consistent definition across $t \in V(T)$. Hence indeed $(\hat{Z}, \hat{\lambda})$ is feasible for **LPz** and attains $\hat{Z}_{\{j\}} = \hat{x}_j$ for each $j = 1, \ldots, n$, as desired. Note that,

effectively, we have argued that the restriction of $\zeta^{supp(\hat{x},[n])}$ to $\mathbb{R}^{2^{Q_t}}$ for $t \in V(T)$ yields a feasible solution to **LPz**.

Next we argue that (3.6b)-(3.6c) defines an integral polyhedron. This is a consequence of the following result, which can be obtained from Lemma 8.18 of [70].

Lemma 3.6.1 *Suppose that (Z, λ) is a feasible solution to* (3.6b)-(3.6c)*. Then there exists a vector $W \in \mathbb{R}^{2^n}$, nonnegative values $\theta_1, \ldots, \theta_k$ and vectors $y^1, \ldots, y^k$ in $\mathbb{R}^n_+$ such that:*

(1) $Z_S = W_S$, for all $S \in \cup_{t \in V(T)} 2^{Q_t}$.

(2) $\sum_{i=1}^k \theta_i \ = \ 1$.

*(3) y^i is feasible for **GB**, for $1 \leq i \leq k$.*

(4) $W \ = \ \sum_{i=1}^k \theta_i \ \zeta^{supp(y^i,[n])}$.

As a consequence of (2)-(4), the vector W is a convex combination of the vectors $\zeta^{supp(y^i,[n])}$ which as argued above yield feasible solutions to **LPz**, thus proving the desired result. We remark that the proof of Lemma 8.18 of [70] is related to that of the tree-junction theorem; this technique, evocative of dynamic programming, was also used in [20] in a closely related setting.

3.7 Alternative reformulation

We now construct a formulation that yields Theorem 3.2.3 (a), as an alternative to the proof outlined in [70] and developed in Section 3.6.

As before, we assume a tree-decomposition (T, Q) with width ω of the intersection graph for formulation (3.1). We use the notation introduced in Definition 3.5.1, and additionally

Definition 3.7.1 *Let $t \in V(T)$. We let Ω_t denote the set of pairs (Y, N) with $Y \cap N = \emptyset$, $Y \cup N \subseteq Q_t$, and such that*

1. $|Y| \leq 1$ and $|N| = 0$, or

2. (Y, N) partition $Q_t \cap Q_{t'}$, for some $t' \in V(T)$ with $\{t, t'\} \in E(T)$.

The variables of this formulation are:

- A variable λ_v^t, for each $t \in V(T)$ and each vector $v \in \mathcal{F}_t$.
- A variable $X[Y, N]$, for each pair $(Y, N) \in 2^n \times 2^n$ with $(Y, N) \in \Omega_t$ for some $t \in V(T)$.

And the formulation is as follows.

$$(\textbf{LP-GB}) : \min \sum_{j=1}^{n} c_j X[\{j\}, \emptyset] \tag{3.8a}$$

s.t. $\forall t \in V(T)$:

$$X[Y, N] = \sum_{v \in \mathcal{F}_t} \lambda_v^t \prod_{j \in Y} v_j \prod_{j \in N} (1 - v_j) \quad \forall\ (Y, N) \in \Omega_t \tag{3.8b}$$

$$\sum_{v \in \mathcal{F}_t} \lambda_v^t = 1, \quad \lambda^t \geq 0. \tag{3.8c}$$

A similar counting argument to the one above can be used to prove that **LP-GB** has size $O(2^\omega n)$.

3.7.1 Correctness of formulation LP-GB

In this section we will show that **LP-GB** is a relaxation of **GB**, that the relaxation is exact and that the polyhedron defined by (3.8b)-(3.8c) is integral in the $X[Y, N]$ variables.

Remark 3.7.2

(f.1) *When (Y, N) partition $Q_t \cap Q_{t'}$ for some edge $\{t, t'\}$ then variable $X[Y, N]$ will appear in the constraint* (3.8b) *arising from t and also that corresponding to t'. This implies an equation involving λ^t and $\lambda^{t'}$.*

(f.2) *The sum on the right-hand side of constraint* (3.8b) *could be empty. This will be the case if for any $v \in \{0, 1\}^{Q_t}$ with $v_j = 1$ for all $j \in Y$ and $v_j = 0$ for all $j \in N$ there exists $1 \leq i \leq m$ with $K_i \subseteq Q_t$ and yet $v_{K_i} \notin S^i$. Then* (3.8b) *states $X[Y, N] = 0$.*

(f.3) *When $Y = N = \emptyset$ the right-hand side of* (3.8b) *is $\sum_{v \in \mathcal{F}_t} \lambda_v^t$. Hence we will have $X[\emptyset, \emptyset] = 1$.*

*(f.4) The λ variables are the same as those in formulation **LPz**. For any edge $\{t, t'\} \in E(T)$ and $Y \subseteq Q_t \cap Q_{t'}$ the terms on the right-hand side of the row* (3.8b) *are a subset of the terms on the right-hand side of the row* (3.6b) *corresponding to Y (additional statements are possible).*

First we show that **LP-GB** is a relaxation for **GB**, in a strong sense.

Lemma 3.7.3 *Let $\tilde{x}$ be a feasible solution to an instance for **GB**.*

*(i) There is a feasible, 0/1-valued solution $(\tilde{X}, \tilde{\lambda})$ to **LP-GB** such that for each variable $X[Y, N]$ in **LP-GB** we have $\tilde{X}[Y, N] = \prod_{j \in Y} \tilde{x}_j \prod_{j \in N}(1 - \tilde{x}_j)$.*

(ii) As a corollary $\sum_{j=1}^{n} c_j \tilde{X}[\{j\}, \emptyset] = c^T \tilde{x}$.

PROOF:

(i) For each variable $X[Y, N]$ in problem **LP-GB** we set $\tilde{X}[Y, N] = \prod_{j \in Y} \tilde{x}_j \prod_{j \in N}(1 - \tilde{x}_j)$. Further, for each $t \in V(T)$ let $\tilde{v}(t) \in \{0, 1\}^{Q_t}$ be the restriction of $\tilde{x}$ to Q_t, i.e. $\tilde{v}(t)_j = \tilde{x}_j$ for each $j \in Q_t$. Since $\tilde{x}$ is feasible, $\tilde{v}(t) \in \mathcal{F}_t$. Then we set $\tilde{\lambda}^t_{\tilde{v}(t)} = 1$ and $\tilde{\lambda}^t_v = 0$ for every vector $v \in \mathcal{F}_t$ with $v \neq \tilde{v}(t)$. By construction for every $t \in V(T)$ and $(Y, N) \in \Omega_t$ we have $\tilde{X}[Y, N] = 1$ iff $\tilde{v}(t)_j = 1$ for all $j \in Y$ and $\tilde{v}(t)_j = 0$ for all $j \in N$; in other words (3.8b) is satisfied.

(ii) This follows from (i).

■

As a consequence of Lemma 3.7.3, Theorem 3.2.3 will follow if we can prove that the constraint matrix in **LP-GB** defines an integral polyhedron in the $X[Y, N]$ variables. This will be done in Lemma 3.7.7 given below. In what follows, we will view T as rooted, i.e. all edges are directed so that T contains a directed path from an arbitrarily chosen **leaf** vertex r (the root of T) to every other vertex. If (v, u) is an edge thus directed, then we say that v is the parent of u and u is a child of v.

Definition 3.7.4 *A rooted subtree $\tilde{T}$ is a subtree of T, such that there exists a vertex u of $\tilde{T}$ so that $\tilde{T}$ contains a directed path from u to every other vertex of $\tilde{T}$. We then say that $\tilde{T}$ is rooted at u.*

Definition 3.7.5 *Let $\tilde{T}$ be a rooted subtree of T.*

(a) We denote by $\Omega(\tilde{T})$ the set $\bigcup_{t\in\tilde{T}}\Omega_t$.

(b) We denote by $\mathcal{V}(\tilde{T})$ the set $\{j \ : \ j \in Q_t$ for some $t \in \tilde{T}\}$.

Below we will prove the following result:

Theorem 3.7.6 *Let $(\hat{X}, \hat{\lambda})$ be a feasible solution to the **LP-GB** problem. Then for every rooted subtree $\tilde{T}$ there is a family of vectors*

$$p^{k,\tilde{T}} \in \{0,1\}^{\Omega(\tilde{T})},$$

vectors

$$x^{k,\tilde{T}} \in \{0,1\}^{\mathcal{V}(\tilde{T})}$$

and reals

$$0 < \mu^{k,\tilde{T}} \leq 1,$$

$(k = 1, 2, \ldots, n(\tilde{T}))$ satisfying the following properties:

*(a) For each $1 \leq k \leq n(\tilde{T})$ and each constraint $1 \leq i \leq m$ of problem **GB**, if $K_i \subseteq Q_t$ for some $t \in \tilde{T}$, then $x^{k,\tilde{T}} \in S^i$.*

(b) For $1 \leq k \leq n(\tilde{T})$ and each pair $(Y, N) \in \Omega(\tilde{T})$,

$$p^{k,\tilde{T}}[Y,N] \ = \ \prod_{j\in Y} x_j^{k,\tilde{T}} \ \prod_{j\in N}\left(1 - x_j^{k,\tilde{T}}\right).$$

As a result, for each $1 \leq k \leq n(\tilde{T})$ and $j \in \mathcal{V}(\tilde{T})$, $x_j^{k,\tilde{T}} = p^{k,\tilde{T}}[\{j\}, \emptyset]$.

(c) $\sum_{k=1}^{n(\tilde{T})} \mu^{k,\tilde{T}} \ = \ 1$.

(d) For each $(Y, N) \in \Omega(\tilde{T})$,

$$\hat{X}[Y,N] \ = \ \sum_{k=1}^{n(\tilde{T})} \mu^{k,\tilde{T}} p^{k,\tilde{T}}[Y,N].$$

The family of vectors $p^{k,\tilde{T}}$ and reals $\mu^{k,\tilde{T}}$ will be called a **decomposition of $(\hat{X}, \hat{\lambda})$ over $\tilde{T}$**.

Pending a proof of Theorem 3.7.6, we can show that the polyhedron defined by the constraints in **LP-GB** is integral.

Lemma 3.7.7 *The polyhedron defined by* (3.8b)-(3.8c) *is integral in the* $X[Y, N]$ *variables and problems* ***GB*** *and* ***LP-GB*** *have the same value.*

PROOF: Let $(\hat{X}, \hat{\lambda})$ be a feasible solution to **LP-GB**. We apply Theorem 3.7.6 with $\tilde{T} = T$ obtaining a family of vectors $p^k \in \{0,1\}^{\Omega(r)}$, vectors $x^k \in \{0,1\}^n$ and reals μ^k, for $1 \leq k \leq n(r)$, satisfying conditions (a)-(d) of the theorem. By (a) each vector x^k is feasible for **GB**, since each K_i induces a clique in the corresponding intersection graph and by Proposition 3.1.6 we can conclude. By (d), the vector $\hat{X}$ is a convex combination of the vectors p^k. From this we conclude. ■

This result completes the proof of Theorem 3.2.3, pending Theorem 3.7.6.

3.7.2 Proof of Theorem 3.7.6

Assume we have a feasible solution $(\hat{X}, \hat{\lambda})$ to **LP-GB**. The proof of Theorem 3.7.6 will be done by induction on the size of $\tilde{T}$. First we handle the base case.

Lemma 3.7.8 *If* $\tilde{T}$ *consists of a single vertex* u*, there is a decomposition of* $(\hat{X}, \hat{\lambda})$ *over* $\tilde{T}$*.*

PROOF: We have that $\Omega(\tilde{T}) = \Omega_u$ (see Definition 3.7.5). By (3.8c) we have $\sum_{v \in \mathcal{F}_u} \hat{\lambda}^u_v = 1$. Let $n(\tilde{T}) > 0$ be the number of elements $v \in \mathcal{F}_u$ with $\hat{\lambda}^u_v > 0$ and denote these vectors by $\{w(1), \ldots, w(n(\tilde{T}))\}$. Then, for $1 \leq k \leq n(\tilde{T})$ let $x^{k,\tilde{T}} = w(k)$ and $\mu^{k,\tilde{T}} = \hat{\lambda}^u_{w(k)}$. Finally, for $1 \leq k \leq n(\tilde{T})$ we define the vector $p^{k,\tilde{T}} \in \Omega_u$ by setting

$$p^{k,\tilde{T}}[Y, N] \;=\; \prod_{j \in Y} x^{k,\tilde{T}}_j \;\prod_{j \in N} \left(1 - x^{k,\tilde{T}}_j\right)$$

for each pair $(Y, N) \in \Omega_u$. Now we will verify that conditions (a)-(d) of Theorem 3.7.6 hold. Clearly (a)-(c) hold by construction. To see that (d) holds, note that $(\hat{X}, \hat{\lambda})$ satisfies (3.8b):

$$\begin{aligned}
\hat{X}[Y, N] &= \sum_{v \in \mathcal{F}_u} \hat{\lambda}^u_v \prod_{j \in Y} v_j \prod_{j \in N} (1 - v_j) \\
&= \sum_{k=1}^{n(\tilde{T})} \mu^{k,\tilde{T}} \prod_{j \in Y} x^{k,\tilde{T}}_j \prod_{j \in N} \left(1 - x^{k,\tilde{T}}_j\right) \\
&= \sum_{k=1}^{n(\tilde{T})} \mu^{k,\tilde{T}} p^{k,\tilde{T}}[Y, N]
\end{aligned}$$

which is condition (d), as desired. ■

Next we prove the general inductive step needed to establish Theorem 3.7.6. The technique used here is related to the junction tree theorem, is similar to one used in [20] and is reminiscent of Lemma L of [70].

Consider a vertex u of T and a subtree $\tilde{T}$ rooted at u with more than one vertex. Let v be a child of u. We will apply induction by partitioning $\tilde{T}$ into two subtrees: the subtree L consisting of v and all its descendants in $\tilde{T}$, and the subtree $H = \tilde{T} \setminus L$. Consider a decomposition of $(\hat{X}, \hat{\lambda})$ over L given by the vectors $p^{k,L} \in \{0,1\}^{\Omega(L)}$ and the positive reals $\mu^{k,L}$ for $k = 1, 2, \ldots, n(L)$, and a decomposition of $(\hat{X}, \hat{\lambda})$ over H given by the vectors $p^{k,H} \in \{0,1\}^{\Omega(H)}$ and the positive reals $\mu^{k,H}$ for $k = 1, 2, \ldots, n(H)$.

Denote by $\tilde{\mathcal{P}}$ the set of partitions of $Q_u \cap Q_v$ into two sets. Thus, by Definition 3.7.1, for each $(\alpha, \beta) \in \tilde{\mathcal{P}}$ we have a variable $X[\alpha, \beta]$. Note that $\Omega(\tilde{T}) = \Omega(H) \cup \Omega(L)$. We construct a family of vectors and reals satisfying (a)-(d) Theorem 3.7.6 for $\tilde{T}$, as follows.

For each $(\alpha, \beta) \in \tilde{\mathcal{P}}$ such that $\hat{X}[\alpha, \beta] > 0$, and each pair i, h such that $1 \leq i \leq n(L)$, $1 \leq h \leq n(H)$, and $p^{h,H}[\alpha, \beta] = p^{i,L}[\alpha, \beta] = 1$ we create a vector $q_{ih}^{\alpha,\beta}$ and a real $\gamma_{ih}^{\alpha,\beta}$ using the following rule. For any vertex t in $\tilde{T}$ and $(Y, N) \in \Omega_t$:

(r.1) If $t \in V(L)$ we set $q_{ih}^{\alpha,\beta}[Y, N] = p^{i,L}[Y, N]$.

(r.2) If $t \in V(H)$ we set $q_{ih}^{\alpha,\beta}[Y, N] = p^{h,H}[Y, N]$.

Further, we set

$$\gamma_{ih}^{\alpha,\beta} \;=\; \frac{\mu^{i,L}\,\mu^{h,H}}{\hat{X}[\alpha, \beta]}.$$

To argue that this construction is valid we note that since $\hat{X}[\alpha, \beta] > 0$, pairs of indices i, h as listed above must exist, by (d) of the inductive assumption applied to H and L. Furthermore, we have $\gamma_{ih}^{\alpha,\beta} > 0$.

Now we will prove that the q_{ih} and the γ_{ih} provide a decomposition of $(\hat{X}, \hat{\lambda})$ over $\tilde{T}$. Let i and h be given. Since the restriction of $p^{i,L}$ (and $p^{h,H}$) to L (resp., H) satisfy (a) and (b) of the inductive assumption, so will q_{ih}. Thus, there remains to prove (c) and (d).

First, consider (d). Let $(Y,N) \in \Omega(\tilde{T})$, say $(Y,N) \in \Omega(H)$. We claim that

$$\begin{aligned}&\sum_{\alpha,\beta,i,h} \gamma_{ih}^{\alpha,\beta} q_{ih}^{\alpha,\beta}[Y,N] \\ &= \sum_{(\alpha,\beta)\in\tilde{\mathcal{P}}\,:\,\hat{X}[\alpha,\beta]>0} \sum_{i=1}^{n(L)} \sum_{h=1}^{n(H)} \frac{\mu^{i,L}\,\mu^{h,H}}{\hat{X}[\alpha,\beta]} p^{i,L}[\alpha,\beta] p^{h,H}[\alpha,\beta] p^{h,H}[Y,N]. \end{aligned} \tag{3.9}$$

This equation holds because in any nonzero term in either expression we must have $p^{i,L}[\alpha,\beta] = p^{h,H}[\alpha,\beta] = 1$ and since $(Y,N) \in \Omega(H)$ we also have that $q_{ih}^{\alpha,\beta}[Y,N] = p^{h,H}[Y,N]$.
Now the right-hand side of (3.9) equals

$$\begin{aligned}&\sum_{(\alpha,\beta)\in\tilde{\mathcal{P}}\,:\,\hat{X}[\alpha,\beta]>0} \left[\left(\sum_{i=1}^{n(L)} \frac{\mu^{i,L} p^{i,L}[\alpha,\beta]}{\hat{X}[\alpha,\beta]} \right) \left(\sum_{h=1}^{n(H)} \mu^{h,H} p^{h,H}[\alpha,\beta] p^{h,H}[Y,N] \right) \right] \\ &= \sum_{(\alpha,\beta)\in\tilde{\mathcal{P}}\,:\,\hat{X}[\alpha,\beta]>0} \left(\sum_{h=1}^{n(H)} \mu^{h,H} p^{h,H}[\alpha,\beta] p^{h,H}[Y,N] \right), \end{aligned} \tag{3.10}$$

by the inductive assumption (d) applied to subtree L. The expression in (3.10) equals

$$\sum_{h=1}^{n(H)} \left(\left[\sum_{(\alpha,\beta)\in\tilde{\mathcal{P}}\,:\,\hat{X}[\alpha,\beta]>0} p^{h,H}[\alpha,\beta] \right] \mu^{h,H} p^{h,H}[Y,N] \right). \tag{3.11}$$

By inductive property (b) applied to subtree H, given $1 \le h \le n(H)$ we have that $p^{h,H}[\alpha,\beta] = 1$ for exactly one partition $(\alpha,\beta) \in \tilde{\mathcal{P}}$, and so expression (3.11) equals

$$\sum_{h=1}^{n(H)} \mu^{h,H} p^{h,H}[Y,N]. \tag{3.12}$$

In summary,

$$\sum_{\alpha,\beta,i,h} \gamma_{ih}^{\alpha,\beta} q_{ih}^{\alpha,\beta}[Y,N] \;=\; \sum_{h=1}^{n(H)} \mu^{h,H} p^{h,H}[Y,N].$$

and by induction applied to the subtree H this quantity equals $\hat{X}[Y,N]$. Thus property (d) does indeed hold.

Finally we turn to (c). Inductively, (c) and (d) hold for trees L and H. Thus, as noted in Remark 3.7.2 (f.3), $X[\emptyset,\emptyset] = 1$. But we have just shown that (d) holds for $\tilde{T}$, and in particular that it holds for $Y = N = \emptyset$. Using Remark 3.7.2 (f.3) we obtain that (c) holds for $\tilde{T}$, as desired. ■

Chapter 4

Mixed-Integer Polynomial Optimization with small tree-width

In Chapter 3 we discussed how to exploit tree-width-based sparsity in pure binary optimization problems. We showed how to obtain an LP reformulation of these problems whose size is parametrized by the *tree-width* of the problem.

In this chapter we will show how to use these results in order to obtain tractable approaches to *mixed-integer* polynomial optimization problems. Due to the inclusion of continuous variables, in this case we will obtain linear programming *approximations* instead of exact *reformulations*. We begin by stating the setting of this chapter, as well as the main theorem to be proved.

4.1 Problem description

Our main goal will be the study of optimization problems of the form

$$(\mathbf{PO}) \ : \ \min \ c^T x \tag{4.1a}$$

$$\text{subject to :} \quad f_i(x) \ \geq \ 0 \qquad 1 \leq i \leq m \tag{4.1b}$$

$$x \in \{0,1\}^p \times [0,1]^{n-p}, \tag{4.1c}$$

where each f_i is a polynomial. Any linear-objective polynomial optimization problem where explicit upper and lower bounds are known for all variables can be brought into this form by

appropriately translating and scaling variables, and restating any equation as two inequalities. We focus on obtaining linear programming approximations to **PO** that attain any desired tolerance (both feasibility and optimality). Our goal is to obtain linear programming approximation of polynomial size when problem **PO** is appropriately sparse. As in the previous chapter, we will make use of tree-width-based sparsity of the intersection graph of **PO**, defined in 3.2.1.

We represent the polynomial of the i-th constraint of **PO**, $f_i(x) \geq 0$, as

$$f_i(x) = \sum_{\alpha \in I(i)} f_{i,\alpha} x^\alpha, \tag{4.2}$$

where $I(i) \subseteq \mathbb{Z}^n_+$. Recall that, given $\alpha \in \mathbb{Z}^n_+$

$$x^\alpha \doteq \prod_{j=1}^{n} x_j^{\alpha_j}.$$

and $\|f_i\|_1 = \sum_{\alpha \in I(i)} |f_{i,\alpha}|$.

Definition 4.1.1 *Given a set of polynomial constraints $f_i(x) \geq 0$ $i = 1, \ldots, m$, and $\epsilon \geq 0$, we say a vector $\hat{x}$ is ϵ-scaled feasible if $f_i(\hat{x}) \geq -\epsilon \|f_i\|_1 \ \forall \ i = 1, \ldots, m$.*

The main result we will show in this chapter is as follows:

Theorem 4.1.2 *Consider an instance of problem **PO** and any $0 < \epsilon < 1$. Let ρ the maximum degree of any of the polynomials f_i. Given a tree-decomposition of the intersection graph of the constraints of width ω, there is a linear programming formulation with $O\left((2\rho/\epsilon)^{\omega+1} n \log(\rho/\epsilon)\right)$ variables and constraints, such that any optimal solution $\hat{x}$ for the LP satisfies:*

1. *$\hat{x}$ is ϵ-scaled feasible for* (4.1b),
2. *$c^T \hat{x} \leq P^* + \|c\|_1 \epsilon$, where P^* is the value of **PO**.*

The statement in Theorem 4.1.2 is indicative of the fact that as $\epsilon \to 0$ we converge to an optimal solution.

In what follows we will provide a proof for Theorem 4.1.2 together with some of our core constructions. Toward this goal, we begin by formally developing an approximation technique for **PO**.

4.2 Binary approximations of polynomial optimization problems

The general approximation scheme we will use for **PO** is based on a discretization method originally proposed in [51]; also see [22, 39, 54] and citations therein for more insight. Let r be a real with $0 \leq r \leq 1$. Then, given $0 < \gamma < 1$ we can approximate r as a sum of inverse powers of 2, within additive error γ, by making use of the following. Let

$$L_\gamma \doteq \lceil \log_2 \gamma^{-1} \rceil,$$

then there exist 0/1-values z_h, $1 \leq h \leq L_\gamma$, so that

$$\sum_{h=1}^{L_\gamma} 2^{-h} z_h \leq r \leq \sum_{h=1}^{L_\gamma} 2^{-h} z_h + 2^{-L_\gamma} \leq \sum_{h=1}^{L_\gamma} 2^{-h} z_h + \gamma \leq 1. \tag{4.3}$$

To apply this idea to problem **PO**, let $0 < \epsilon < 1$ and as before let ρ denote the maximum degree of any polynomial in **PO**. Choose γ so that

$$\epsilon = 1 - (1 - \gamma)^\rho, \tag{4.4}$$

then for each $j = p+1, \ldots, n$ (i.e, the indices of continuous variables) we will approximately represent x_j as $\sum_{h=1}^{L_\gamma} 2^{-h} z_{j,h}$ where each $z_{j,h}$ is a (new) binary variable.

For each $1 \leq i \leq m$ and $\alpha \in I(i)$ we write

$$Z(i, \alpha) = \{j : \alpha_j \neq 0, 1 \leq j \leq p\},$$

in other words, set $Z(i, \alpha)$ is the set given by the indices of the binary variables for **PO** that appear explicitly in monomial x^α. We consider the following replacement for problem **PO**:

$$(\mathbf{GB}(\gamma)): \quad \min \sum_{j=1}^{p} c_j x_j + \sum_{j=p+1}^{n} c_j \left(\sum_{h=1}^{L_\gamma} 2^{-h} z_{j,h} \right)$$

$$\text{s.t.} \quad \sum_{\alpha \in I(i)} f_{i,\alpha} \left[\prod_{j \in Z(i,\alpha)} x_j \prod_{j=p+1}^{n} \left(\sum_{h=1}^{L_\gamma} 2^{-h} z_{j,h} \right)^{\alpha_j} \right] \geq -\epsilon \|f_i\|_1, \ 1 \leq i \leq m \quad (4.5a)$$

$$x_j = \sum_{h=1}^{L_\gamma} 2^{-h} z_{j,h}, \ p+1 \leq j \leq n \quad (4.5b)$$

$$x_j \in \{0,1\}, \ 1 \leq j \leq p \quad (4.5c)$$

$$z_{j,h} \in \{0,1\}, \ p+1 \leq j \leq n, \ 1 \leq h \leq L_\gamma \quad (4.5d)$$

where $\epsilon = 1 - (1-\gamma)^\rho$. Note that the dependency of ϵ and γ can be swapped, as they are linked through a bijection.

Next we prove a series of results describing the quality of the approximation to problem **PO** provided by $\mathbf{GB}(\gamma)$. Toward this goal first we establish a technical property.

Lemma 4.2.1 *Suppose that for $j = 1, \ldots s$ we have values $u_j \geq 0$, $v_j \geq 0$, $q_j \in \mathbb{Z}_+$ with $u_j + v_j \leq 1$. Then*

$$\prod_{j=1}^{s} (u_j + v_j)^{q_j} - \prod_{j=1}^{s} u_j^{q_j} \leq 1 - \prod_{j=1}^{s} (1 - v_j)^{q_j}.$$

PROOF: Take any fixed index $1 \leq i \leq s$. The expression

$$\prod_{j=1}^{s} (u_j + v_j)^{q_j} - \prod_{j=1}^{s} (u_j)^{q_j}$$

is a non-decreasing function of u_i when all u_j and v_j are non-negative, and so in the range $0 \leq u_i \leq 1 - v_i$ it is maximized when $u_i = 1 - v_i$. ■

Next we establish our approximation result.

Lemma 4.2.2 *(a) Suppose $\bar{x}$ is feasible for **PO**. Then there is a feasible solution $(\tilde{x}, \tilde{z})$ for $\mathbf{GB}(\gamma)$ with objective value at most $c^T \bar{x} + \epsilon \|c\|_1$.*

(b) Suppose $(\hat{x}, \hat{z})$ is feasible for $\mathbf{GB}(\gamma)$. Then $\hat{x}$ is ϵ-scaled feasible for (4.1b) *and*

$$c^T \hat{x} = \sum_{j=1}^{p} c_j \hat{x}_j + \sum_{j=p+1}^{n} c_j \left(\sum_{h=1}^{L_\gamma} 2^{-h} \hat{z}_{j,h} \right).$$

PROOF:

(a) For each $j \in \{1, \ldots, p\}$ define $\tilde{x}_j = \bar{x}_j$. For $j \in \{p+1, \ldots, n\}$ choose binary values $\tilde{z}_{j,h} = \tilde{z}_{j,h}(\bar{x}_j)$ $h = 1, \ldots, L_\gamma$ so as to attain the approximation for $\bar{x}_j$ as in (4.3). We define $\tilde{x}_j$ for $j \in \{p+1, \ldots, n\}$ from $\tilde{z}$ according to (4.5b).

We claim that for each $1 \leq i \leq m$ and $\alpha \in I(i)$

$$\prod_{j=p+1}^{n} \left(\sum_{h=1}^{L_\gamma} 2^{-h} \tilde{z}_{j,h} \right)^{\alpha_j} \leq \prod_{j=p+1}^{n} \bar{x}_j^{\alpha_j} \leq \prod_{j=p+1}^{n} \left(\sum_{h=1}^{L_\gamma} 2^{-h} \tilde{z}_{j,h} \right)^{\alpha_j} + \epsilon. \tag{4.6}$$

The left-hand inequality follows from (4.3). To obtain the right-hand inequality we apply Lemma 4.2.1 with $s = n$ and for $p+1 \leq j \leq n$, $q_j = \alpha_j$, $u_j = \sum_{h=1}^{L_\gamma} 2^{-h} \tilde{z}_{j,h}$, and $v_j = \bar{x}_j - u_j$. Using (4.3) and the definition (4.4) of ϵ, Lemma 4.2.1 yields

$$\begin{aligned} \prod_{j=p+1}^{n} \bar{x}_j^{\alpha_j} - \prod_{j=p+1}^{n} \left(\sum_{h=1}^{L_\gamma} 2^{-h} \tilde{z}_{j,h} \right)^{\alpha_j} &\leq 1 - \prod_{j=p+1}^{n} (1 - v_j)^{\alpha_j} \\ &\leq 1 - \prod_{j=p+1}^{n} (1 - \gamma)^{\alpha_j} \\ &\leq 1 - (1 - \gamma)^{\rho} \\ &= \epsilon \end{aligned}$$

as desired. Therefore, $(\tilde{x}, \tilde{z})$ is feasible for **GB(γ)** and the second assertion in (a) is similarly proved.

(b) By definition of $\hat{x}$, for each $1 \leq i \leq m$ we have

$$\begin{aligned} f_i(\hat{x}) &= \sum_{\alpha \in I(i)} f_{i,\alpha} \prod_{j=1}^{n} \hat{x}_j^{\alpha_j} \\ &= \sum_{\alpha \in I(i)} f_{i,\alpha} \left[\prod_{j \in Z(i,\alpha)} \hat{x}_j \prod_{j=p+1}^{n} \left(\sum_{h=1}^{L_\gamma} 2^{-h} \hat{z}_{j,h} \right)^{\alpha_j} \right] \\ &\geq -\epsilon \, \|f_i\|_1 \end{aligned}$$

as $(\hat{x}, \hat{z})$ is feasible for **GB(γ)**. The second assertion holds by definition of $\hat{x}$.

■

Corollary 4.2.3 *Let P^* be the optimal value of problem **PO**, and let $(\hat{x}(\gamma), \hat{z}(\gamma))$ be optimal for **GB(γ)**. Then $c^T\hat{x}(\gamma) \leq P^* + \epsilon\|c\|_1$ and $\hat{x}(\gamma)$ is ϵ-scaled feasible for* (4.1b).

Remark 4.2.4 *As per the corollary, $\hat{x}$ achieves feasibility and optimality tolerance proportional to ϵ for problem **PO**. However, $\hat{x}$ may actually be super-optimal for **PO**. Nevertheless for any sequence $\gamma_k \to 0^+$, the vectors $\hat{x}(\gamma_k)$ will have an accumulation point x^*, and this point necessarily must be feasible and thus optimal for **PO**.*

The reader may also notice in the proof of Lemma 4.2.2 (b) that the error bound $-\epsilon\|f_i\|_1$ is conservative, because an approximation error is only incurred on terms of $f_i(x)$ that are non-constant and that involve at least one continuous variable. Accounting for this fact will provide a tighter approximation estimate, which we skip to simplify notation.

Additionally, note that in the proof of Lemma 4.2.1, the only property that each q_j is actually required to satisfy is that either $q_j = 0$ or $q_j \geq 1$. One can use this observation to generalize Theorem 4.1.2 beyond polynomials. However this produces additional complications such as the approximation of rational powers. We will take up these and related issues in future work.

Corollary 4.2.3 implies that given any $\epsilon > 0$, problem **GB(γ)** with

$$\gamma = 1 - (1 - \epsilon)^{1/\rho}$$

provides an "ϵ-scaled" approximation to **PO**. Replacing the equality constraints (4.5b), problem **GB(γ)** becomes a pure binary problem, with m constraints and at most

$$nL_\gamma = O(n \log_2(\rho/\epsilon)) \quad \text{variables.}$$

Henceforth we refer to **GB(γ)** as a *pure binary* problem. However, for the sake of notation, in the solution vectors of **GB(γ)** we will include the continuous components when referring to the x variables.

In the next section we will reformulate this problem as a linear program, thus yielding an LP approximation to **PO** and completing the proof of Theorem 4.1.2.

4.3 Linear reformulation of the binary approximation

4.3.1 Sparsity of the approximation

We already established that the pure binary problems **GB(γ)** provides an approximation to **PO** within guaranteed tolerance. This result holds for all problems **PO** regardless of sparsity. In this section we will show how to use structural sparsity to reformulate the pure binary problems as equivalent LPs of moderate size.

To this effect, suppose that (T, Q) is a tree-decomposition for the intersection graph of an instance of problem **PO** and choose γ as in (4.4) in order to obtain the desired tolerance ϵ. We will now construct a tree-decomposition for the intersection graph for the corresponding instance of problem **GB(γ)**. This tree-decomposition will be of the form (T, Q') (note: same tree T) where for any vertex t of T we set

$$Q'_t \;=\; \{\, x_j \,:\, x_j \in Q_t,\ j = 1, \ldots, p\} \cup \{\, z_{j,h} \,:\, x_j \in Q_t,\ j = p+1, \ldots, n \text{ and } 1 \le h \le L_\gamma \,\}.$$

Lemma 4.3.1

*(a) (T, Q') is a tree-decomposition of the intersection graph for problem **GB(γ)**.*

(b) Further, if (T, Q) has width w then (T, Q') has width at most $L_\gamma(w+1) - 1$.

Proof:

(a) We need to establish properties (i)-(iii) of Definition (3.1.1); (iii) is clear. Note that $z_{j,h} \in Q'_t$ if and only if $x_j \in Q_t$. This proves (i). To see that (ii) holds, note that any edge of the intersection graph for **GB(γ)** arises from some constraint (4.1b).

(b) The width statement is true by construction of the sets Q'_t and the fact that every $|Q_t| \le w + 1$.

■

Remark 4.3.2 *As a consequence of this result, not only does problem **GB(γ)** provide a close approximation to problem **PO**, but when **PO** is structurally sparse (small tree-width of the intersection graph) then so is **GB(γ)**, modulo the $O(L_\gamma)$ multiplicative increase in tree-width.*

Recall that Theorem 3.2.3 yields an LP reformulation of pure binary problems whose size is parametrized by the tree-width of the intersection graph. Therefore, an application of Theorem 3.2.3 along with Lemma 4.3.1 to **GB(γ)** will provide the desired result. We put together these parts next.

4.3.2 From sparse PO to small LP approximations

Here we use the above results to prove Theorem 4.1.2. Consider an instance of **PO** on n variables. As before, let ρ represent the maximum degree of any of the polynomials f_i. Suppose we have a tree-decomposition of width ω of the intersection graph of the constraints. Given $0 < \epsilon < 1$ we proceed as follows:

1. We choose
$$\gamma = 1 - (1-\epsilon)^{1/\rho},$$
so that $\epsilon = 1 - (1-\gamma)^{\rho}$ as per (4.4). Note that without loss of generality ϵ is small enough so that $\gamma > \frac{\epsilon}{2\rho}$.

2. We apply Theorem 3.2.3 to construct the linear programming reformulation of the all-binary problem **GB(γ)** (formulation (4.5)). Let us call this linear program LP(γ).

3. As per Corollary 4.2.3, **GB(γ)** and thus, LP(γ), yields a vector $\hat{x}(\gamma)$ that is ϵ-scaled feasible for (4.1b) and $c^T\hat{x}(\gamma) \leq P^* + \epsilon\|c\|_1$.

Next we analyze the size of LP(γ).

1. By Lemma 4.3.1, there is a tree-decomposition of the intersection graph for **GB(γ)** of width at most $L_\gamma(\omega+1)-1$, where
$$L_\gamma \;=\; \lceil \log_2 \gamma^{-1} \rceil < \log_2(2\rho/\epsilon) + 1$$
for ϵ small enough.

2. Further, **GB(γ)** has at most $nL_\gamma = O(n\log_2(\rho/\epsilon))$ variables.

3. Thus, since the tree-width of the intersection graph of **GB(γ)** is at most $L_\gamma(\omega+1)-1$, by Theorem 3.2.3, the number of variables and constraints in LP(γ) is
$$O((2\rho/\epsilon)^{\omega+1} n \log_2(\rho/\epsilon))$$

The proof of Theorem 4.1.2 is now complete.

4.4 Final comments

To conclude this chapter, we discuss additional aspects of the approximations we just introduced. We will discuss if an improvement on the dependency of ϵ in Theorem 4.1.2 is possible, and provide a complete example on the resulting LP approximation we obtain applying the technique outlined in Section 4.3.2.

4.4.1 Can the dependence on ϵ be improved upon?

The approximation scheme given by Theorem 4.1.2 has two characteristics: first, it allows a violation of each constraint by ϵ times the 1-norm of the constraint, and second, the running time is pseudo-polynomial in ϵ^{-1}. One may wonder if either characteristic can be improved. For example, one might ask for constraint violations that are at most ϵ, independent of the 1-norm of the constraints. However this is not possible even for a *fixed* value of ϵ, unless P=NP. For completeness, we include a detailed analysis of this fact in Section C.1 of the Appendix. Intuitively, if we were allowed to approximately satisfy every constraint with an error that does not depend on the data, we could appropriately scale constraint coefficients so as to obtain exact solutions to NP-hard problems.

Similarly, it is not possible to reduce the pseudo-polynomial dependency on ϵ^{-1} in general. The precise statement is given in Section C.2 of the Appendix, and the intuitive reasoning is similar: if there was a formulation of size polynomially dependent on $\log(\epsilon^{-1})$ (and not on ϵ^{-1}) we could again solve NP-hard problems in polynomial time.

4.4.2 Example of LP approximation to PO

We now give a full detailed example with the explicit formulation we will obtain when approximating a problem **PO** with the pure binary problem **GB(γ)**, and then reformulating the latter as an LP. Consider the following low-dimensional example for problem **PO**:

$$\min\{\, x_1 + x_2 + x_3 \,:\, x_1^2 + x_2^2 \geq 1.95,\ x_3^2 + x_2^2 \geq 1.95,\ x \in [0,1]^3 \,\}.$$

Suppose we build our binary powers approximation using $\gamma = 1/8 = 0.125$ (and as per (4.4) $\epsilon = 0.234375$), thus $L_\gamma = 3$. We obtain the pure binary approximation with $N = 9$ variables:

$$\text{GB}(\tfrac{1}{8}): \quad \min \sum_{h=1}^{3} 2^{-h} z_{1,h} + \sum_{h=1}^{3} 2^{-h} z_{2,h} + \sum_{h=1}^{3} 2^{-h} z_{3,h} \tag{4.7a}$$

$$\text{s.t.} \quad \left(\sum_{h=1}^{3} 2^{-h} z_{1,h}\right)^2 + \left(\sum_{h=1}^{3} 2^{-h} z_{2,h}\right)^2 \geq 1.95 - 3.95\epsilon \approx 1.024 \tag{4.7b}$$

$$\left(\sum_{h=1}^{3} 2^{-h} z_{3,h}\right)^2 + \left(\sum_{h=1}^{3} 2^{-h} z_{2,h}\right)^2 \geq 1.95 - 3.95\epsilon \tag{4.7c}$$

$$z_{j,h} \in \{0,1\}, \quad 1 \leq j \leq 3, \quad 1 \leq h \leq 3. \tag{4.7d}$$

A tree-decomposition of the intersection graph is (T, Q) where

- T is the tree with vertices α and β and the single edge $\{\alpha, \beta\}$, and
- $Q_\alpha = \{z_{1,1}, z_{1,2}, z_{1,3}, z_{2,1}, z_{2,2}, z_{2,3}\}$
- $Q_\beta = \{z_{3,1}, z_{3,2}, z_{3,3}, z_{2,1}, z_{2,2}, z_{2,3}\}$

Next, in order to construct an LP formulation of (4.7) using Theorem 3.2.3, we construct the set $\mathcal{F}_\alpha$, i.e. the set of assignments of binary values to the members of Q_α such that constraint (4.7b) holds. We will argue that there are six such assignments, which we indicate as binary vectors using the same ordering as in the definition of Q_α:

$$\begin{array}{lll} \text{(a)} \quad (1,0,1,1,1,1), & \text{(b)} \quad (1,1,0,1,1,0), & \text{(c)} \quad (1,1,0,1,1,1) \\ \text{(d)} \quad (1,1,1,1,0,1), & \text{(e)} \quad (1,1,1,1,1,0), & \text{(f)} \quad (1,1,1,1,1,1). \end{array}$$

To show that (a)-(d) are the only feasible vectors, we first note that if either $z_{1,1} = 0$ or $z_{2,1} = 0$ then (4.7b) is violated. Also, if $z_{1,u} = 0$ and $z_{2,v} = 0$ for some u, v then $u = v = 3$ and all other $z_{1,w}$ and $z_{1,w}$ must equal 1. Likewise there are six similar assignments corresponding to the members of Q_β; we denote these (g), (h), (i), (j), (k), (l). We assume these are constructed in the same way as (a)-(f).

Now we turn to the construction of **LPz** (formulation (3.6)). As discussed there, we need to add constraints (3.6b) in just two kinds of cases: singletons S, and S which are common

subsets of Q_α and Q_β. In this latter case we have all $S \subseteq \{z_{2,1}, z_{2,2}, z_{2,3}\}$. We can now write the LP (3.6) for our example. In this LP we will abbreviate a variable $z_{j,h}$ as the pair (j,h).

$$\min \sum_{j=1}^{3}\sum_{h=1}^{3} 2^{-h} Z_{\{(j,h)\}} \tag{4.8a}$$

$$\text{s.t. } Z_{\{(1,1)\}} = \lambda_a^\alpha + \lambda_b^\alpha + \lambda_c^\alpha + \lambda_d^\alpha + \lambda_e^\alpha + \lambda_f^\alpha \tag{4.8b}$$

$$Z_{\{(1,2)\}} = \lambda_b^\alpha + \lambda_c^\alpha + \lambda_d^\alpha + \lambda_e^\alpha + \lambda_f^\alpha \tag{4.8c}$$

$$Z_{\{(1,3)\}} = \lambda_a^\alpha + \lambda_d^\alpha + \lambda_e^\alpha + \lambda_f^\alpha \tag{4.8d}$$

$$Z_{\{(2,1)\}} = \lambda_a^\alpha + \lambda_b^\alpha + \lambda_c^\alpha + \lambda_d^\alpha + \lambda_e^\alpha + \lambda_f^\alpha \tag{4.8e}$$

$$Z_{\{(2,2)\}} = \lambda_a^\alpha + \lambda_b^\alpha + \lambda_c^\alpha + \lambda_e^\alpha + \lambda_f^\alpha \tag{4.8f}$$

$$Z_{\{(2,3)\}} = \lambda_a^\alpha + \lambda_c^\alpha + \lambda_d^\alpha + \lambda_f^\alpha \tag{4.8g}$$

$$Z_{\{(2,1)\}} = \lambda_g^\beta + \lambda_h^\beta + \lambda_i^\beta + \lambda_j^\beta + \lambda_k^\beta + \lambda_l^\beta \tag{4.8h}$$

$$Z_{\{(2,2)\}} = \lambda_g^\beta + \lambda_h^\beta + \lambda_i^\beta + \lambda_k^\beta + \lambda_l^\beta \tag{4.8i}$$

$$Z_{\{(2,3)\}} = \lambda_g^\beta + \lambda_i^\beta + \lambda_k^\beta + \lambda_l^\beta \tag{4.8j}$$

$$Z_{\{(3,1)\}} = \lambda_g^\beta + \lambda_h^\beta + \lambda_i^\beta + \lambda_j^\beta + \lambda_k^\beta + \lambda_l^\beta \tag{4.8k}$$

$$Z_{\{(3,2)\}} = \lambda_h^\beta + \lambda_i^\beta + \lambda_j^\beta + \lambda_k^\beta + \lambda_l^\beta \tag{4.8l}$$

$$Z_{\{(3,3)\}} = \lambda_g^\beta + \lambda_j^\beta + \lambda_k^\beta + \lambda_l^\beta \tag{4.8m}$$

$$Z_{\{(2,1),(2,2)\}} = \lambda_a^\alpha + \lambda_b^\alpha + \lambda_c^\alpha + \lambda_e^\alpha + \lambda_f^\alpha \tag{4.8n}$$

$$Z_{\{(2,1),(2,2)\}} = \lambda_g^\beta + \lambda_h^\beta + \lambda_i^\beta + \lambda_k^\beta + \lambda_l^\beta \tag{4.8o}$$

$$Z_{\{(2,1),(2,3)\}} = \lambda_a^\alpha + \lambda_c^\alpha + \lambda_d^\alpha + \lambda_f^\alpha \tag{4.8p}$$

$$Z_{\{(2,1),(2,3)\}} = \lambda_g^\beta + \lambda_i^\beta + \lambda_j^\beta + \lambda_l^\beta \tag{4.8q}$$

$$Z_{\{(2,2),(2,3)\}} = \lambda_a^\alpha + \lambda_c^\alpha + \lambda_f^\alpha \tag{4.8r}$$

$$Z_{\{(2,2),(2,3)\}} = \lambda_g^\beta + \lambda_i^\beta + \lambda_l^\beta \tag{4.8s}$$

$$Z_{\{(2,1),(2,2),(2,3)\}} = \lambda_a^\alpha + \lambda_c^\alpha + \lambda_f^\alpha \tag{4.8t}$$

$$Z_{\{(2,1),(2,2),(2,3)\}} = \lambda_g^\beta + \lambda_i^\beta + \lambda_l^\beta \tag{4.8u}$$

$$\lambda_a^\alpha + \lambda_b^\alpha + \lambda_c^\alpha + \lambda_d^\alpha + \lambda_e^\alpha + \lambda_f^\alpha = \lambda_g^\beta + \lambda_h^\beta + \lambda_i^\beta + \lambda_j^\beta + \lambda_k^\beta + \lambda_l^\beta = 1 \tag{4.8v}$$

$$\lambda \geq 0.$$

The above LP (4.8) produces the solution $x_1 = x_3 = .625$ and $x_2 = .875$ with value 2.15.

The optimum solution for the original problem is on the other hand attained by $x_1 = x_3 \approx 0.9747$, $x_2 = 1.0$, with objective value ≈ 2.9494.

Remark 4.4.1 *The goal of this example is to illustrate our techniques with an instance that yields a small LP formulation. Should we use* $\gamma = 1/16$ *(*$L_\gamma = 4$*) for example, we would obtain* $\epsilon \approx 0.12$ *which would reduce our approximation error by a factor of approximately two at the cost of a larger LP.*

Chapter 5

Network Polynomial Optimization

In this chapter we focus on polynomial optimization problems **PO** where there is network structure as an explicit element of the problem description. In these problems we are given a network[1], as in the AC-OPF problem, and both variables and constraints are associated with the network structure directly. We will study how low tree-width in this underlying network affects tractability, and show how different this is from assuming low tree-width in the intersection graph, as in the previous chapters.

5.1 Problem description

We consider problems we term Network Polynomial Optimization problems (**NPO**); these are **PO** problems defined using the structure of a network. As was mentioned in the Introduction, in an **NPO** over a network $\mathcal{G}$, for each node $u \in V(\mathcal{G})$ there is a set $\mathcal{X}_u$ of variables associated with u and a set $\mathcal{K}_u$ of constraints associated with u. The optimization problem is of the following form:

$$(\textbf{NPO}): \qquad \min \ c^T x \tag{5.1a}$$

$$\text{subject to:} \qquad \sum_{\{u,v\}\in\delta_{\mathcal{G}}(u)} p^{(k)}_{(u,v)}(\mathcal{X}_u \cup \mathcal{X}_v) \ \geq \ 0, \quad k \in \mathcal{K}_u, \quad u \in V(\mathcal{G}) \tag{5.1b}$$

$$x \in \{0,1\}^p \times [0,1]^{n-p}, \tag{5.1c}$$

[1]We use the term 'network' to contrast with 'graph' which we reserve for intersection graphs. Likewise we will typically use 'node' to refer to a vertex of a network.

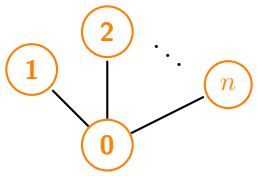


Figure 5.1: Star network witn $n+1$ nodes.

where $\delta_{\mathcal{G}}(u)$ is the set of edges of $\mathcal{G}$ incident with u and each $p^{(k)}_{(u,v)}(\mathcal{X}_u \cup \mathcal{X}_v)$ is a polynomial associated with the ordered pair (u,v). The notation $p^{(k)}_{(u,v)}(\mathcal{X}_u \cup \mathcal{X}_v)$ stresses the *ordered* pair (u,v) and indicates that the polynomial only involves variables in[2] $\mathcal{X}_u \cup \mathcal{X}_v$. The $p^{(k)}_{(u,v)}$ will be termed the *arc polynomials* of the problem. We do not assume that the sets $\mathcal{X}_u$ are pairwise disjoint. Rather (for a technical reason) we allow intersections but we assume that for any variable x_j, the set of $u \in V(\mathcal{G})$ with $x_j \in \mathcal{X}_u$ induces a connected subgraph of $\mathcal{G}$. For future reference, we will refer to this assumption as the *connectedness requirement*.

The definition of **NPO** can be seen as a direct generalization of the AC-OPF problem, where the sets $\mathcal{X}_u$ correspond to the voltage variables in each node u, and the arc polynomials are defined using the power flow equations.

We are interested in achieving tractability of **NPO** when $\mathcal{G}$ is appropriately sparse, i.e, exploting its tree-width. In devising an algorithm for problem **NPO** parametrized by the tree-width of $\mathcal{G}$, a direct reliance on Theorem 4.1.2 runs into a difficulty as detailed next.

Example 5.1.1 *Consider an **NPO** where*

(i) The network $\mathcal{G}$ is a star on $n+1$ nodes $0,1,\ldots,n$ and edge set $\{0,j\}$ for $1 \le j \le n$. See Figure 5.1.

(ii) $\mathcal{X}_0 = \emptyset$ and $\mathcal{X}_j = \{x_j\}$ for $1 \le j \le n$.

(iii) There is a single constraint (5.1b) *and it is associated with node* 0*. In this constraint we have $p^{(1)}_{(0,j)}(\mathcal{X}_0 \cup \mathcal{X}_j) = jx_j^3 - 1/n$ for $1 \le j \le n$.*

[2]Using this notation we allow cases where only variables in $\mathcal{X}_u$ or in $\mathcal{X}_v$ are actually involved. Without loss of generality we assume that for a given k, no two $p^{(k)}_{(u,v)}$ and $p^{(k)}_{(u,v')}$ use a common monomial.

Thus the (single) constraint (5.1b) *for this example is:* $\sum_{j=1}^{n} jx_j^3 \geq 1$. *This is a dense constraint, even though the input network* $\mathcal{G}$ *is a tree. In fact, the intersection graph of this constraint is a clique on* n *vertices (with tree-width* $n-1$*), and, as a consequence, if we directly apply Theorem 4.1.2 we will obtain a formulation of size exponential in* n.

Another example of this issue can be given using the well-known kanpsack problem:

Example 5.1.2 *Consider a knapsack problem* $\min\{\, c^T x \;:\; a^T x \geq b,\; x \in \{0,1\}^n \,\}$. *This can be cast as* ***NPO****, using a star network on* $n+1$ *nodes as in Example 5.1.1, which has tree-width* 1. *Yet, if* $a_j \neq 0$ *for all* j*, the intersection graph is a clique of size* n.

In summary: even if an **NPO** problem arises from a network with small tree-width, the problem, if viewed directly as an instance of **PO**, may yield a very dense formulation. However, we will argue that one can always *reformulate* any instance of **NPO** over a small tree-width network $\mathcal{G}$ so as to obtain an equivalent instance which, when viewed as a general problem **PO** gives rise to an intersection graph with (still) small tree-width.

For simplicity, in some cases we abbreviate constraint (5.1b) as $p_u^{(k)}(x) \geq 0$. The main result we will prove in this chapter is as follows.

Theorem 5.1.3 *Consider an instance of* ***NPO*** *and any* $0 < \epsilon < 1$. *Let* D, Δ *and* ρ *be such that:*

- *The network* $\mathcal{G}$ *has at most* D *edges incident with any node,*
- *The number of variables plus the number of constraints associated with any node of* $\mathcal{G}$ *is at most* Δ*,*
- *Every polynomial* $p_u^{(k)}$ *has maximum degree* $\leq \rho$

Then there is a linear program of size $O((D\rho/\epsilon)^{O(\Delta\omega)}\, n \log(\rho/\epsilon))$*, such that any optimal solution* $\hat{x}$ *for the LP satisfies:*

1. $\hat{x}$ *is* ϵ*-scaled feasible for* (5.1b)*,*
2. $c^T\hat{x} \leq P^* + \|c\|_1\epsilon$*, where* P^* *is the value of* ***NPO****.*

In Example 5.1.1 $\omega = 1$, $D = n$, $\rho = 3$ and $\Delta = 1$. Thus Theorem 5.1.3 yields an LP formulation of size $O((n/\epsilon)^{O(1)} \log(1/\epsilon))$.

5.2 Impact of node-degrees in an NPO

Let us continue analyzing Example 5.1.1, which illustrates how it is not possible to directly obtain Theorem 5.1.3 as a consequence of Theorem 4.1.2. In this example the network $\mathcal{G}$ was a "star" with node set $0, 1, \ldots, n$ and center node 0. The **NPO** had a single constraint (5.1b), associated with node 0, with arc polynomials $p^{(1)}_{(0,j)}(\mathcal{X}_0 \cup \mathcal{X}_j) = jx_j^3 - \frac{1}{n}$ for $1 \leq j \leq n$. The corresponding constraint (5.1b) reads: $\sum_{j=1}^{n} jx_j^3 \geq 1$. This is a dense constraint and a direct application of Theorem 4.1.2 will produce a formulation of size exponential in n.

This example illustrates the point that if, in an **NPO**, a node has high degree, the intersection graph of the **NPO** will likely have high tree-width. This observation suggests that we should try to reformulate an **NPO** into an equivalent **NPO** on a network where every node has small degree. The following construction is a counterpoint to Example 5.1.1 and is a central component in our approach to proving Theorem 5.1.3.

Consider an **NPO**, $\bar{\mathcal{P}}$, on a network $\bar{\mathcal{G}}$ all of whose nodes have degree at most three, and with associated sets of variables $\bar{\mathcal{X}}_u$. Let $(\bar{T}, \bar{\mathcal{Q}})$ be a tree decomposition of $\bar{\mathcal{G}}$. Form the pair $(\bar{T}, \bar{\mathcal{Q}}')$, where for each $t \in V(\bar{T})$ we define

$$\bar{\mathcal{Q}}'_t = \bigcup \{\bar{\mathcal{X}}_v \,:\, v \in \bar{\mathcal{Q}}_t \text{ or } \{u, v\} \in E(\bar{\mathcal{G}}) \text{ for some } u \in \bar{\mathcal{Q}}_t\}. \tag{5.2}$$

We have that:

Theorem 5.2.1 *$(\bar{T}, \bar{\mathcal{Q}}')$ is a tree-decomposition for the intersection graph of **NPO** $\bar{\mathcal{P}}$. If the width of $(\bar{T}, \bar{\mathcal{Q}})$ is $\bar{w}$, then the width of $(\bar{T}, \bar{\mathcal{Q}}')$ is at most $3(\bar{w} + 1)\bar{k} - 1$, where $\bar{k} = \max_{u \in V(\bar{\mathcal{G}})} |\bar{\mathcal{X}}_u|$.*

We will prove this result shortly. The result suggests a way to obtain Theorem 5.1.3. Namely, given an **NPO**, $\mathcal{P}$, on a general network of "small" tree-width we reformulate it as an equivalent **NPO**, $\bar{\mathcal{P}}$, on a network $\bar{\mathcal{G}}$ with nodes of degree at most three *and* such that $\bar{\mathcal{G}}$ also has "small" tree-width. If, in addition the parameter $\bar{k}$ in the statement of Theorem 5.2.1 is also "small", then our reformulation will be an **NPO** which, as a general polynomial

optimization problem, has an intersection graph with small tree-width and thus can be handled using Theorem 4.1.2. Note that, additionally, Theorem 4.1.2 only yields approximately feasible solutions, and so part of our task will be to guarantee that approximate feasibility for $\bar{\mathcal{P}}$ carries over to $\mathcal{P}$.

PROOF: (Theorem 5.2.1) The width statement follows directly from definition (5.2). Thus our main task is to show that (a) any pair of variables of $\bar{\mathcal{P}}$ that occur in a common constraint of $\bar{\mathcal{P}}$ are also found in some common set $\bar{\mathcal{Q}}'_t$, and (b) for any variable x_j of $\bar{\mathcal{P}}$ the set of vertices t of $\bar{\mathcal{T}}$ such that $\bar{\mathcal{Q}}'_t$ contains x_j induces a subtree of $\bar{\mathcal{T}}$.

(a) Consider a pair of variables $\{x_i, x_j\}$ that occur a common constraint of $\bar{\mathcal{P}}$, say in a constraint associated with node u. Then

$$\{x_i, x_j\} \subseteq \bar{\mathcal{X}}_u \cup \bar{\mathcal{X}}_v \cup \bar{\mathcal{X}}_w,$$

where $\{u, v\}$ and $\{u, w\}$ are edges of $\bar{\mathcal{G}}$ (possibly $v = w$). But in that case for any $t \in \bar{T}$ such that $u \in \bar{\mathcal{Q}}_t$ we will have that $\{x_i, x_j\} \subseteq \bar{\mathcal{Q}}'_t$ by construction of the set $\bar{\mathcal{Q}}'_t$ in (5.2).

(b) Here we use Remark 3.1.5. Consider any variable x_j. Let us define

$$C_j \doteq \{u \in V(\bar{\mathcal{G}}) : x_j \in \bar{\mathcal{X}}_u\}$$

and

$$N_j \doteq C_j \cup \{v \in V(\bar{\mathcal{G}}) : \{u, v\} \in E(\bar{\mathcal{G}}) \text{ for some } u \in C_j\}.$$

These sets are relevant because we defined $\bar{\mathcal{Q}}'_t$ such that $x_j \in \bar{\mathcal{Q}}'_t$ iff $\mathcal{Q}_t$ intersects N_j. But we note that C_j induces a connected subgraph of $\bar{\mathcal{G}}$ – in Section 5.1 this attribute was called the *connectedness requirement* for an **NPO**. As a result, N_j also induces a connected subgraph of $\bar{\mathcal{G}}$. If we apply the Remark 3.1.5 to N_j we obtain that, indeed, the set of t such that $x_j \in \bar{\mathcal{Q}}'_t$ induces a subtree of $\bar{\mathcal{T}}$, as desired.

■

In the following sections we will follow the approach to proving Theorem 5.1.3 suggested above, beginning with the reformulation of an **NPO** into an equivalent **NPO** on a network with degree ≤ 3 node, aiming toward the use of Theorem 5.2.1.

5.3 Example of NPO reformulation

As motivation for the general recipe we just outlined, we will argue that the problem in Example 5.1.1 can be reformulated as an equivalent **NPO** on network of maximum degree three (and with small tree-width). And then, using the strategy in Theorem 5.2.1 we will argue that the intersection graph of such reformulation has small tree-width.

To fix ideas, let us consider the case $n = 4$. We will first produce the equivalent **NPO** by first constructing an extended formulation equivalent to $\sum_{j=1}^{4} jx_j^3 \geq 1$. Later we will show that this extended formulation amounts to a new **NPO**.

The extended formulation has additional variables y_j $(1 \leq j \leq 7)$, and the following system of constraints whose sum yields $\sum_{j=1}^{4} jx_j^3 \geq 1$.

$$y_j \leq jx_j^3 - \frac{1}{4}, \quad 1 \leq j \leq 4, \tag{5.3a}$$

$$y_5 \leq y_1 + y_2, \quad y_6 \leq y_3 + y_4, \quad y_7 \leq y_5 + y_6, \tag{5.3b}$$

$$y_7 \geq 0. \tag{5.3c}$$

Effectively, this system splits the sum $\sum_{j=1}^{4} jx_j^3$ into partial sums with two terms each; y_5 and y_6 are stand-ins for these partial sums and y_7 represents the complete sum[3].

System (5.3) is equivalent to $\sum_{j=1}^{4} jx_j^3 \geq 1$ in the sense that the projection to x-space of the set of solutions to (5.3) equals $\{x \in \mathbb{R}^4 : \sum_{j=1}^{4} jx_j^3 \geq 1\}$. Let us put aside, for the moment, the issue that this equivalence might require some of the y_j to take values outside of the range $[0, 1]$, which is not allowed in our formal definition (5.1) for an **NPO**. Modulo this point, we can argue that (5.3) is the system of constraints for an **NPO**. To construct this **NPO** we use a binary tree with nodes $1, \ldots, 7$ shown in Figure 5.2.

The variables for this **NPO** will be all the x_j and y_j. We associate with node 5 a family of arc polynomials that will yield (5.3a) for $j = 1, 2$ and the first inequality in (5.3b). Namely, we associate with node 5 the set of variables $\{x_1, \ldots, x_4\} \cup \{y_1, y_2, y_5\}$, define the arc polynomials

$$p_{(5,1)}^{(1)} = -y_1 + x_1^3 - \frac{1}{4}, \qquad p_{(5,2)}^{(1)} = 0,$$
$$p_{(5,2)}^{(2)} = -y_2 + 2x_2^3 - \frac{1}{4}, \qquad p_{(5,1)}^{(2)} = 0,$$

[3]This "splitting" technique is reminiscent of sparsification techniques for linear systems [52].

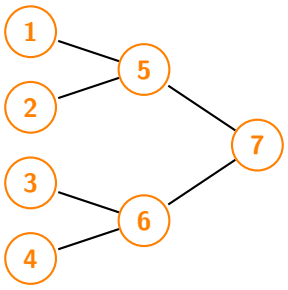


Figure 5.2: Binary tree replacement for star with 4 leaves.

$$p^{(3)}_{(5,1)} = -\frac{1}{2}y_5 + y_1, \qquad p^{(3)}_{(5,2)} = -\frac{1}{2}y_5 + y_2,$$

and impose the **NPO** constraints (associated with node 5):

$$p^{(k)}_{(5,1)} + p^{(k)}_{(5,2)} \geq 0, \quad k = 1, 2, 3. \tag{5.4}$$

Likewise, we associate with node 6 the set $\{x_1, \ldots, x_4\} \cup \{y_3, y_4, y_6\}$, define

$$\begin{aligned} p^{(1)}_{(6,3)} &= -y_3 + 3x_3^3 - \frac{1}{4}, & p^{(1)}_{(6,4)} &= 0, \\ p^{(2)}_{(6,4)} &= -y_4 + 4x_4^3 - \frac{1}{4}, & p^{(2)}_{(6,3)} &= 0, \\ p^{(3)}_{(6,3)} &= -\frac{1}{2}y_6 + y_3, & p^{(3)}_{(6,4)} &= -\frac{1}{2}y_6 + y_4, \end{aligned}$$

and impose

$$p^{(k)}_{(6,3)} + p^{(k)}_{(6,4)} \geq 0, \quad k = 1, 2, 3, \tag{5.5}$$

which yields (5.3a) for $j = 3, 4$ and the second inequality in (5.3b). Finally we associate with node 7 the variables $\{x_1, \ldots, x_4\} \cup \{y_7\}$, define

$$p^{(1)}_{(7,5)} = y_5 - \frac{1}{2}y_7, \quad p^{(1)}_{(7,6)} = y_6 - \frac{1}{2}y_7, \tag{5.6}$$

and set the constraint

$$p^{(1)}_{(7,5)} + p^{(1)}_{(7,6)} \geq 0. \tag{5.7}$$

This yields the third inequality in (5.3b). The bound $y_7 \geq 0$ is already implicit in the definition of an **NPO**. Thus, indeed, system (5.3) arises as the constraint set for an **NPO**.

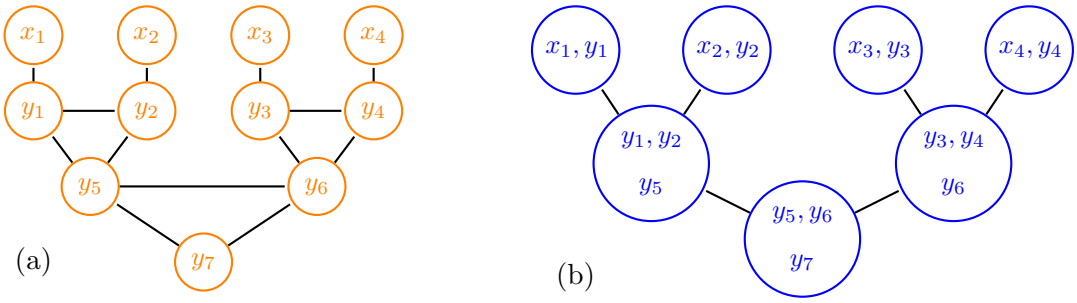


Figure 5.3: (a) Intersection graph for reformulation of Ex. 5.1.1. (b) A tree-decomposition.

Next we claim that as a *general* polynomial optimization problem, the problem with constraints (5.4)-(5.7) has an intersection graph of tree-width 2.

The intersection graph is shown in Figure 5.3 (a) and a tree decomposition, of width 2, is given in Figure 5.3 (b). Note that the tree in Figure 5.3 (b) is isomorphic to that in Figure 5.2 – this is not an accidental event and, rather, reflects the structure of constraints (5.4)-(5.7). It is clear that the above process can be applied to the general case of $\sum_{j=1}^{n} jx_j^3 \geq 1$ so as to always yield a reformulation as an **NPO** on a binary tree with n leaves, a y_j variable for each internal node (and so less than n y_j variables), and such that the intersection graph has tree-width 2.

In summary, system (5.3) represents a natural way to rewrite our original **NPO** as a problem with structured sparsity. To complete this argument, however, we return to the aforementioned issue that the equivalence between system (5.3) and $\sum_{j=1}^{4} jx_j^3 \geq 1$ might require that some y_j falls outside $[0, 1]$. We handle this issue through a further reformulation, using the familiar trick of representing a real variable as the difference between two nonnegative variables and scaling. System (5.8) implements these ideas; we discuss the choice of scaling factors below:

$$(j + 1/4)(\pi_j - \mu_j) \leq jx_j^3 - 1/4, \qquad 1 \leq j \leq 4 \tag{5.8a}$$

$$(3 + 1/2)(\pi_5 - \mu_5) \leq (1 + 1/4)(\pi_1 - \mu_1) + (2 + 1/4)(\pi_2 - \mu_2), \tag{5.8b}$$

$$(7 + 1/2)(\pi_6 - \mu_6) \leq (3 + 1/4)(\pi_3 - \mu_3) + (4 + 1/4)(\pi_4 - \mu_4), \tag{5.8c}$$

$$11\,\pi_7 \leq (3 + 1/2)(\pi_5 - \mu_5) + (7 + 1/2)(\pi_6 - \mu_6), \tag{5.8d}$$

$$x \in [0, 1]^4,\ \pi, \mu \in [0, 1]^6,\ p_7 \in [0, 1]. \tag{5.8e}$$

The scaling constant $j+1/4$ in (5.8a) can intuitively be justified by noting that if $0 \leq x_j \leq 1$ then (5.8a) allows us to assume that $\pi_j \leq 1$ and $\mu_j \leq 1$. Indeed, we could have used $j-1/4$ for this purpose; but the choice of $j+1/4$ will simplify language in our proofs later. The $j+1/4$ constants are then carried into the right-hand side of (5.8b) and (5.8c) with e.g. the coefficients $3+1/2$ chosen so that we can argue that without loss of generality $0 \leq \pi_5, \mu_5 \leq 1$, and similarly with π_6, μ_6. Finally in (5.8d) we use the scaling multiplier of 11 so that $\pi_7 \leq 1$.

We can verify that the projection of the feasible set for (5.8) to x-space is $\{x \in [0,1]^4 : \sum_{j=1}^4 jx_j^3 \geq 1\}$: first, summing the inequalities in (5.8a)-(5.8d), and using $\pi_7 \geq 0$ yields $\sum_{j=1}^4 jx_j^3 - 1 \geq 0$. Conversely, given x such that $\sum_{j=1}^4 jx_j^3 - 1 \geq 0$ we can choose π, μ so that (5.3) holds using the following procedure. For $1 \leq j \leq 4$ we set $\pi_j = \frac{4}{4j+1} \max\{jx_j^3 - \frac{1}{4}, 0\} \in [0,1]$ and $\mu_j = \frac{4}{4j+1} \max\{-jx_j^3 + \frac{1}{4}, 0\} \in [0,1]$, $\pi_5 = \frac{2}{7} \max\{(1+1/4)(\pi_1 - \mu_1) + (2+1/4)(\pi_2 - \mu_2), 0\} \in [0,1]$, and so on.

To complete the switch from the y to the π, μ variables, in the tree-decomposition in Figure 5.3 (b) we replace each y_j by the pair π_j, μ_j for $1 \leq j \leq 6$ (and y_7 with π_7), obtaining a tree-decomposition for the intersection graph of constraints (5.8) of width 5.

In summary, we have verified that the **NPO** in Example 5.1.1 can be reformulated as an equivalent **NPO** whose intersection graph has small tree-width. As a result, Theorem 4.1.2 yields a polynomial-size linear program that approximates the latter **NPO**.

5.4 NPO reformulation: general case

In the following sections we will construct a proof for Theorem 5.1.3. As Example 5.1.1 shows, given an **NPO** on a network $\mathcal{G}$ it may be difficult to directly apply Theorem 4.1.2 so as to obtain Theorem 5.1.3 if $\mathcal{G}$ has nodes of high (graph-theoretic) degree. Our method for handling Example 5.1.1 effectively *splits* the high-degree node 0 into a binary tree and appropriately reformulate the **NPO**.

Our general methodology builds on these ideas using the following strategy:

(s.1) We will present a general technique for reformulating any **NPO** as an equivalent **NPO** on a network where every node has degree at most three, obtained through a sequence of node splitting operations. In general, several such reformulations will exist.

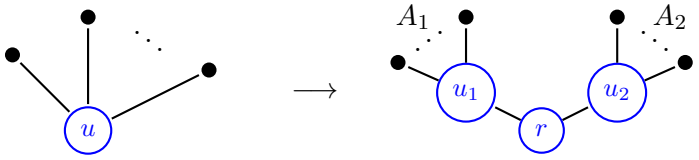


Figure 5.4: Node splitting

(s.2) If $\mathcal{G}$ has small tree-width, we will show that there is a specific reformulation as in (s.1) where the resulting **NPO**, as a general polynomial optimization problem, has an intersection graph of small tree-width.

In Sections 5.4.1 and 5.4.2 we will focus on (s.1); here we will essentially repeat the technique used above in the reformulation of Example 5.1.1. Section 5.4.3 outlines how the proof of Theorem 5.1.3 is obtained if (s.2) is assumed, and Section 5.4.4 focuses on (s.2) and will complete the proof.

5.4.1 Transforming the network and reformulating

In this section we discuss a procedure that, given an **NPO** on a network $\mathcal{G}$, yields an equivalent **NPO** on a new network where every node has degree at most three. This procedure will operate sequentially, at each step modifying the current network so as to decrease by one the number of nodes of degree larger than three, while at the same time producing an **NPO** on the modified network which is equivalent to the original **NPO**.

We use the idea of node splitting, which we first discuss informally in a more general context than Example 5.1.1. Suppose $u \in V(\mathcal{G})$ has degree larger than three and consider a partition of $\delta(u)$ into two sets A_1, A_2. We obtain a new network $\mathcal{G}'$ from $\mathcal{G}$ by replacing u with three new nodes, u_1, u_2 and r, introducing the edges $\{u_1, r\}$ $\{r, u_2\}$ and replacing each edge $\{u, v\} \in A_i$ (for $i = 1, 2$) with $\{u_i, v\}$. See Figure 5.4.

Iterating this procedure, given $u \in V(\mathcal{G})$ with $\deg_{\mathcal{G}}(u) > 3$, we can replace u and the set of edges $\delta(u)$ with a tree, where each internal node will have degree equal to three except for the special node r. See Figure 5.5.

In this figure, a degree-5 node u and the set of edges $\{(u, v_i) : 1 \leq i \leq 5\}$ is converted into a tree with four internal nodes (r, a, b and c). There is a one-to-one correspondence between

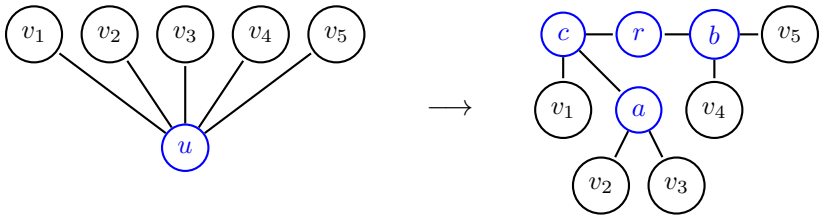


Figure 5.5: Complete node splitting

the edges $\{u, v_i\}$ in the original network, and edges of the tree incident with leaves. Also note that this operation does not change the degree of the v_i.

In general there are several possible complete node splittings, and alternate sequences of node splittings can produce networks with dramatically different tree-width, an issue that we will tackle in Section 5.4.4.

Formally, the splitting operation goes as follows. Consider $u \in V(\mathcal{G})$ with $\deg_{\mathcal{G}}(u) > 3$. Let $\hat{T}_u$ be an arbitrary tree where

(t.1) The set of non-leaf or *internal* nodes of $\hat{T}_u$ is disjoint from $V(\mathcal{G})$. Further, the set of leaves of $\hat{T}_u$ is $\{v \in V(\mathcal{G}) \ : \ \{u, v\} \in E(\mathcal{G})\}$.

(t.2) Each internal node of $\hat{T}_u$ has degree equal to three, except for a special node r (the root), of degree two.

Completely splitting u *using* $\hat{T}_u$ yields a new network, $\mathcal{G}'$ where $V(\mathcal{G}')$ consists of $V(\mathcal{G}) \setminus \{u\}$ together the internal nodes of $\hat{T}_u$, and $E(\mathcal{G}') \ = \ (E(\mathcal{G}) \setminus \delta_{\mathcal{G}}(u)) \cup E(\hat{T}_u)$. For convenience, below we will think of $\hat{T}_u$ as *rooted* at r (i.e., with all edges oriented away from r). Note that the degree of any $v \in V(\mathcal{G})$ with $v \neq u$ is unchanged in this process, thus we can iteratively apply this process to any node with degree four or more. Such algorithm will terminate with a network where each node has degree at most three. Each complete splitting operation will be accompanied by a reformulation of the **NPO** that yields an equivalent **NPO**. This reformulation is described next.

Assume a complete splitting at $u \in V(\mathcal{G})$ using $\hat{T}_u$. First, at all nodes $v \neq u$ of $\mathcal{G}$ the original **NPO** remains unchanged. For each internal node i of $\hat{T}_u$, in our reformulation we first have:

(v.0) All variables in $\mathcal{X}_u$ are associated with i in the new **NPO**.

Let $k \in \mathcal{K}_u$ be the index representing a constraint (5.1b) that is associated with u. We will replace this constraint with a family of constraints associated with the internal nodes of $\hat{T}_u$. This will be done in steps (v.k), (c.1.k) and (c.2.k) given next. Define

$$\mathcal{N}_{i,k} = \sum \left\{ \|p^{(k)}_{(u,v)}\|_1 \,:\, \{u,v\} \in \delta(u),\, v \text{ a descendant of } i \text{ in } \hat{T}_u \right\}$$

The quantities 5/4, 5/2 and 5 that we saw in (5.8) are precisely the $\mathcal{N}_{i,k}$ for Example 5.1.1. Note that if i is an internal node of $\hat{T}_u$ with children j, l, then our definition implies that $\mathcal{N}_{i,k} = \mathcal{N}_{j,k} + \mathcal{N}_{l,k}$.

We now associate the following additional variables with node i:

(v.k) If $i \neq r$ we additionally associate two variables, $\pi_{i,k},\ \mu_{i,k} \in [0,1]$. If $i = r$ we associate a single additional variable $\pi_{r,k} \in [0,1]$ with r.

If i has a child v that is a *leaf*, and hence $\{u,v\} \in \delta_{\mathcal{G}}(u)$, then we associate with i two additional variables $\pi_{v,k},\ \mu_{v,k} \in [0,1]$. We emphasize that these variables are associated with i but not with v.

Next we describe the arc polynomials. We omit polynomials that are identically zero.

(c.1.k) Suppose v is a child of i which is a leaf, i.e. $\{u,v\} \in \delta_{\mathcal{G}}(u)$. Then we define the arc polynomial $\|p^{(k)}_{(u,v)}\|_1(\pi_{v,k} - \mu_{v,k})$ and write the constraint (also associated with i)

$$\|p^{(k)}_{(u,v)}\|_1 \left(\pi_{v,k} - \mu_{v,k}\right) \quad \leq \quad p^{(k)}_{(u,v)}(x). \tag{5.9}$$

[See constraint (5.8a).] We note that this constraint is well-defined as an **NPO** constraint, because by (v.0) and (v.k) all variables in (5.9) are associated with node i or with node v.

(c.2.k) Let j, l be the children of i. Then we define the (linear) arc polynomials $\mathcal{N}_{j,k}\left(\pi_{j,k} - \mu_{j,k}\right)$ and $\mathcal{N}_{l,k}\left(\pi_{l,k} - \mu_{l,k}\right)$, associated with edges $\{i,j\}$ and $\{i,k\}$, respectively.

If $i \neq r$, then we define the additional arc polynomial $\mathcal{N}_{i,k}(\pi_{i,k} - \mu_{i,k})$ (which can be associated with any edge incident with i) and write the constraint

$$\mathcal{N}_{i,k}\,(\pi_{i,k} - \mu_{i,k}) \;\leq\; \mathcal{N}_{j,k}\,(\pi_{j,k} - \mu_{j,k}) + \mathcal{N}_{l,k}\,(\pi_{l,k} - \mu_{l,k}). \tag{5.10}$$

[See constraints (5.8b), (5.8c).]

If on the other hand $i = r$, then we write the constraint [see (5.8d)]

$$\mathcal{N}_{r,k}\,\pi_{r,k} \;\leq\; \sum_{s=j,l} \mathcal{N}_{s,k}(\pi_{s,k} - \mu_{s,k}). \tag{5.11}$$

We have just described how the variables and constraints associated with node u in the original **NPO** are to be reformulated in the new **NPO**. To complete the description of the new **NPO** we consider a generic node $v \in V(\mathcal{G})$ with $v \neq u$. Formally,

(v') We keep the same set $\mathcal{X}_v$ of variables associated with v.

We also keep the same set of constraints $\mathcal{K}_v$ associated with v. This is done by, effectively, keeping the same set of arc polynomials $p_{(v,w)}$. Specifically, consider any edge $\{v, w\} \in E(\mathcal{G}')$:

(c'.1) If $w \neq u$, the original arc polynomials $p_{(v,w)}$ are arc polynomials in the reformulated problem because of (v') above.

(c'.2) If $w = u$ then as discussed in item (t.1) above, there is one edge $\{i, v\}$ of $\hat{T}_u$. By item (v.0), all variables in $\mathcal{X}_u$ are associated with node i. Hence, we any arc polynomial $p_{(v,u)}$ in the original **NPO** can be viewed as an arc polynomial $p_{(v,i)}$ in the reformulated **NPO**.

5.4.2 Validity of the reformulation

Here we consider a single node u and and index $k \in \mathcal{K}(u)$ as in Section 5.4.1. Lemma 5.4.1 proves that the original and reformulated **NPO**s are equivalent, and Lemma 5.4.3 discusses the quality of the approximation to the first **NPO** provided by an approximate solution to the second.

Lemma 5.4.1 *(a) Suppose* $(\hat{\pi}, \hat{\mu}, \hat{x})$ *satisfies constraints* (5.10), (5.11), (5.9) *and additionally* $\hat{\pi}_{r,k} \geq 0$. *Then* x *satisfies the constraint* $\sum_{\{u,v\}\in\delta_{\mathcal{G}}(u)} p^{(k)}_{(u,v)}(x) \geq 0$.

(b) Conversely, if $\sum_{\{u,v\}\in\delta_{\mathcal{G}}(u)} p^{(k)}_{(u,v)}(\hat{x}) \geq 0$ *with* $\hat{x}_j \in [0,1]$ *for all* j, *then we can find* $(\hat{\pi}, \hat{\mu}, \hat{\pi}_{r,k})$ *that satisfy* (5.10), (5.11), (5.9) *with all variables in the range* $[0,1]$.

PROOF:

(a) This fact follows because summing (5.10), (5.11), (5.9) yields

$$\sum_{\{u,v\}\in\delta_{\mathcal{G}}(u)} p^{(k)}_{(u,v)}(\hat{x}) \;\geq\; \mathcal{N}_{r,k}\,\pi_{r,k}.$$

(b) This fact is obtained as in the discussion following equations (5.8). We proceed, inductively, up the tree, choosing values $\hat{\pi}$ and $\hat{\nu}$ so that all constraints (5.10) and (5.9) hold as equality, and so that at each internal node i, $\hat{\pi}_{i,k} \geq 0$, $\hat{\mu}_{i,k} \geq 0$ and $\hat{\pi}_{i,k}\hat{\mu}_{i,k} = 0$. Inductively, by definition of the constants $\mathcal{N}_{i,k}$, no variable will ever exceed 1. At termination we will be able to likewise set $\hat{\pi}_{r,k}$ so that (5.11) also holds an equation.

■

Part (a) of Lemma 5.4.1 shows that a feasible solution for the updated **NPO** gives rise to a feasible solution to the original **NPO**. If, however, we were to apply Theorem 4.1.2 to the new **NPO** we would only obtain an approximately feasible solution $(\hat{\pi}, \hat{\mu}, \hat{x})$ and we need to address the infeasibility of $\hat{x}$ with respect to the original **NPO**. This issue is handled by Lemma 5.4.3. We first need a technical result.

Lemma 5.4.2 *The sum of* 1*-norms of all constraints* (5.10), (5.11), (5.9) *is at most*

$$4\deg_{\mathcal{G}}(u) \sum_{(u,v)\in\delta_{\mathcal{G}}(u)} \|p^{(k)}_{(u,v)}\|_1.$$

PROOF: Write $d = \deg_{\mathcal{G}}(u)$. The 1-norm of a constraint (5.10) at an internal node i (or (5.11) if $i = r$) equals $2\mathcal{N}_{i,k}$. Since all internal nodes of $\hat{T}_u$, except for r, have degree three in $\hat{T}_u$ and $\hat{T}_u$ has d leaves, the total number of internal nodes equals $d-1$. Thus the sum of 1-norms of all constraints (5.10) or (5.11) is at most $2d\sum_{(u,v)\in\delta_{\mathcal{G}}(u)} \|p^{(k)}_{(u,v)}\|$. A similar observation regarding constraints (5.9) yields the result. ■

Lemma 5.4.3 *Let $\gamma > 0$. Suppose $(\hat{\pi}, \hat{\mu}, \hat{x})$ is γ-scaled feasible for constraints* (5.10), (5.11), (5.9)*, and satisfies $\hat{\pi}_{r,k} \geq 0$. Then*

$$\sum_{\{u,v\}\in\delta_{\mathcal{G}}(u)} p^{(k)}_{(u,v)}(\hat{x}) \geq -4\deg_{\mathcal{G}}(u)\,\gamma \sum_{\{u,v\}\in\delta_{\mathcal{G}}(u)} \|p^{(k)}_{(u,v)}\|_1,$$

i.e. $\hat{x}$ is $(4\deg_{\mathcal{G}}(u)\,\gamma)$-scaled feasible for constraint $\sum_{\{u,v\}\in\delta_{\mathcal{G}}(u)} p^{(k)}_{(u,v)}(x) \geq 0$.

PROOF: Writing a generic constraint (5.10), (5.11), (5.9) as $f(\pi, \mu, x) \geq 0$ we have by assumption that $f(\hat{\pi}, \hat{\mu}, \hat{y}_r, \hat{x}) \geq -\|f\|_1\gamma$. Since the sum of all constraints (5.10), (5.11), (5.9) yields

$$\sum_{(u,v)\in\delta_{\mathcal{G}}(u)} p^{(k)}_{(u,v)}(x) \;\geq\; \mathcal{N}_{r,k}\,\pi_{r,k}$$

we therefore obtain

$$\sum_{(u,v)\in\delta_{\mathcal{G}}(u)} p^{(k)}_{(u,v)}(\hat{x}) \;\geq\; \mathcal{N}_{r,k}\,\hat{\pi}_{r,k} \;-\; S\gamma,$$

where S is the sum of L_1 norms of all constraints (5.10), (5.11), (5.9). Applying Lemma 5.4.2 yields the desired result. ■

We can now comment on the quality of an approximate solution to the reformulation, in terms of the initial problem. Let us suppose that given an **NPO** on a network $\mathcal{G}$ (we will term this the *initial* **NPO**) we apply a sequence of complete node splittings and reformulations so as to obtain a *final* **NPO**. In what follows, we will denote $\mathcal{P}^f$ is the final **NPO**, and $\mathcal{G}^f$ the final network.

Lemma 5.4.4 *$\mathcal{P}^f$ is equivalent to the initial **NPO** and a solution that is γ-scaled feasible for the constraints in $\mathcal{P}^f$ yields a $(4D\gamma)$-scaled feasible solution to the initial **NPO**, where*

$$D = \max_{v\in V(\mathcal{G})} \deg_{V(\mathcal{G})}(v).$$

PROOF: Equivalence follows by applying Lemma 5.4.1 inductively. Applying Lemma 5.4.3 inductively proves the approximation error bound. ■

For future reference we also state the following result concerning the size of the reformulated **NPO**:

Lemma 5.4.5 *Let $\Delta = \max_{v \in V(\mathcal{G})}\{|\mathcal{K}_v|\}$. Suppose the original **NPO** has n variables in total, and $|E(\mathcal{G})| = m$. Then $\mathcal{G}^f$ has $O(m)$ nodes and edges. Further, the number of variables and constraints in $\mathcal{P}^f$*

$$n \; + \; O\left(\sum_{u \in V(\mathcal{G})} \deg_{\mathcal{G}}(u)|\mathcal{K}_u|\right) \; = \; n + O(D\Delta). \tag{5.12}$$

PROOF: Since $\sum_{v \in V(\mathcal{G})} \deg_{\mathcal{G}}(v) = 2m$ we obtain that $\mathcal{G}^f$ has the desired size. The variables for $\mathcal{P}^f$ are the original variables, plus all π and μ variables, which are accounted for in the second term of (5.12). ■

In the next section we group all arguments together, including the approximate LP formulations that arise from applying Theorem 4.1.2 to $\mathcal{P}^f$.

5.4.3 Implications to Theorem 5.1.3

Let us follow the above recipe to obtain a final reformulated **NPO** $\mathcal{P}^f$ on a network $\mathcal{G}^f$. We will now apply Theorem 4.1.2 by treating $\mathcal{P}^f$ as a general polynomial optimization problem of the form **PO**. In this section we address the size of the resulting LP and its approximation guarantees. In particular, suppose that we want to produce a solution for the initial **NPO** within tolerance ϵ, i.e. ϵ-scaled feasible and with objective value exceeding the optimum by at most $\epsilon\|c\|_1$.

1. Using Lemma 5.4.4 we see that we need to find a solution to $\mathcal{P}^f$ within tolerance γ, where $\gamma = \epsilon/(4D)$.

2. Suppose we have a tree-decomposition of $\mathcal{G}^f$ of width $W \geq 1$, say. Then by Theorem 5.2.1, there is a tree-decomposition of the intersection network of $\mathcal{P}^f$ of width $\leq O(\Delta W)$.

3. The number of variables in $\mathcal{P}^f$, by Lemma 5.4.5, is $n + O(D\Delta)$.

4. Let ρ be the maximum degree of any of the arc polynomials in the initial **NPO** (also the maximum degree of arc polynomials in $\mathcal{P}^f$). We apply Theorem 4.1.2 to find a

solution to $\mathcal{P}^f$ as per item 1 above. This step will produce an LP of size

$$O\left((2\rho/\gamma)^{O(\Delta W)+1}(n+D\Delta)\log(D\rho/\gamma)\right) = O\left((D\rho/\epsilon)^{O(\Delta W)+1} n \log(\rho/\epsilon)\right).$$

Thus the construction *almost* yields Theorem 5.1.3. The key issue is the parameter W in this estimation. This is the width of a tree-decomposition of the final network $\mathcal{G}^f$ rather than the initial network $\mathcal{G}$. We are thus faced with a purely graph-theoretic issue: given an arbitrary network, is there a sequence of complete node splittings that render a network with maximum degree at most three, and without a large increase in tree-width?

This is precisely what will be done in the next section. We will prove that if the initial network has a tree-decomposition of width ω then there is a (polynomial-time constructible) network $\mathcal{G}^f$ with a tree-decomposition of width $O(\omega)$. Combining this fact with the above observations yields a proof of Theorem 5.1.3.

5.4.4 Constructing reformulations on small tree-width networks

Consider the example given in Figure 5.6. Here the vertices of the graph shown in (a) are arranged into $k > 1$ columns. The odd-numbered columns have k vertices each, which induce a path, while the even-numbered columns have a single vertex which is adjacent to all vertices in the preceding and following columns. It can be shown that this graph has tree-width 3. In (b) we show the outcome after splitting vertices so that the maximum degree is four. This second graph has tree-width k, and further splitting the degree-four vertices will not change this fact.

In contrast to this situation, suppose that we split the graph in Figure 5.6(a) in two steps as shown in Figure 5.7. The tree-width of the final graph is also 3. The difference between Figures 5.6 and 5.7 is explained by the fact that the splitting initiated by the "first step" in Figure 5.7 exploits the tree-decomposition of width 3 of the graph Figure 5.6(a).

Next we turn to a formal approach that produces the desired outcome in the general setting. Given a graph G, a *simplification* of G will be graph $\bar{G}$ obtained by a sequence of complete vertex splittings, such that the maximum degree of a vertex in $\bar{G}$ is ≤ 3. The following Lemma will show how to obtain a simplification of a graph via a vertex splitting that maintains tree-width up to a constant factor. In the proof, the trees $\hat{T}_u$ that yield the

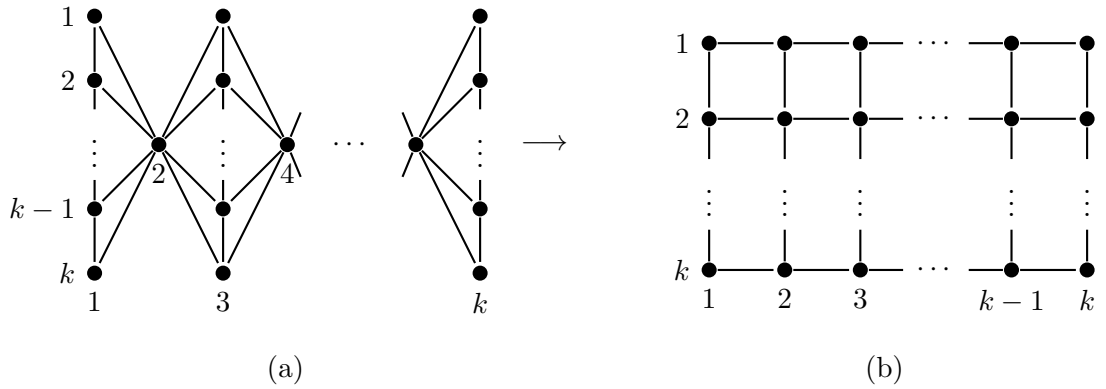


Figure 5.6: Incorrect vertex splitting.

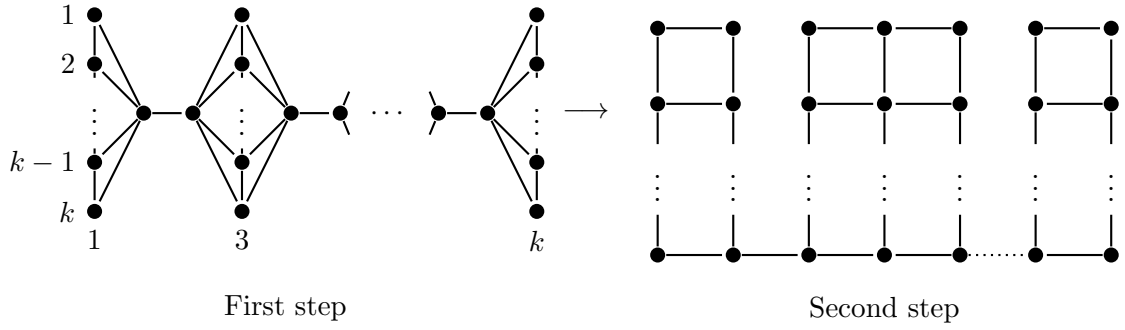


Figure 5.7: Correct vertex splitting of graph in Figure 5.6(a).

general splitting procedure described in Section 5.4.4 will be explicitly defined.

Lemma 5.4.6 *Let G be a connected undirected graph and (T, Q) a tree-decomposition of G of width Z. Then there is a simplification $\bar{G}$ of G and a tree-decomposition $(\bar{T}, \bar{Q})$ of $\bar{G}$ of width at most $2Z + 1$.*

PROOF: We first modify (T, Q) in a sequence of steps.

Step 1. For any edge $e = \{u, v\} \in E(G)$, choose an arbitrary $t \in V(T)$ with $e \subseteq Q_t$. Then we modify T by adding to T a new vertex, t^e and the edge $\{t^e, t\}$. Further, we set $Q_{t^e} = \{u, v\}$.

Step 2. Without loss of generality, every vertex of T has degree at most 3. To attain this condition, consider any $t \in V(T)$ with $\delta_T(t) = \{s_1, \ldots, s_d\}$ (say) where $d > 3$. Then we alter T by replacing t with two adjacent vertices t^1 and t^2, such that t^1 is also adjacent to s_1 and s_2 and t^2 is adjacent to $s_3, \ldots, s_d$. Finally, we set $Q_{t^1} = Q_{t^2} = Q_t$. Continuing inductively we will attain the desired condition.

Step 3. For any vertex $u \in V(G)$ let T_u be the subtree of T consisting of vertices t with $u \in Q_t$, and $\breve{T}_u$ be the subtree of T_u that spans $\{t^e : e \in \delta_G(u)\}$ (which is a subset of the leaves of T_u). Then we modify (T, Q) by replacing T_u with $\breve{T}_u$, yielding a new tree-decomposition of same or smaller width. In other words, without loss of generality *every* leaf of T_u is of the form t^e for some $e \in \delta_G(u)$.

We can now describe our vertex splitting scheme. Consider $u \in V(G)$ with $\deg_G(u) > 3$. We say that a vertex of T_u is *blue* if it is either a leaf or of degree three in T_u. Now we form the tree $\hat{T}_u$ whose vertex-set is the set of blue vertices of T_u, and whose edge-set is obtained as follows. By construction, $E(T_u)$ can be partitioned into a set of paths whose endpoints are blue and which contain no other blue vertices. For each such path, with endpoints a and b (say), the tree $\hat{T}_u$ will contain the edge $\{a, b\}$ (in other words, T_u can be obtained from $\hat{T}_u$ by subdividing some edges and so T_u and $\hat{T}_u$ are topologically equivalent). Note that $\hat{T}_u$ has $\deg_G(u)$ leaves, each internal vertex with degree 3, and for each edge $\{u, v\} \in E(G)$ there is

one pendant edge, as needed.

Let $\breve{G}$ be the graph obtained by the complete splitting of u using $\hat{T}_u$. For each internal vertex $t \in V(\hat{T}_u)$ we name u^t the corresponding new vertex in $\breve{G}$, to emphasize that each non-leaf vertex in $\hat{T}_u$ will create a vertex in the complete splitting (recall that the leaves in $\hat{T}_u$ will correspond to the neighbors of u). This operation does not change the degree of any vertex $v \in V(G)$ with $v \neq u$. The eventual graph $\bar{G}$ in the proof will be obtained by applying a complete splitting of this type at every vertex of degree > 3 in G.

Returning to $\breve{G}$, we construct, for it, a tree decomposition $(T, \breve{Q})$ as follows. First, let us regard the tree T_u as rooted at some internal blue vertex $r(u)$. For a vertex $t \in V(T_u)$ let $\mathcal{A}_u(t)$ be the closest blue *ancestor* of t in T_u; we write $\mathcal{A}_u(r(u)) = r(u)$. Then, for $t \in V(T)$, we set

$$\breve{Q}_t \;=\; \begin{cases} (Q_t - u) \cup \{u^{\mathcal{A}_u(t)}\}, & \text{if } t \in V(T_u) \text{ and } \deg_{T_u}(t) = 2\ , \\ (Q_t - u) \cup \{u^t, u^{\mathcal{A}_u(t)}\}, & \text{if } t \in V(T_u) \text{ and } \deg_{T_u}(t) \neq 2, \\ Q_t, & \text{if } t \notin V(T_u). \end{cases} \tag{5.13}$$

Now we argue that $(T, \breve{Q})$ is a tree-decomposition of $\breve{G}$. To see this, note that if $t \in V(\hat{T}_u)$ then $u^t \in \breve{Q}_s$ iff $s = t$ or s is a child of t in $\hat{T}_u$, thus the endpoints of any edge $\{u^t, u^s\}$, where t is the parent of s, will be contained in $\breve{Q}_s$. Further, for any edge of $e = \{u, v\}$ of G, by Step 3 above there will be a leaf t^e of T_u such that the edge $\{t^e, \mathcal{A}_u(t^e)\} \in E(\hat{T}_u)$. This corresponds to a pendant edge $\{u^{\mathcal{A}_u(t^e)}, v\} \in E(\breve{G})$ and by construction both $v \in \breve{Q}_{t^e}$ and $u^{\mathcal{A}_u(t^e)} \in \breve{Q}_{t^e}$. The fact that every vertex in $\breve{G}$ induces a connected subgraph in T can be easily verified. This completes the argument $(T, \breve{Q})$ is a tree-decomposition of $\breve{G}$.

Notice that for $v \in V(G)$ with $v \neq u$, the subtree T_v is the same in (T, Q) and $(T, \breve{Q})$, and that $\breve{G}$ has one less node of degree greater than three than G. Thus, iteratively applying the complete splitting of every vertex of $\breve{G}$ of degree greater than three, and modifying the tree-decomposition as in (5.13) will produce a tree-decomposition $(T, \bar{Q})$ of the final graph $\bar{G}$.

By construction, for each $t \in V(T)$ we obtain $\bar{Q}_t$ from Q_t by replacing each element with (at most) two new elements. Thus, since $|Q_t| \leq Z + 1$, the width of $(T, \breve{Q})$ is at most $2(Z + 1) - 1$. ■

Chapter 6

Cutting planes for Polynomial Optimization

In the previous chapter we introduced a theoretical framework, along with tractability results, involving linear programming techniques used to approximate polynomial optimization (**PO**) problems. Tractability was parametrized by *tree-width*, a graph theoretical measure of structured sparsity. In this chapter we follow on the idea of using linear programming to approximate **PO** problems using the well-studied technique of *cutting planes*, widely used in the context of mixed-integer programming. As before, we will be concerned on **PO** problems which we recall for convenience of the reader:

$$(\mathbf{PO}) \ : \ \min \ c^T x \tag{4.1a}$$

$$\text{subject to} : \quad f_i(x) \ \geq \ 0 \qquad 1 \leq i \leq m \tag{4.1b}$$

$$x \in \{0,1\}^p \times [0,1]^{n-p}, \tag{4.1c}$$

where each f_i is a polynomial. In a different, or complementary, spirit from Chapter 4 we will not assume structured sparsity on **PO**. Instead, we will aim at developing *general* techniques that can be *computationally* efficient and stable. To this purpose, we discuss 2 families of cutting planes for **PO** problems: *digitization* and *intersection* cuts.

Digitization cuts make use of the maturity of Mixed-Integer Programming (MIP) solvers in order to find valid cuts to **PO** problems. Using the digitization technique already introduced in Section 4.2, and used there to obtain theoretical tractability results, we will devise a separation procedure for **PO** that uses MIP solvers as a sub-routine. As for Intersection cuts, we will provide a review and details on the computational aspects of the cuts introduced in [23]. There, a generalization of classical Intersection cuts for IP is developed, along with a practical procedure for finding such cuts in the **PO** context.

6.1 Digitization Cuts

6.1.1 Approximations of Polynomial Optimization revisited

Given a **PO** problem, in Section 4.2 we discussed the following *pure binary* approximation:

$$
\begin{aligned}
(\mathbf{GB}(\gamma)): \quad & \min \ \sum_{j=1}^{p} c_j x_j \ + \ \sum_{j=p+1}^{n} c_j \left(\sum_{h=1}^{L_\gamma} 2^{-h} z_{j,h} \right) \\
\text{s.t.} \quad & \sum_{\alpha \in I(i)} f_{i,\alpha} \left[\prod_{j \in Z(i,\alpha)} x_j \prod_{j=p+1}^{n} \left(\sum_{h=1}^{L_\gamma} 2^{-h} z_{j,h} \right)^{\alpha_j} \right] \geq -\epsilon \|f_i\|_1, \ 1 \leq i \leq m \quad (4.5a) \\
& x_j = \sum_{h=1}^{L_\gamma} 2^{-h} z_{j,h}, \ p+1 \leq j \leq n \quad (4.5b) \\
& x_j \in \{0,1\}, \ 1 \leq j \leq p \quad (4.5c) \\
& z_{j,h} \in \{0,1\}, \ p+1 \leq j \leq n, \ 1 \leq h \leq L_\gamma \quad (4.5d)
\end{aligned}
$$

where $\epsilon = 1-(1-\gamma)^\rho$. This approximation can be used to obtain solutions to **PO** that are ϵ-scaled feasible (Definition 4.1.1) and it is based on the fact that, for any $0 \leq r \leq 1$, given $0 < \gamma < 1$ we can approximate r as a sum of inverse powers of 2:

$$
\sum_{h=1}^{L_\gamma} 2^{-h} z_h \ \leq \quad r \quad \leq \sum_{h=1}^{L_\gamma} 2^{-h} z_h \ + \ 2^{-L_\gamma} \ \leq \ \sum_{h=1}^{L_\gamma} 2^{-h} z_h \ + \ \gamma \ \leq \ 1, \tag{6.1}
$$

for some $z_h \in \{0,1\}$, $1 \leq h \leq L_\gamma$ and where

$$
L_\gamma \doteq \lceil \log_2 \gamma^{-1} \rceil.
$$

What **GB(γ)** effectively does is, starting from **PO**, it replaces each continuous variable by a digitized version of it, given by the leftmost expression in (6.1). Using the binary nature of the variables, all non-linearities in **GB(γ)** can be expressed as linear expressions in a lifted space. This follows from the fact that, given $x_1, x_2 \in \{0,1\}$:

$$y = x_1 \cdot x_2 \Longleftrightarrow \max\{x_1 + x_2 - 1, 0\} \leq y \leq \min\{x_1, x_2\}.$$

Thus, with a slight abuse of notation, we will consider **GB(γ)** as a binary *linear* problem. The following result was proved in Section 4.2, which justifies the usefulness of this approximation.

Lemma 4.2.2 *(a) Suppose $\bar{x}$ is feasible for **PO**. Then there is a feasible solution $(\tilde{x}, \tilde{z})$ for **GB(γ)** with objective value at most $c^T\bar{x} + \epsilon\|c\|_1$.*

*(b) Suppose $(\hat{x}, \hat{z})$ is feasible for **GB(γ)**. Then $\hat{x}$ is ϵ-scaled feasible for* (4.1b) *and*

$$c^T\hat{x} = \sum_{j=1}^{p} c_j\hat{x}_j + \sum_{j=p+1}^{n} c_j \left(\sum_{h=1}^{L_\gamma} 2^{-h}\hat{z}_{j,h} \right).$$

In this chapter we will aim at obtaining valid inequalities to **PO** from **GB(γ)**. However, even though Lemma 4.2.2 indicates that **GB(γ)** acts as a form of *relaxation*, a valid inequality for **GB(γ)** might not be valid to **PO**. We justify this in the next example.

Example 6.1.1 *Consider the following simple feasible set for **PO***

$$S = \left\{ (x_1, x_2) \ : x_1^2 + x_2^2 \leq \frac{5}{4},\ (x_1, x_2) \in [0,1]^2 \right\}.$$

*Choose $\gamma = 1/4$, thus $L_\gamma = 2$ and $\epsilon = 0.4375$. Formulating **GB(γ)** renders the following feasible set:*

$$\left(\frac{1}{2}z_{1,1} + \frac{1}{4}z_{1,2}\right)^2 + \left(\frac{1}{2}z_{2,1} + \frac{1}{4}z_{2,2}\right)^2 \leq \frac{5}{4} + \frac{13}{4}\epsilon = 2.671875 \tag{6.2a}$$

$$x_1 = \frac{1}{2}z_{1,1} + \frac{1}{4}z_{1,2} \tag{6.2b}$$

$$x_2 = \frac{1}{2}z_{2,1} + \frac{1}{4}z_{2,2} \tag{6.2c}$$

$$z_{i,j} \in \{0,1\}. \tag{6.2d}$$

Constraint (6.2a) *is redundant, hence the projection onto x-space of* (6.2) *is simply*

$$S' = \left\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\right\}^2.$$

On the other side, note that $(\sqrt{5/8}, \sqrt{5/8}) \in S$. *However, as* $\sqrt{5/8} \approx 0.79$,

$$(\sqrt{5/8}, \sqrt{5/8}) \notin conv(S').$$

This shows that a valid inequality for $conv(S')$ *(e.g,* $x_1 \leq 3/4$*) might be invalid for* S.

The issue with **GB(**γ**)** is that, even though it provides an approximation with provable guarantees, it only uses one side of approximation (6.1). Therefore, there might be feasible points of **PO** outside the convex hull of the feasible region of **GB(**γ**)**. We fix this problem by introducing a more elaborate version of this approximation, which considers both of the leftmost inequalities in (6.1) simultaneously. We call this new approximation $\mathbf{GB}(\gamma)^+$:

$$
\begin{aligned}
(\mathbf{GB}(\gamma)^+): \quad & \min \ \sum_{j=1}^{p} c_j x_j \ + \ \sum_{j=p+1}^{n} c_j \left(\sum_{h=1}^{L_\gamma+1} d_h z_{j,h} \right) \\
\text{s.t.} \quad & \sum_{\alpha \in I(i)} f_{i,\alpha} \left[\prod_{j \in Z(i,\alpha)} x_j \prod_{j=p+1}^{n} \left(\sum_{h=1}^{L_\gamma+1} d_h z_{j,h} \right)^{\alpha_j} \right] \geq -\epsilon \|f_i\|_1, \ 1 \leq i \leq m \quad (6.3a) \\
& x_j = \sum_{h=1}^{L_\gamma+1} d_h z_{j,h}, \ p+1 \leq j \leq n \quad (6.3b) \\
& x_j \in \{0,1\}, \ 1 \leq j \leq p \quad (6.3c) \\
& z_{j,h} \in \{0,1\}, \ p+1 \leq j \leq n, \ 1 \leq h \leq L_\gamma + 1 \quad (6.3d)
\end{aligned}
$$

where $d_h = 2^{-h}$ for $h = 1, \ldots, L_\gamma$ and $d_{L_\gamma+1} = 2^{-L_\gamma}$. $\mathbf{GB}(\gamma)^+$ includes one extra digit in the digitization of each continuous variable, which yields a relaxation with better properties than **GB(**γ**)**. As a starting point, we can easily prove the same result as in Lemma 4.2.2 for $\mathbf{GB}(\gamma)^+$:

Lemma 6.1.2

(a) Suppose $\bar{x}$ *is feasible for* ***PO****. Then there is a feasible solution* $(\tilde{x}, \tilde{z})$ *for* $\boldsymbol{GB}(\gamma)^+$ *with objective value at most* $c^T \bar{x} + \epsilon \|c\|_1$.

(b) Suppose $(\hat{x}, \hat{z})$ *is feasible for* $\boldsymbol{GB}(\gamma)^+$. *Then* $\hat{x}$ *is* ϵ*-scaled feasible for* (4.1b) *and*

$$c^T \hat{x} = \sum_{j=1}^{p} c_j \hat{x}_j + \sum_{j=p+1}^{n} c_j \left(\sum_{h=1}^{L_\gamma+1} d_h \hat{z}_{j,h} \right).$$

PROOF: For part (a), we use the same proof as in Lemma 4.2.2 and set the $z_{j,L_\gamma+1}$ variables to 0. Part (b) is direct. ■

The importance of the extra digit included in $\mathbf{GB}(\gamma)^+$ will become evident in the next Lemma, and it will justify why $\mathbf{GB}(\gamma)^+$ can be used to derive valid cuts for **PO**.

Lemma 6.1.3 *Suppose $\tilde{x}$ is feasible for **PO**. Then there exists (x^i, z^i) $i = 0, \ldots, 2^{n-p} - 1$ feasible solutions for $\mathbf{GB}(\gamma)^+$ such that*

$$\tilde{x} \in conv(\{x^i \mid i = 0, \ldots, 2^{n-p} - 1\}).$$

PROOF: We follow a similar procedure as in the proof of Lemma 4.2.2. We start by constructing (x^0, z^0) to fix ideas. For each $j \in \{1, \ldots, p\}$ define $x^0_j = \tilde{x}_j$, for $j \in \{p+1, \ldots, n\}$ choose $z^0_{j,h}$ $h = 1, \ldots, L_\gamma$ so as to attain the approximation for $\tilde{x}_j$ as in (6.1), and set $z^0_{j,L_\gamma+1} = 0$. We define x^0_j for $j \in \{p+1, \ldots, n\}$ from z^0 according to (6.3b). From the analysis in Lemma 4.2.2 we know (x^0, z^0) is feasible for $\mathbf{GB}(\gamma)^+$. Building (x^0, z^0) can be seen as a "rounding" of $\tilde{x}$ to the nearest *smaller* inverse power of 2, but we can also "round up" using the last digit.

For $i = 1, \ldots, 2^{n-p} - 1$ we define (x^i, z^i) the same way as (x^0, z^0), except for $z^i_{p+j,L_\gamma+1}$, $j = 1, \ldots, n-p$, which we set to 1 if and only if the j-th digit of the binary encoding of i is 1. The idea is to have every possible combination of 0's and 1's in the last digit. From (6.1), for every i:

$$\sum_{h=1}^{L_\gamma} 2^{-h} z^i_{p+j,h} \;\leq\; \tilde{x}_{p+j} \;\leq\; \sum_{h=1}^{L_\gamma} 2^{-h} z^i_{p+j,h} + 2^{-L_\gamma}.$$

We claim that for every i:

$$\prod_{j=1}^{n-p} \left(\sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h} \right)^{\alpha_j} - \epsilon \;\leq\; \prod_{j=1}^{n-p} \tilde{x}^{\alpha_j}_{p+j} \;\leq\; \prod_{j=1}^{n-p} \left(\sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h} \right)^{\alpha_j} + \epsilon. \tag{6.4}$$

For each i we denote $Y_i \subseteq \{1, \ldots, n-p\}$ the digits with 1 in the binary encoding of i. Similarly we denote N_i the digits with 0 in the binary encoding of i. Thus $z^i_{p+j,L_\gamma+1} = 1$ iff $j \in Y_i$. From the proof of Lemma 4.2.2, we have

$$\prod_{j \in N_i} \bar{x}^{\alpha_j}_{p+j} \;\leq\; \prod_{j \in N_i} \left(\sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h} \right)^{\alpha_j} + \epsilon.$$

and $\forall j \in Y_i$

$$\tilde{x}_{p+j} \leq \sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h} \leq 1.$$

This implies

$$\begin{aligned}
\prod_{j=1}^{n-p} \tilde{x}_{p+j}^{\alpha_j} &= \left(\prod_{j\in N_i} \bar{x}_{p+j}^{\alpha_j}\right)\left(\prod_{j\in Y_i} \bar{x}_{p+j}^{\alpha_j}\right) \\
&\leq \left(\prod_{j\in N_i}\left(\sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h}\right)^{\alpha_j} + \epsilon\right) \prod_{j\in Y_i}\left(\sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h}\right)^{\alpha_j} \\
&\leq \prod_{j=1}^{n-p}\left(\sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h}\right)^{\alpha_j} + \epsilon,
\end{aligned}$$

which proves the right-hand side inequality of (6.4). To obtain the left-hand side inequality, we apply Lemma 4.2.1 for the indices $j \in Y_i$. We identify $u_j = \tilde{x}_{p+j}$ and $v_j = \sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h} - \tilde{x}_{p+j} \geq 0$. Clearly, since $j \in Y_i$

$$v_j = 2^{-L_\gamma} + \sum_{h=1}^{L_\gamma} d_h z^i_{p+j,h} - \tilde{x}_{p+j} \leq 2^{-L_\gamma} \leq \gamma$$

therefore,

$$\begin{aligned}
\prod_{j\in Y_i}\left(\sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h}\right)^{\alpha_j} - \prod_{j\in Y_i} \bar{x}_{p+j}^{\alpha_j} &\leq 1 - \prod_{j\in Y_i}(1-v_j)^{\alpha_j} \\
&\leq 1 - \prod_{j\in Y_i}(1-\gamma)^{\alpha_j} \\
&\leq 1 - (1-\gamma)^{\rho} = \epsilon.
\end{aligned}$$

This implies

$$\begin{aligned}
\prod_{j=1}^{n-p} \tilde{x}_{p+j}^{\alpha_j} + \epsilon &= \left(\prod_{j\in N_i} \bar{x}_{p+j}^{\alpha_j}\right)\left(\prod_{j\in Y_i} \bar{x}_{p+j}^{\alpha_j}\right) + \epsilon \\
&\geq \left(\prod_{j\in N_i} \bar{x}_{p+j}^{\alpha_j}\right)\left(\prod_{j\in Y_i} \bar{x}_{p+j}^{\alpha_j} + \epsilon\right)
\end{aligned}$$

$$\begin{aligned} &\geq \left(\prod_{j\in N_i}\left(\sum_{h=1}^{L_\gamma} d_h z^i_{p+j,h}\right)^{\alpha_j}\right)\left(\prod_{j\in Y_i}\left(\sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h}\right)^{\alpha_j}\right) \\ &= \prod_{j=1}^{n-p}\left(\sum_{h=1}^{L_\gamma+1} d_h z^i_{p+j,h}\right)^{\alpha_j}, \end{aligned}$$

which completes the proof of (6.4). It immediately follows that $\forall i \in \{0,\dots,2^{n-p}-1\}$ the vector (x^i, z^i) is feasible for $\mathbf{GB}(\gamma)^+$.

By definition, for each fixed $j \in \{1,\dots,n-p\}$ all $z^i_{p+j,h}$, $h = 1,\dots,L_\gamma$ are equal. The difference only exists in the last digit. Thus we can define

$$C_j = \sum_{h=1}^{L_\gamma} 2^{-h} z^i_{p+j,h}.$$

Now define the polyhedron

$$P_\gamma(\tilde{x}) = \{x \in [0,1]^n \mid x_j = \tilde{x}_j \ j = 1,\dots,p,\ C_k \leq x_{p+k} \leq C_k + 2^{-L_\gamma} \ k = 1,\dots n-p\}.$$

We clearly see that x^i $i = 0,\dots,2^{n-p}-1$ is the set of all extreme points in $P_\gamma(\tilde{x})$, and since $\tilde{x} \in P_\gamma(\tilde{x})$, the convex combination claim follows. ■

In this case, we can also provide a tighter result than the one in Corollary 4.2.3:

Corollary 6.1.4 *Let P^* be the optimal value of problem **PO**, and let $(\hat{x}(\gamma), \hat{z}(\gamma))$ be optimal for $\mathbf{GB}(\gamma)^+$. Then $c^T\hat{x}(\gamma) \leq P^*$ and $\hat{x}(\gamma)$ is ϵ-scaled feasible for* (4.1b).

Proof: Follows directly from Lemma 6.1.3. ■

Example 6.1.5 *Let us continue with Example 6.1.1 to illustrate how $\mathbf{GB}(\gamma)^+$ avoids the aforementioned issue of $\mathbf{GB}(\gamma)$. In this example,*

$$S = \left\{(x_1,x_2) \ : x_1^2 + x_2^2 \leq \frac{5}{4},\ (x_1,x_2) \in [0,1]^2\right\}.$$

We choose $\gamma = 1/4$ as before, thus $L_\gamma = 2$ and $\epsilon = 0.4375$. Instead of the feasible set defined

by (6.2), $\boldsymbol{GB}(\gamma)^+$ *defines the following one:*

$$\left(\frac{1}{2}z_{1,1}+\frac{1}{4}z_{1,2}+\frac{1}{4}z_{1,3}\right)^2+\left(\frac{1}{2}z_{2,1}+\frac{1}{4}z_{2,2}++\frac{1}{4}z_{2,3}\right)^2 \leq 2.671875 \tag{6.5a}$$

$$x_1=\frac{1}{2}z_{1,1}+\frac{1}{4}z_{1,2}+\frac{1}{4}z_{1,3} \tag{6.5b}$$

$$x_2=\frac{1}{2}z_{2,1}+\frac{1}{4}z_{2,2}+\frac{1}{4}z_{2,3} \tag{6.5c}$$

$$z_{i,j}\in\{0,1\}. \tag{6.5d}$$

As before, constraint (6.5a) *is redundant. We can discard it and see that the feasible set defined in the x-space is now*

$$S''=\left\{0,\frac{1}{4},\frac{1}{2},\frac{3}{4},1\right\}^2.$$

Clearly, $conv(S'')=[0,1]^2$ *and thus*

$$(\sqrt{5/8},\sqrt{5/8})\in conv(S'').$$

Even though these examples used inequalities that become redundant after the approximation is introduced, this might not be the case in general. This choice was made only to simplify the discussion and argue the need of Lemma 6.1.3 even in easy cases.

6.1.2 Why we need to approximate

A reader may wonder why or if "exact" feasibility (or optimality) for **PO** cannot be guaranteed. From a trivial perspective, we point out that there exist simple instances of **PO** (in fact convex, quadratically constrained problems) where all feasible solutions have irrational coordinates. Should that be the case, if any algorithm outputs an explicit numerical solution in finite time, such a solution will be infeasible.

Example 6.1.6 *A simple 1-dimensional example evidencing the need for approximation is given by the following problem:*

$$\begin{aligned} \max \quad & x \\ \textit{subject to:} \quad & x^2 \leq \frac{1}{2} \\ & x\in[0,1] \end{aligned}$$

where the optimal solution is an irrational number.

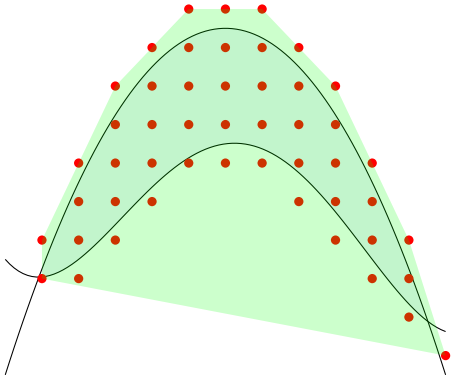


Figure 6.1: Graphical representation of Digitization and the corresponding convex hull

Example 6.1.7 *A more elaborate example is given by the following region in* $\mathbb{R}^2$*:*

$$\left\{(x,y) \ : x^2+y^2 \leq 1,\ xy \geq \frac{1}{2},\ (x,y) \in [0,1]^2\right\}.$$

The set is expressed only using rational data, and the unique feasible point is $(1/\sqrt{2}, 1/\sqrt{2})$.

Another question one might ask, is whether it is possible to achieve constraint violations that are at most ϵ, independent of the 1-norm of the constraints. Intuitively, this should not be possible since in such case one could augment the coefficients on the constraints and obtain arbitrarily high-quality solutions. For a more rigorous analysis of this see Section C.1 of the Appendix, where we argue why the *scaled* feasibility feature is needed, even in the bounded tree-width case.

6.1.3 Digitization-based Cuts

In Figure 6.1 we show a graphical representation on what $\mathbf{GB}(\gamma)^+$ is doing with respect to **PO**: it replaces each continuous variable with a weighted sum of binary variables. Moreover, by Lemma 6.1.3, the digitization is such that the convex hull contains the original feasible set. The cuts we will discuss in this section will aim to separate from such convex hull. In what follows, we assume ϵ (and thus γ) are predetermined values.

To make the analysis precise, we label the different feasible sets we are considering the

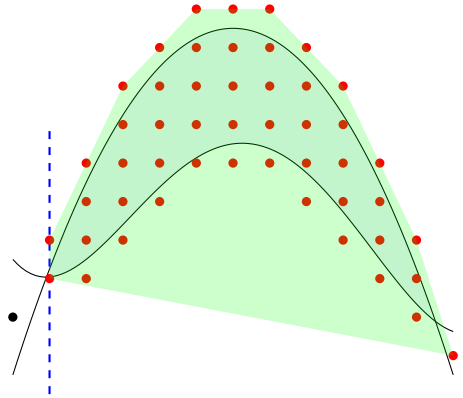


Figure 6.2: Graphical representation of Digitization Cut

following way:

$$F = \{x \in \{0,1\}^p \times [0,1]^{n-p} \mid x \text{ is feasible for } \mathbf{PO}\}$$

$$F_\epsilon = \{x \in \{0,1\}^p \times [0,1]^{n-p} \mid x \text{ is } \epsilon\text{-scaled feasible for } \mathbf{PO}\}$$

$$F_\gamma = \{(x,z) \in \{0,1\}^p \times [0,1]^{n-p} \times \{0,1\}^{(n-p)L_\gamma} \mid (x,z) \text{ is feasible for } \mathbf{GB}(\gamma)^+\}$$

Lemma 6.1.8 *$conv(F) \subseteq conv(proj_x(F_\gamma)) \subseteq conv(F_\epsilon)$*

PROOF: This is direct from Lemma 6.1.3 and from the definition of $\mathbf{GB}(\gamma)^+$. ■

Our goal will be to use F_γ to produce ϵ-feasible solutions to **PO** by separating from its convex hull. We use the following algorithmic template:

1. Find a polyhedral relaxation of F and an extreme point solution $\bar{x}$.
2. If $\bar{x} \in F_\epsilon$, then STOP.
3. Otherwise, $\bar{x} \notin \text{conv}(\text{proj}_x(F_\gamma))$ (by Lemma 6.1.8). We look for a cutting plane separating $\bar{x}$ from $\text{conv}(\text{proj}_x(F_\gamma))$ and add it to the polyhedral relaxation of F. See Figure 6.2, where the black point represents $\bar{x}$.

Separating from F_γ can be computationally challenging, as it can be as hard as optimizing over $\mathbf{GB}(\gamma)^+$ in general. Thus, in order to speed up the cut-finding procedure, we digitize only a subset of the continuous variables.

Let $J \subseteq \{p+1, \dots, n\}$, we define

$$\mathcal{I}(J) = \{i \mid \text{supp}(f_i) \subseteq \{1, \dots, p\} \cup J\}$$

i.e, the indices of the constraints that involve only continuous variables contained in J and binary variables. Given $I \subseteq \{1, \dots, m\}$, we denote as $\text{Poly}(I)$ a polyhedral relaxation (potentially using lifted variables) of the set

$$\{x \in \{0,1\}^p \times [0,1]^{n-p} \mid f_i(x) \geq 0, i \in I\}.$$

Given $\eta \in \mathbb{R}^n$, we define

$$\begin{aligned}
(\text{GB}(\gamma, J, \eta)^+): \quad & \max \ \sum_{j=1}^{n} \eta_j x_j \\
\text{subject to:} \quad & x \in \text{Poly}(\mathcal{I}(J)^c) && \text{(6.6a)} \\
& \sum_{\alpha \in I(i)} f_{i,\alpha} \left[\prod_{j \in Z(i,\alpha)} x_j \prod_{j \in J} \left(\sum_{h=1}^{L_\gamma+1} d_h z_{j,h} \right)^{\alpha_j} \right] \geq -\epsilon \|f_i\|_1, \ i \in \mathcal{I}(J) && \text{(6.6b)} \\
& z_{j,h} \in \{0,1\}, \ j \in J, \ 1 \leq h \leq L_\gamma + 1 && \text{(6.6c)} \\
& x_j = \sum_{h=1}^{L_\gamma+1} d_h z_{j,h}, \ j \in J && \text{(6.6d)} \\
& x_j \in [0,1], \ j \notin J && \text{(6.6e)} \\
& x_j \in \{0,1\}, \ 1 \leq j \leq p && \text{(6.6f)}
\end{aligned}$$

In simple words, we are just digitizing the continuous variables in J instead of all of them. As for the constraints, we are only imposing constraints in $\mathcal{I}(J)$ explicitly, as these can be represented using only binary variables and thus can be reformulated using *linear* expressions. The rest of the constraints, i.e, the ones with indices in $\mathcal{I}(J)^c$ are relaxed using some polyhedron. This renders problem $\text{GB}(\gamma, J, \eta)^+$ a *mixed-integer linear problem* with $p + L_\gamma |J|$ binary variables and $n - p$ continuous variables.

Following the same notation as before, we denote as $F_\gamma(J)$ the feasible region of problem $\text{GB}(\gamma, J, \eta)^+$. Clearly

$$F_\gamma = F_\gamma(\{p+1, \dots, n\})$$

and using the same argument as in Lemma 6.1.3 we see

$$\text{conv}(F) \subseteq \text{conv}(\text{proj}_x(F_\gamma(J))) \quad \forall J \subseteq \{p+1, \dots, n\}.$$

Therefore, every valid inequality derived from $F_\gamma(J)$ will be valid for F, and if we keep J of moderate size, $\mathrm{GB}(\gamma, J, \eta)^+$ will be a MIP with a small number of binary variables. This can make finding such valid inequality a manageable task.

Let $\bar{x} \notin F_\epsilon$. In order compute a cut we solve the following problem

$$\begin{aligned} SEP_\gamma(J): \quad & \max \ \delta \\ \text{subject to:} \quad & \alpha^T(\bar{x} - x) \geq \delta \quad \forall x \in \mathrm{proj}_x(F_\gamma(J)) \\ & \|\alpha\| \leq 1 \\ & \delta \geq 0 \end{aligned}$$

with $\|\cdot\|$ some norm. We solve problem $SEP_\gamma(J)$ using a row generation approach, i.e, at any given step we count with a finite number of solutions $\{x^k\}_{k\in K} \subseteq \mathrm{proj}_x(F_\gamma(J))$ for some K finite. Then we solve the following relaxation of $SEP_\gamma(J)$:

$$\begin{aligned} SEP^{rel}_\gamma(J): \quad & \max \ \delta \\ \text{subject to:} \quad & \alpha^T(\bar{x} - x^k) \geq \delta \quad \forall x^k, k \in K \\ & \|\alpha\| \leq 1 \\ & \delta \geq 0. \end{aligned}$$

If the norm $\|\cdot\|$ is linearly representable (e.g, 1-norm, infinity norm), problem $SEP^{rel}_\gamma(J)$ is an LP. Given $(\tilde{\alpha}, \tilde{\delta})$ optimal for $SEP^{rel}_\gamma(J)$, we solve $\mathrm{GB}(\gamma, J, \tilde{\alpha})^+$. Let (x', z') be its optimal solution.

- If $\tilde{\alpha}^T\bar{x} - \tilde{\delta} < \tilde{\alpha}^T x'$, then $(\tilde{\alpha}, \tilde{\delta})$ is not feasible for $SEP_\gamma(J)$, and we add x' to K in order to separate $(\tilde{\alpha}, \tilde{\delta})$.
- If $\tilde{\alpha}^T\bar{x} - \tilde{\delta} \geq \tilde{\alpha}^T x'$, then by optimality of (x', z') the vector $(\tilde{\alpha}, \tilde{\delta})$ is feasible for $SEP_\gamma(J)$, and thus optimal.

Using this procedure we can directly look for cutting planes. The performance of it will depend heavily on the size of J, thus the choice of J should aim for both a strong relaxation, and for efficiency. Below we use a natural choice of J that can leverage these notions if the constraints are sparse.

6.1.4 Computational Experiments

In the previous section we discussed a cutting plane approach that separates from $F_\gamma(J)$, as a way of generating valid inequalities for **PO**. We implemented an iterative cutting plane method that, in each iteration, chooses a different set J to attempt to separate the current solution $\bar{x} \notin F_\epsilon$ from $F_\gamma(J)$. The following considerations were taken in the implementation:

- The algorithm is stopped if no cut is found, or if some numerical instability is detected.
- At each iteration, given an extreme point solution $\bar{x}$, we choose J to be the support of the *most violated constraint* by $\bar{x}$.
- We used $\epsilon = 10^{-12}$ for the approximation quality.
- When solving $SEP^{rel}_\gamma(J)$, the 1-norm was used.

Computations were run on a 4-core laptop computer with an Intel Core i7-5600U CPU and 8 GB of RAM. The code was written in Python 2.7.13. The linear programming solver is Gurobi 7.0.1 with default settings.

Test instances are taken from Floudas et al. [46] (available via GLOBALLib [82]). Even though all cuts can handle arbitrary polynomial terms, for implementation purposes we restricted ourselves to problems with quadratic objective and constraints. Whenever an objective function is non-linear, we use the usual epigraph reformulation that brings the non-linearities to the constraints. To evaluate the quality of the cuts obtained, we compare with [97, 98].

We choose the initial LP relaxation to be the standard RLT relaxation of QCQP, i.e, we include a new variable representing each bilinear term and add the McCormick estimators for them (see e.g. [2]). At each subsequent iteration we construct the polyhedra Poly$(\mathcal{I}(J)^c)$ using the same RLT technique.

The results are summarized in Table 6.1. Let OPT denote the optimal value of an instance, RLT the optimal value of the standard RLT relaxation, and GLB the objective value obtained after applying the cutting plane procedure. We use the following metric for determining the quality of the solutions:

$$\text{Gap Closed} = \frac{GLB - RLT}{OPT - RLT}.$$

	SBL-Gap		SBL-Time		Digitization		
Instance	Gap Closed	Time (s)	Gap Closed	Time (s)	Gap Closed	Time (s)	Cuts
Ex2_1_5	99.98%	0.173	77.09%	0.049	98.6%	2.3	8
Ex2_1_6	99.97%	0.385	99.96%	0.265	77.8%	0.12	3
Ex2_1_7	88.09%	221.363	69.74%	11.179	100%	224.54	5
Ex2_1_8	99.90%	1.692	99.64%	0.645	9.94%	49.7	100
Ex2_1_9	93.96%	0.985	93.96%	0.985	37.9%	10.8	3
Ex3_1_1	15.94%	3600	0.35%	0.186	87.76%	63.63	7
Ex5_2_2	0.22%	0.150	0.22%	0.150	100%	0.53	1
Ex5_2_4	79.31%	68.927	73.30%	1.448	94.72%	235.24	4
Ex5_3_2	7.27%	245.821	0.0%	2.665	0%	5.3	1
Ex5_4_2	27.57%	3614	0.52%	0.177	30%	92.02	11
Ex8_4_1	91.84%	3659	89.05%	3.050	0%	Num	-
Ex9_1_4	0.0%	0.057	0.0%	0.057	39.53%	5.43	36
Ex9_2_1	68.74%	0.563	68.74%	0.563	0%	No Cut	-
Ex9_2_7	61.34%	0.899	59.30%	0.468	0%	No Cut	-

Table 6.1: Digitization Cuts Experiments

SBL-Gap shows the values obtained in [97, 98] that achieve the best Gap Closed, regardless of the time spent in the cutting plane algorithm. SBL-Time shows the values obtained in the same articles, but restricted to results obtained with a running time in order of seconds.

Table 6.1 shows encouraging results. First of all, only using a few cuts we are able to close considerable gap in many instances, with moderate running times. This shows potential, as fewer cuts yield algorithms that are numerically more stable. An interesting case is given by instance *Ex8_4_1*, which presents numerical instability while generating the very first cut. In the last 2 instances, *Ex9_2_1* and *Ex9_2_7*, the algorithm was not able to find a cutting plane with our choice of J.

Overall these results show how useful this approach is. Digitization cuts are able to provide good solutions in short times, only relying in MIP solvers, and the flexibility on the choice of J opens a gate for more elaborate heuristics that can improve on these results.

6.2 Intersection Cuts

In this section we review the family of cuts introduced in [23], and show the computational aspects in detail, in order to analyze the practical performance of them in challenging **PO** instances. At the core of this family there is the concept of *S-free sets*, which we review next.

6.2.1 Review of S-free sets and Intersection cuts

Consider a generic optimization problem

$$\min \quad c^T x \tag{6.7a}$$

$$\text{subject to:} \quad x \in S \cap P, \tag{6.7b}$$

where $P \subseteq \mathbb{R}^n$ is a polyhedron and $S \subseteq \mathbb{R}^n$ is a closed set. In order to address this problem, we will follow a cutting plane approach as before, i.e, we construct a polyhedral relaxation of $S \cap P$, find and extreme point solution $\bar{x}$, and if $\bar{x} \notin S$ we aim to find $(\pi, \pi_0) \in \mathbb{R}^{n+1}$ such that

$$\pi^T x \leq \pi_0 \;\; \forall x \in S \cap P \quad \wedge \quad \pi^T \bar{x} > \pi_0.$$

A typical example is given by integer programming, where S is the integer lattice $\mathbb{Z}^n$. In our case, we will use a set S drawn from an equivalent formulation to **PO**, and then we will derive cuts from convex forbidden zones, or S-free sets, which are defined as follows:

Definition 6.2.1 *A set $C \subset \mathbb{R}^n$ is* S-free *if $int(C) \cap S = \emptyset$ and C is convex.*

Remark 6.2.2 *For any S-free set C we have $S \cap P \subseteq clconv(P \setminus int(C))$, and so any valid inequalities for $clconv(P \setminus int(C))$ are valid for $S \cap P$.*

To generate cuts, we will adopt the standard procedure of finding a simplicial cone P' containing P and apply Balas' intersection cut [9] for $P' \setminus \text{int}(C)$. For relevant literature related to this well studied family of cuts see [35, 37, 36]. Balas and Margot [11] propose strengthened intersection cuts by using a tighter relaxation of P.

Definition 6.2.3 *We say $P' \supseteq P$ is a simplicial conic relaxation of P when P' is a displaced polyhedral cone with apex $\bar{x}$ and defined by the intersection of n linearly independent halfspaces.*

A simplicial cone may be written as follows:

$$P' = \left\{ x | x = \bar{x} + \sum_{j=1}^{n} \lambda_j r^{(j)}, \lambda \geq 0 \right\}. \tag{6.8}$$

Remark 6.2.4 *Any n linearly independent constraints describing P can be used to define a simplicial conic relaxation, using the corresponding extreme point of P as the apex of P′.*

Each extreme ray is of the form $\bar{x} + \lambda_j r^{(j)}$*, where each* $r^{(j)} \in \mathbb{R}^n$ *is a direction and* $\lambda_j \in \mathbb{R}_+$.

As mentioned before, we will assume $\bar{x} \notin S$, with the goal of separating $\bar{x}$ from S. Let C be an S-free set with $\bar{x} \in \text{int}(C)$, and P' a simplicial cone relaxation with apex $\bar{x}$. Each extreme ray of P' must satisfy one of the following:

- Either it is entirely contained in C:

$$\bar{x} + \lambda_j r^{(j)} \in \text{int}(C) \quad \forall \lambda_j \geq 0$$

- Or there is an intersection point with the boundary:

$$\exists \lambda_j^* : \bar{x} + \lambda_j^* r^{(j)} \in \text{bd}(C)$$

Then the *intersection cut* is defined as follows:

Definition 6.2.5 *Let* $\bar{x} \notin S$ *be an extreme point of a polyhedral relaxation of* $S \cap P$*, C be an S-free set with* $\bar{x} \in int(C)$*, and P′ a simplicial cone relaxation of P with apex* $\bar{x}$*. The intersection cut is the halfspace whose boundary contains each intersection point (given by* λ_j^**) and that is parallel to all extreme rays of P′ fully contained in C.*

In [9] a closed-form expression for the cut $\pi^T x \leq \pi_0$ is provided. Consider P' in inequality form, $P' = \{x | Ax \leq b\}$, where A is an $n \times n$ invertible matrix and $\bar{x} = A^{-1}b$. Then the coefficients of the intersection cut defined in Definition 6.2.5 can be expressed as

$$\pi_0 = \sum_{i=1}^{n} (1/\lambda_i^*) b_i - 1, \ \pi_j = \sum_{i=1}^{n} (1/\lambda_i^*) a_{ij}. \tag{6.9}$$

Here, $1/\lambda_j^*$ is treated as zero if the step-length is infinite. Note that, without loss of generality, the rays are assumed to be aligned with the inequality form, in the sense that for

each j, $-r^{(j)}$ is the j-th column of the inverse of A. A quick calculation shows the hyperplane $\pi^T x = \pi_0$ intersects all intersection points. Let j be such that $\lambda_j^* < \infty$:

$$\begin{aligned}
\pi^T\left(\bar{x} + \lambda_j^* r^{(j)}\right) &= \pi^T \bar{x} + \lambda_j^* \pi^T r^{(j)} \\
&= \sum_{k=1}^{n} \pi_k \bar{x}_k + \lambda_j^* \sum_{k=1}^{n} \pi_k r_k^{(j)} \\
&= \sum_{k=1}^{n} \pi_k (A^{-1}b)_k + \lambda_j^* \sum_{k=1}^{n}\sum_{i=1}^{n} (1/\lambda_i^*) a_{ik} r_k^{(j)} \\
&= \sum_{i=1}^{n} (1/\lambda_i^*) \sum_{k=1}^{n} a_{ik} (A^{-1}b)_k + \lambda_j^* \sum_{i=1}^{n} (1/\lambda_i^*) \sum_{k=1}^{n} a_{ik} r_k^{(j)} \\
&= \sum_{i=1}^{n} (1/\lambda_i^*) b_i - \lambda_j^* \sum_{i=1}^{n} (1/\lambda_i^*) \mathbf{1}_{\{i=j\}} \\
&= \sum_{i=1}^{n} (1/\lambda_i^*) b_i - 1 = \pi_0
\end{aligned}$$

and clearly the halfspace is parallel to all $r^{(j)}$ such that $\lambda_j^* = \infty$.

The final step on constructing the intersection cut is to determine the right direction of the inequality. To this end, let $\beta_0 := \operatorname{sgn}(\pi^T \bar{x} - \pi_0)$, where $\operatorname{sgn}(\cdot)$ denotes the sign function. The intersection cut formula is then given by:

$$\beta_0(\pi^T x - \pi_0) \leq 0. \tag{6.10}$$

The following proposition concludes the argument for the correctness of the cut:

Proposition 6.2.6 *Let $V := \{x \mid \beta_0(\pi^T x - \pi_0) \leq 0\}$ be the halfspace defined by the intersection cut. Then*

$$V \supseteq P' \setminus \operatorname{int}(C).$$

Furthermore, if all step lengths are finite, then $V \cap P' = \operatorname{conv}(P' \setminus \operatorname{int}(C))$.

For a proof of this statement see [9, Theorem 1].

From this section we see that the bottleneck of constructing intersection cuts efficiently is given by determining the step-lengths λ^*. These will depend on the S-free set constructed for each $\bar{x}$, and in general they might not be easily computable.

In the next section we will review how we can construct S-free sets for **PO** problems, as well as showing how to compute the step-lengths for each one of the proposed S-free sets.

6.2.2 S-free sets for Polynomial Optimization

We now turn to the construction of S-free sets for **PO** problems. We begin by casting **PO** as a problem of the type (6.7) so as to identify S precisely.

6.2.2.1 Moment-based reformulation for PO problems

We will make use of the moment-based reformulation of **PO** problems (see [67, 70]). Let ρ be the maximum degree among polynomials f_i defined in (4.1b). Let $m_r(x) = [1, x_1, ..., x_n, x_1x_2, ..., x_n^2, ...x_n^r]$ be a vector of all monomials up to degree r. Then it is easy to see that whenever $2r \geq \rho$ any polynomial f_i can be written as

$$f_i(x) = m_r^T(x) A_i m_r(x)$$

where A_i is an appropriately defined symmetric matrix obtained from f_i. Then,

$$f_i(x) \geq 0 \iff m_r^T(x) A_i m_r(x) \geq 0 \iff \langle m_r(x) m_r(x)^T, A_i \rangle$$

and we can apply this transformation to **PO** to obtain the following reformulation:

$$\begin{aligned} \text{(LPO)}: \ & \min \langle A_0, X \rangle \\ \text{subject to: } & \langle A_i, X \rangle \geq 0, \quad i = 1, ..., m, && \text{(6.11a)} \\ & X = m_r(x) m_r^T(x). && \text{(6.11b)} \end{aligned}$$

Remark 6.2.7 *A problem of the form LPO can be obtained even when not assuming the objective function in* ***PO*** *is linear.*

Condition (6.11b) can be equivalently stated as $X \succeq 0$, $\text{rank}(X) = 1$, and consistency constraints among the expressions that represent the same monomial. Constraint $X \succeq 0$ is a convex constraint, thus the non-convexity of the original **PO** problem is now absorbed in the rank constraint. Dropping the latter yields the standard semidefinite relaxation of **PO** [102]. In some cases, (e.g. [78, 67, 71, 90]) it can be proven that there exists sufficiently large r to ensure an exact relaxation. However, there is a rapid increase in the size of LPO and its semidefinite relaxation with respect to r.

Note that LPO was written in a convenient way: constraints (6.11a) are polyhedral constraints, and constraints (6.11b) represent a closed set. Hence, we are left with a natural definition of S:

$$S = \{xx^T \ : \ x \in \mathbb{R}^{N(r)}\} \tag{6.12}$$

with $N(r)$ appropriately defined. Furthermore, we let P correspond to the linear constraints (6.11a) together with the consistency constraints and $X_{11} = 1$. This brings a **PO** problem to a problem of the form (6.7), and justifies the study of *outer-product-free* sets, i.e, sets of matrices that are *not* symmetric outer-products of vectors. In what follows, we will drop the dependency on r in the definition N for alleviating notation.

For $X \in \mathbb{S}^{N\times N}$ consider the vectorized matrix $\text{vec}(X) \in \mathbb{R}^{N(N+1)/2}$, with entries from the upper triangular part of the matrix. We will work in this vector space, thus all notions of the interior, convexity, and so forth are with respect to it. However, for simplicity of the discussion, we drop the explicit vectorization where obvious, as translating the results in matrix form to vector form is straight-forward.

6.2.2.2 Oracle-based S-free sets for PO

When there is an oracle that can provide the distance of any point to the set S, a simple S-free set can be constructed. Say $\bar{x} \notin S$, and let $d > 0$ be the distance from $\bar{x}$ to S, then clearly the ball of center $\bar{x}$ and radius d, denoted $\mathcal{B}_{\text{oracle}}(\bar{x}, d)$ is S-free and an intersection cut can be easily derived from it. In [23], an in-depth study of these oracle-based cuts is performed, providing theoretical guarantees when using them in an iterative fashion.

In the context of **PO**, given a matrix $\bar{X}$ that is not an *outer-product*, we would like to compute the distance of $\bar{X}$ to the nearest symmetric outer product, or equivalently the nearest positive semidefinite matrix with rank at most one. Fortunately, this distance can be obtained using a low-rank approximation scheme. Given an integer $r > 0$, we define

$$\begin{aligned} \text{(PMA)}: \quad \min_{Y} \quad & \|\bar{X} - Y\|_F \\ \text{s.t.} \quad & \text{rank}(Y) \leq r, \\ & Y \succeq 0. \end{aligned}$$

In what follows, we denote as $\{\lambda_i\}_{i=1}^n$ the eigenvalues of $\bar{X}$, in decreasing order, and $\{d_i\}_{i=1}^n$ the corresponding eigenvectors. Dax [40] proves the following:

Theorem 6.2.8 (Dax's Theorem [40]) *Let k be the number of nonnegative eigenvalues of $\bar{X}$. For $1 \leq r \leq N-1$, an optimal solution to PMA is given by $Y = \sum_{i=1}^{\min\{k,r\}} \lambda_i d_i d_i^T$.*

When $\bar{X}$ is not negative semidefinite, the solution from Dax's theorem coincides with Eckart-Young-Mirsky [83, 43] solution to PMA without the positive semidefinite constraint. With this, we can construct an outer-product-free ball:

$$\mathcal{B}_{\text{oracle}}(\bar{X}) := \begin{cases} \mathcal{B}(\bar{X}, \|\bar{X}\|_F), & \text{if } \bar{X} \text{ is NSD,} \\ \mathcal{B}(\bar{X}, \|\sum_{i=2}^n \lambda_i d_i d_i^T\|_F), & \text{otherwise.} \end{cases}$$

And directly from Dax's Theorem:

Corollary 6.2.9 *$\mathcal{B}_{\text{oracle}}(\bar{X})$ is outer-product-free.*

This construction centers the oracle ball around around $\bar{X}$, however, for LPO we can obtain a larger ball containing the original one by re-centering it. Consider a ball $\mathcal{B}(X, r)$. Let $s > 0$ and suppose Q is in the boundary of the ball. We construct the "shifted" ball $\mathcal{B}(Q + (s/r)(X - Q), s)$. This ball has radius s and its center is located on the ray that goes through X emanating from Q, and moreover, whenever $s > r$ the shifted ball contains the original ball.

This allows us to construct $\mathcal{B}_{\text{shift}}(\bar{X}, s)$, a shifted oracle ball. We will use the nearest symmetric outer product as the boundary point in our construction:

$$\mathcal{B}_{\text{shift}}(\bar{X}, s) := \begin{cases} \mathcal{B}(s\bar{X}/\|\bar{X}\|_F, s), & \text{if } \bar{X} \text{ is NSD,} \\ \mathcal{B}\left(\lambda_1 d_1 d_1^T + \frac{s}{\|\bar{X} - \lambda_1 d_1 d_1^T\|_F}(\bar{X} - \lambda_1 d_1 d_1^T),\, s\right), & \text{otherwise.} \end{cases}$$

As the reader might notice, the definition depends on the value of s. This will determine when the shifted ball is outer-product-free. In [23], the following is proved:

Proposition 6.2.10 *Suppose $\bar{X}$ is not an outer product. If $\lambda_2 \leq 0$ then $\mathcal{B}_{\text{shift}}(\bar{X}, \|\bar{X}\|_F + \epsilon)$ is outer-product-free and strictly contains $\mathcal{B}_{\text{oracle}}(\bar{X})$ for $\epsilon > 0$. If $0 < \lambda_2 < \lambda_1$, then for $\left\|\sum_{i=2}^n \lambda_i d_i d_i^T\right\|_F < s \leq \frac{\lambda_1}{\lambda_2}\left\|\sum_{i=2}^n \lambda_i d_i d_i^T\right\|_F$, $\mathcal{B}_{\text{shift}}(\bar{X}, s)$ is outer-product-free and strictly contains $\mathcal{B}_{\text{oracle}}(\bar{X})$.*

Additionally, the following Theorem allows to further improve outer-product-free sets we have discussed so far.

Theorem 6.2.11 *Let $C \subset \mathbb{S}^{N\times N}$ be a full-dimensional outer-product-free set. Then clcone(C) is outer-product-free.*

This implies that, provided s is chosen appropriately so $\mathcal{B}_{\text{shift}}(\bar{X}, s)$ is outer-product-free, the closure of the conic hull, clcone$(\mathcal{B}_{\text{shift}}(\bar{X}, s))$, is also outer-product-free.

6.2.2.3 Maximal S-free sets for PO

While in principle any S-free set can be used to obtain intersection cuts, larger S-free sets can be useful for generating deeper cuts [35]. Thus, we aim to find *maximal* S-free sets.

Definition 6.2.12 *An S-free set C is* maximal *if $V \not\supset C$ for all S-free V.*

The following results of [23] yield two families of *maximal* outer-product-free sets.

Theorem 6.2.13 *A halfspace $\langle A, X\rangle \geq 0$ is maximal outer-product-free iff A is negative semidefinite.*

Remark 6.2.14 *It is not hard to see that whenever the S-free set is of the form $\langle A, X\rangle \geq 0$, then the intersection cut associated to it is given by $\langle A, X\rangle \leq 0$.*

Theorem 6.2.13 gives an interesting interpretation to the well known *outer-approximation* of the PSD cone, which can be used to solve SDPs. Say $\bar{X} \not\succeq 0$, then there exists c such that

$$c^T \bar{X} c < 0$$

and the set

$$\{X \in \mathbb{S}^{N\times N} |\ \langle -cc^T, X\rangle \geq 0\}$$

is maximal outer-product free. By Remark 6.2.14 the intersection cut will be given by

$$\langle -cc^T, X\rangle \leq 0$$

which will cut off $\bar{X}$.

Remark 6.2.15 *We can assume the matrix A in Theorem 6.2.13 is of rank 1. If not, let $r > 1$ be its rank, $\lambda_i < 0$ $i = 1, \dots, r$ its eigenvalues and d_i $i = 1, \dots, r$ its eigenvectors. Then, all sets*

$$\{X \in \mathbb{S}^{N\times N} |\ \langle -d_i d_i^T, X\rangle \geq 0\} \quad i = 1, \dots, r$$

will be outer-product free, and they will generate the intersection cuts

$$\langle -d_i d_i^T, X\rangle \leq 0.$$

These cuts imply

$$\left\langle \sum_{i=1}^{r} \lambda_i d_i d_i^T, X \right\rangle \leq 0$$

which is the cut $\langle A, X\rangle \geq 0$ induces.

Another family of maximal outer-product-free sets is defined next.

Definition 6.2.16 *A 2×2 PSD cone is of the form $\mathcal{C}^{i,j} := \{X \in \mathbb{S}^{N\times N} | X_{[i,j]} \succeq 0\}$, for some pair $1 \leq i \neq j \leq N$.*

Remark 6.2.17 *The boundary of $\mathcal{C}^{i,j}$ is the set of $X \in \mathbb{S}^{N\times N}$ such that $X_{[i,j]}$ has rank one and is PSD.*

Theorem 6.2.18 *Every 2×2 PSD cone is maximal outer-product-free.*

In [23] it is proven that for $N = 2$ every *maximal* outer-product-free sets is either a halfspace, or the PSD cone (which corresponds to the single 2×2 PSD cone in this case). However, for $N \geq 3$ there exist other maximal outer-product-free sets.

6.2.3 Cutting plane procedure

From the previous section we can extract 4 families of intersection cuts, which are summarized in Table 6.2. Since the 2×2 cut will separate any rank two or greater PSD matrix, and the outer approximation cuts will separate NSD or indefinite matrices, then the two families of cuts can be considered complementary. $\mathcal{B}_{\text{shift}}$ provides a stronger cut than $\mathcal{B}_{\text{oracle}}$ and can be used alone or in combination with the other cuts.

Cut Name	S-free set	Separation Condition
2×2 Cut	$X_{[i,j]} \succeq 0$	$\bar{X}_{[i,j]} \succ 0$
Outer Approximation Cut	$c^T X c \leq 0$	$\bar{X}$ NSD or indefinite
Oracle Ball Cut	$\mathcal{B}_{\text{oracle}}$	$\bar{X}$ not an outer product
Strengthened Oracle Cut	clcone($\mathcal{B}_{\text{shift}}$)	$\bar{X}$ not an outer product

Table 6.2: Proposed Intersection Cuts

In order to test the efficacy of these cuts, we construct a pure cutting plane procedure that makes use of these intersection cuts. From equations (6.10) and (6.9) we observe most of the complexity of finding the cuts of a given S-free set is to find the associated step-lengths λ^*. In the case of the 4 families described above, the step-lengths can be computed using closed-form expressions. This is one of the strongest features of these families, as finding the corresponding cuts should not present a heavy computational burden. We specify how to compute these next. In all cases we assume $\bar{X}$ is matrix which is not an outer-product, and as before we denote as λ_i and d_i its eigenvalues and eigenvectors, respectively, with the eigenvalues sorted in decreasing order. We also assume we computed the rays $D^{(k)}$ $k = 1, \ldots, N$ of the simplicial cone relaxation in matrix form.

6.2.3.1 Oracle Ball step-length

The outer-product-free set in this case is given by

$$\mathcal{B}_{\text{oracle}}(\bar{X}) := \begin{cases} \mathcal{B}(\bar{X}, \|\bar{X}\|_F), & \text{if } \bar{X} \text{ is NSD,} \\ \mathcal{B}(\bar{X}, \|\sum_{i=2}^n \lambda_i d_i d_i^T\|_F), & \text{otherwise.} \end{cases}$$

Clearly the step-length in all directions will be given by the radius of the ball. Such computation involves computing either $\|\bar{X}\|_F$ or $\|\sum_{i=2}^n \lambda_i d_i d_i^T\|_F$.

6.2.3.2 Strengthened Oracle Ball step-length

The analysis to obtain the step-lengths associated with clcone($\mathcal{B}_{\text{shift}}(\bar{X}, s)$) is more complicated, and requires separating into several different cases according to Proposition 6.2.10. We summarize these cases next. For details we refer the reader to the full-length paper [23].

- If $\bar{X}$ is NSD, it can be proven that the conic extension of $\mathcal{B}_{\text{shift}}(\bar{X}, \|\bar{X}\|_F + \epsilon)$ is a halfspace tangent to the shifted ball, at the origin. The best possible cut is then the halfspace:

$$\langle \bar{X}/\|X\|_F, X\rangle \leq 0.$$

  Hence, we do not require a step-length computation.

- If $\bar{X}$ is not NSD, and λ_2 is non-positive, then we may use the halfspace that contains $\lambda_1 d_1 d_1^T$ on its boundary and that is perpendicular to the vector from $\bar{X}$ to $\lambda_1 d_1 d_1^T$. The best possible cut is a halfspace:

$$\langle \bar{X} - \lambda_1 d_1 d_1^T, X - \lambda_1 d_1 d_1^T\rangle \leq 0.$$

  Again, we do not require a step-length computation.

- If $\bar{X}$ is not NSD and λ_2 is positive, we use the maximum shift of Proposition 6.2.10: $s = \frac{\lambda_1}{\lambda_2}\left\|\sum_{i=2}^n \lambda_i d_i d_i^T\right\|_F$. This gives us a shifted ball with center

$$X_C := \lambda_1 d_1 d_1^T + \frac{\lambda_1}{\lambda_2}(\bar{X} - \lambda_1 d_1 d_1^T)$$

  and radius

$$q := \frac{\lambda_1}{\lambda_2}\left\|\bar{X} - \lambda_1 d_1 d_1^T\right\|_F.$$

  Given a ray $D^{(k)}$, a simple computation can distinguish whether it is fully contained on the cone or not. This is determined using the projection of the direction vector onto the axis of the cone: $\langle D^{(k)}, X_C\rangle/\|X_C\|_F$.

  If the step length is finite, using geometric arguments we can prove that the step λ_k^* required to reach the boundary of cone($\mathcal{B}(X_C, q)$) using ray $D^{(k)}$ satisfies the quadratic

equation $a\lambda_k^2 + b\lambda_k + c = 0$, where

$$\begin{aligned}
a &:= q^2\langle D^{(k)}, X_C\rangle^2/(\|X_C\|_F^2 - q^2) - \|Z_4\|_F^2,\\
b &:= 2q^2\langle \bar{X}, X_C\rangle\langle D^{(k)}, X_C\rangle/(\|X_C\|_F^2 - q^2) - 2\langle Z_3, Z_4\rangle,\\
c &:= q^2\langle \bar{X}, X_C\rangle^2/(\|X_C\|_F^2 - q^2) - \|Z_3\|_F^2,\\
Z_3 &:= \|X_C\|_F\bar{X} - \langle \bar{X}, X_C\rangle X_C/\|X_C\|_F,\\
Z_4 &:= \|X_C\|_F D^{(k)} - \langle D^{(k)}, X_C\rangle X_C/\|X_C\|_F.
\end{aligned}$$

λ_k^* is equal to the positive root of this quadratic equation.

6.2.3.3 Outer Approximation step-length

From the discussion below Theorem 6.2.13, given a matrix $\bar{X} \not\succeq 0$, a maximal outer-product-free set can be computed directly using any negative eigenvalue of $\bar{X}$ (and the corresponding eigenvector) of $\bar{X}$. This outer-product-free set will be a halfspace, and the corresponding intersection cut can be obtained directly. Computing the step-lengths in this case is again unnecessary, and the intersection cut coefficients computation rely only on the eigenvalue decomposition of $\bar{X}$.

6.2.3.4 2×2 Cut step-length

Given $\bar{X}$, in order to find a 2×2 PSD cone containing $\bar{X}$ in its interior we first cycle through all pairs $1 \leq i \neq j \leq N$ and check whether $\bar{X}_{[i,j]} \succ 0$ or not. This can be done by checking $\bar{X}_{ii}, \bar{X}_{jj} > 0$ and $\bar{X}_{ii}\bar{X}_{jj} > \bar{X}_{ij}^2$.

If some pair i, j is such that $\bar{X}_{[i,j]} \succ 0$, then $\bar{X} \in \text{int}(\mathcal{C}^{i,j})$ and the step-lengths can be found using the 2×2 principal sub-matrices $\bar{X}_{[i,j]}$ and $D^{(k)}_{[i,j]}$ corresponding to $\bar{X}$ and some extreme ray direction $D^{(k)}$, respectively.

First suppose $D^{(k)}_{[i,j]} \not\succeq 0$. Then we need to compute $\lambda_k \geq 0$ such that $\bar{X}_{[i,j]} + \lambda_k D^{(k)}_{[i,j]}$ lies on the boundary of the 2×2 cone, i.e. its minimum eigenvalue is zero. For a 2×2 symmetric matrix

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

the minimum eigenvalue is given by

$$(a + c - \sqrt{a^2 - 2ac + 4b^2 + c^2})/2.$$

The eigenvalue is zero when $ac = b^2, a + c \geq 0$. Applying this to $\bar{X}_{[i,j]} + \lambda_k D^{(k)}_{[i,j]}$ we obtain the following equation:

$$((D^{(k)}_{ij})^2 - D^{(k)}_{ii} D^{(k)}_{jj})\lambda_k^2 + (2D^{(k)}_{ij} \bar{X}_{ij} - D^{(k)}_{ii} \bar{X}_{jj} - D^{(k)}_{jj} \bar{X}_{ii})\lambda_k + \bar{X}^2_{ij} - \bar{X}_{ii}\bar{X}_{jj} = 0.$$

The desired step length is given by the greater root of this quadratic equation with respect to λ_k (the lesser root sets the maximum eigenvalue to zero).

In the case $D^{(k)}_{[i,j]} \succeq 0$, $\bar{X}_{[i,j]} + \lambda_k D^{(k)}_{[i,j]}$ is positive semidefinite for all nonnegative λ_k, which yields an intersection at infinity.

Remark 6.2.19 *If some extreme rays are fully contained in an outer-product-free set, then the intersection cut can be strengthened by using negative step lengths λ^*_j [42, 15, 14]. The strengthening procedure involves changing one step-length at a time, which rotates the intersection cut in the axis defined by the other $(n-1)$ intersection points. See [23] for details on how we can apply this procedure to our proposed families of outer-product-free sets.*

6.2.4 Computational Experiments

We present numerical experiments using a pure cutting plane algorithm implementation of the cuts described above. These experiments are meant to provide a sense on how useful the cuts are, and not to be used as a complete polynomial optimization solver.

The cutting plane procedure follows the same structure as usual: we find a polyhedral relaxation of $S \cap P$ and compute an extreme point optimal solution $\bar{x}$. If $\bar{x} \notin S \cap P$, we add intersection cuts to the relaxation separating $\bar{x}$, and repeat. In our implementation we used the following stopping conditions:

- Time limit of 600 seconds
- Objective value not improving for 10 iterations
- Maximum violation of a cut not more than 10^{-8}. Here, if $\pi^T x \leq \pi_0$ is the cut, i.e, a valid inequality such that $\pi^T x^* > \pi_0$, we define the violation as

$$(\pi^T x^* - \pi_0)/\|\pi\|_1$$

- Linear program becomes numerically unstable

The machine where the tests were made is a 20-core server with an Intel Xeon E5-2687W v33.10GHz CPU and 264 GB of RAM. Even though these tests were performed in a computer with high specifications, we confirmed that similar performance can be obtained with a laptop of the same specifications as the Digitization cuts. The code is written in Python 2.7.13 using the Anaconda 4.3 distribution. The linear programming solver is Gurobi 7.0.1 with default settings. The code and full experimental data is available at `https://github.com/g-munoz/poly_cuts`.

Test instances are taken from two sources. First, as in Digitization cuts, we include problem instances from GLOBALLib [82] that have quadratic objective and constraints. Second, we include all 99 instances of BoxQP developed by several authors [106, 31]. As before, we choose the initial LP relaxation to be the standard RLT relaxation of QCQP. Note that, unlike the Digitization cuts, Intersection cuts do not need the epigraph reformulation to bring non-linearities of the objective function to the constraints. To obtain variable bounds for some of the GLOBALLib instances we apply a simple bound tightening procedure: minimize/maximize a given variable subject to the RLT relaxation.

Results averaged over all included instances are shown in Table 6.3 for GLOBALLib and Table 6.4 for BoxQP. Cut family indicates one of 5 cut configurations: Oracle Ball cuts (OB), Strengthened Oracle cuts (SO), Outer Approximation cuts (OA), 2x2 cuts together with OA (2x2 + OA), and finally 2x2, OA and SO cuts together (SO + 2x2 + OA). As before, let OPT denote the optimal value of an instance (with objective minimized), RLT the optimal value of the standard RLT relaxation, and GLB the objective value obtained after applying the cutting plane procedure. Then the optimality gaps (per instance) are

$$\text{Initial Gap} = \frac{OPT - RLT}{|OPT| + \epsilon},$$
$$\text{End Gap} = \frac{OPT - GLB}{|OPT| + \epsilon},$$
$$\text{Gap Closed} = \frac{GLB - RLT}{OPT - RLT}.$$

where $\epsilon := 1$ is a parameter to avoid division by zero. The Initial Gap values are the same irrespective of the cut configuration, so only one entry in the corresponding column is needed on Table 6.3 and Table 6.4. Iters is the number of cutting plane algorithm iterations per problem instance. Time is the total time in seconds spent by the algorithm. LPTime is the percentage of total time spent solving linear programs.

Large initial gaps are observed due to our choice of a simple initial relaxation. It is also important to mention that initial and end gaps averages are negatively affected by a few instances with high gap, thus we also provide a distribution of outcomes in Table 6.6. Table 6.5 shows a distribution on the closed gaps, which we believe is a more appropriate measure for comparing averages, as it is normalized to between 0% and 100%.

In the oracle-based cuts (OB and SO), we first observe what was expected: the strengthened oracle cuts produce better results than the basic oracle ball cuts. However, both families exhibit tailing off behavior, which results in modest closed gaps. The $2\times2+$ OA configuration provides substantially better performance compared to OB, SO, and OA, on GLOBALLib instances, indicating the benefits of separating over the 2×2 family of cuts. OA produces the best results in terms of average end gap and closed gap on BoxQP instances.

The small LPTime values indicate a relatively high percent of time spent in producing cuts. This was counter-intuitive, as generating the cuts do not require complex computations. A closer inspection showed that the majority of the time was spent in linear algebra operations (eigenvalue computations, norms, solving linear systems, etc). For these operations we rely on the NumPy library, thus our simple Python implementation is bound to be as efficient as Python and NumPy, which are mainly built for scripting and not for high performance computing. We strongly believe using a lower-level language should increase efficiency drastically.

Per-instance data for 2×2+OA can be found in Appendix D; for other configurations we refer the reader to the online supplement mentioned at the beginning of this section. The appendix also provides data showing that 2×2+OA offers comparable bounds at substantially faster computation times compared to the V2 configuration of Saxena, Bonami, and Lee [97]. However, we emphasize that cuts are complementary and not competitive.

Cut Family	Initial Gap	End Gap	Closed Gap	# Cuts	Iters	Time (s)	LPTime (%)
OB	1387.92%	1387.85%	1.00%	16.48	17.20	2.59	2.06%
SO		1387.83%	8.77%	18.56	19.52	4.14	2.29%
OA		1001.81%	8.61%	353.40	83.76	33.25	7.51%
2x2 + OA		1003.33%	32.61%	284.98	118.08	30.40	15.03%
SO+2x2+OA		1069.59%	31.91%	174.79	107.16	29.55	12.56%

Table 6.3: Averages for GLOBALLib instances

Cut Family	Initial Gap	End Gap	Closed Gap	# Cuts	Iters	Time (s)	LPTime (%)
OB	103.59%	103.56%	0.04%	12.84	13.62	127.15	0.40%
SO		103.33%	0.34%	14.34	15.45	132.07	0.49%
OA		30.88%	75.55%	676.90	137.52	459.28	31.80%
2x2 + OA		32.84%	74.52%	349.21	140.40	473.18	28.76%
SO+2x2+OA		33.43%	74.03%	227.39	136.93	475.38	26.59%

Table 6.4: Averages for BoxQP instances

	GLOBALLib					**BoxQP**				
Closed Gap	OB	SO	OA	2x2 + OA	SO+2x2 +OA	OB	SO	OA	2x2 + OA	SO+2x2 +OA
>98%	0	1	0	3	3	0	0	37	37	37
90-98 %	0	1	1	2	1	0	0	16	15	15
75-90 %	0	0	0	0	0	0	0	14	11	10
50-75 %	0	0	1	3	4	0	0	13	15	15
25-50 %	0	0	1	4	4	0	0	5	7	8
<25 %	25	23	22	13	13	99	99	14	14	14

Table 6.5: Distribution of Closed Gaps for GLOBALLib and BoxQP

	GLOBALLib						**BoxQP**					
	Initial				2x2	SO+2x2	Initial				2x2	SO+2x2
End Gap	Gap	OB	SO	OA	+ OA	+OA	Gap	OB	SO	OA	+ OA	+OA
<1 %	2	2	2	2	4	4	0	0	0	35	34	34
1 - 25 %	4	4	4	4	4	4	4	4	4	32	32	31
25-50 %	6	6	6	6	4	4	14	14	17	11	9	9
50-75 %	3	3	3	3	3	3	19	19	16	11	8	8
75-100 %	3	3	3	4	5	5	18	18	18	2	8	9
100-500 %	3	3	3	2	2	2	44	44	44	8	8	8
>500%	4	4	4	4	3	3	0	0	0	0	0	0

Table 6.6: Distribution of End Gaps for GLOBALLib and BoxQP

Chapter 7

Conclusions

The development of new polynomial optimization techniques is a cornerstone in the expansion of the current reach computational optimization has. A vast number of articles have showed a significant shift towards handling non-linear and non-convex expressions not only from a theoretical standpoint, but also from the need of solving important real-word application such as the AC-OPF problem. The gap between theory and computations related to polynomial optimization problems seems to be decreasing, and we expect this work can help in this regard by boosting a synergy between mixed-integer programming and polynomial optimization.

We showed how useful this approach can be in the AC-OPF and ACTS problems. These are difficult nonlinear, non-convex optimization problems with an important impact in our daily lives. By introducing mixed-integer programming techniques that yield simple relaxations, we are able to address problems quickly and effectively. Even though there is a loss in expressiveness by only limiting ourselves to LP formulations, there is a large gain in terms on numerical reliability and speed, and furthermore, these methodologies can also be of use alongside other frameworks such as SDP or SOCP.

We expect this approach can also be of use in Power Systems problems that involve renewable energy sources. One can envision models that merge AC-OPF models with elaborate uncertainty models arising from the presence of alternative energy sources, however, the current computational boundaries in optimization can hinder the development of them.

Counting with quick procedures in the classical problems can make the transition to complex models involving renewable sources more manageable.

From a more abstract perspective, our thorough study on the effect on structured sparsity on polynomial problems given by *tree-width* provided strong results regarding how far we can go using LP formulations, and even though some extra machinery is needed to bring this to practice, we provided an important contribution in terms of analyzing much can tree-width be exploited in a very general setting. There are two main questions that arise from this. First, is there another graph-theoretical structure we can grasp on in general? More specifically, if there is a family of instances whose intersection graphs have unbounded tree-width, can we expect to solve them efficiently? The work of [34] shows that in the pure binary case the answer is negative, and we strongly believe this is also the case for our approximation results in the continuous setting. The second question is related to how tight the sizes of our LP approximations are. It would be interesting to show some examples where the sizes given in our results cannot be improved.

The introduction of Network Polynomial Optimization problems, and the study of their tractability with respect to the structured sparsity of the underlying network, also presents a significant contribution. This concept encompasses many classical optimization problem, and can provide a unified framework for the study of them; our tractability result being an important example of this. This result provides a glimpse on how much can be achieved, even in a fairly generic setting.

Finally, the proposed cutting plane procedures for Polynomial Optimization we studied provide an interesting approach to quickly compute bounds of challenging problems without assuming much about their structure. These approaches are flexible enough so we can easily extend an build upon them. We expect they can be further exploited and blended into more elaborate solution methods such as branching.

In general, we were able to give an in-depth study on the reach of mixed-integer programming techniques in Polynomial Optimization. And we believe this is only a starting point. From both a theoretical and computational standpoint we see an important contribution, providing novel insights into the Polynomial Optimization field, and hopefully giving a boost into the computational aspects of it.

Bibliography

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice Hall, 1993.

[2] K M Anstreicher. Semidefinite programming versus the reformulation linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43:471–484, 2009.

[3] R. Aris, G.L. Nemhauser, and D.J. Wilde. Optimization of multistage cycle and branching systems by serial procedures. *Journal of American Institute of Chemical Engineers*, 10:913–919, 1964.

[4] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability. A survey. *BIT Numerical Mathematics*, 25:2–23, 1985.

[5] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

[6] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems on graphs embedded in k-trees. Technical Report TRITA-NA-8404, The Royal Institute of Technology, Stockholm, 1984.

[7] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

[8] Y. H. Au and L. Tunçel. A comprehensive analysis of polyhedral lift-and-project methods, December 2013. arXiv:1312.5972.

[9] Egon Balas. Intersection cuts—a new type of cutting planes for integer programming. *Operations Research*, 19(1):19–39, 1971.

[10] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Program.*, 58(3):295–324, 1993.

[11] Egon Balas and François Margot. Generalized intersection cuts and a new cut generating paradigm. *Mathematical Programming*, 137(1-2):19–35, 2013.

[12] Xiaowei Bao, Nikolaos V Sahinidis, and Mohit Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically constrained quadratic programs. *Optimization Methods & Software*, 24(4-5):485–504, 2009.

[13] Imre Bárány and Attila Pór. 0-1 polytopes with many facets. *Advances Math.*, 161:209–228, 2001.

[14] Amitabh Basu, Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Maximal lattice-free convex sets in linear subspaces. *Mathematics of Operations Research*, 35(3):704–720, 2010.

[15] Amitabh Basu, Gérard Cornuéjols, and Giacomo Zambelli. Convex sets and minimal sublinear functions. *Journal of Convex Analysis*, 18(2):427–432, 2011.

[16] A.R. Bergen and V. Vittal. *Power Systems Analysis*. Prentice Hall, 1999.

[17] M.W Bern, E.L Lawler, and A.L Wong. Linear-time computation of optimal subgraphs of decomposable graphs. *Journal of Algorithms*, 8(2):216 – 235, 1987.

[18] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.

[19] D. Bienstock and M. A. Langston. Chapter 8 algorithmic implications of the graph minor theorem. In C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 481 – 502. Elsevier, 1995.

[20] D. Bienstock and N. Özbay. Tree-width and the Sherali-Adams operator. *Discrete Optimization*, 1(1):13–21, June 2004.

[21] D. Bienstock and M. Zuckerberg. Subset algebra lift operators for 0-1 integer programming. *SIAM Journal on Optimization*, 15(1):63–95, 2005.

[22] Daniel Bienstock. Histogram models for robust portfolio optimization. *Journal of Computational Finance*, 11:1–64, 2007.

[23] Daniel Bienstock, Chen Chen, and Gonzalo Muñoz. Outer-product-free sets for polynomial optimization and oracle-based cuts. *arXiv preprint arXiv:1610.04604*, 2016.

[24] H. L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In Timo Lepistö and Arto Salomaa, editors, *Automata, Languages and Programming*, volume 317 of *Lecture Notes in Computer Science*, pages 105–118. Springer Berlin Heidelberg, 1988.

[25] Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

[26] Hans L Bodlaender, Pål Grnås Drange, Markus S Dregi, Fedor V Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A cˆkn 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.

[27] R.B. Borie, R.G. Parker, and C.A. Tovey. Automatic generation of linear algorithms on recursive graphs from a predicate calculus. *Algorithmica*, 7:555–581, 1992.

[28] S. Bose, S.H. Low, T. Teeraratkul, and B. Hassibi. Equivalent relaxations of optimal power flow. *IEEE Transactions on Automatic Control*, 60:729–742, 2014.

[29] G. Braun, S. Pokutta, and A. Roy. *Strong Reductions for Extended Formulations*, pages 350–361. Springer International Publishing, Cham, 2016.

[30] D. J. Brown, M. R. Fellows, and M. A. Langston. Polynomial-time self-reducibility: Theoretical motivations and practical results. *International Journal of Computer Mathematics*, 31:19, 1989.

[31] S. Burer. Optimizing a polyhedral-semidefinite relaxation of completely positive programs. *Mathematical Programming Computation*, 2(1):1–19, 2010.

[32] M.B. Cain, R.P. ONeill, and A. Castillo. The IV formulation and linear approximations of the ac optimal power flow problem. *FERC staff technical paper*, 2012.

[33] J. Carpentier. Contribution á l'étude du dispatching économique. *Bulletin de la Société Franćaise des Électriciens*, 3:431–447, 1962.

[34] V. Chandrasekaran, N. Srebro, and P. Harsha. Complexity of Inference in Graphical Models. In *Proc. 24th Conference on Uncertainty in Artificial Intelligence*, 2008.

[35] Michele Conforti, Gérard Cornuéjols, Aris Daniilidis, Claude Lemaréchal, and Jérôme Malick. Cut-generating functions and s-free sets. *Mathematics of Operations Research*, 40(2):276–391, 2014.

[36] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Equivalence between intersection cuts and the corner polyhedron. *Operations Research Letters*, 38(3):153–155, 2010.

[37] Gérard Cornuéjols, Laurence Wolsey, and Sercan Yıldız. Sufficiency of cut-generating functions. *Mathematical Programming*, pages 1–9, 2013.

[38] W. H. Cunningham and Jim Geelen. On integer programming and the branch-width of the constraint matrix. In Matteo Fischetti and DavidP. Williamson, editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *Lecture Notes in Computer Science*, pages 158–166. 2007.

[39] S. Dash, O. Günlük, and A. Lodi. On the mir closure of polyhedra. In *Integer Programming and Combinatorial Optimization*, pages 337–351. Springer Berlin Heidelberg, 2007.

[40] Achiya Dax. Low-rank positive approximants of symmetric matrices. *Advances in Linear Algebra & Matrix Theory*, 4(3):172–185, 2014.

[41] R. Dechter and J. Pearl. Tree clustering for constraint networks (research note). *Artif. Intell.*, 38:353–366, 1989.

[42] Santanu S Dey and Laurence A Wolsey. Constrained infinite group relaxations of MIPs. *SIAM Journal on Optimization*, 20(6):2890–2912, 2010.

[43] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[44] M. R. Fellows and M. A. Langston. Nonconstructive advances in polynomial-time complexity. *Information Processing Letters*, 26(3):157162, 1987.

[45] E.B. Fisher, R.P. O'Neill, and M.C. Ferris. Optimal transmission switching. *IEEE Transactions on Power Systems*, 23:1346–1355, 2008.

[46] Christodoulos A Floudas, Panos M Pardalos, Claire Adjiman, William R Esposito, Zeynep H Gümüs, Stephen T Harding, John L Klepeis, Clifford A Meyer, and Carl A Schweiger. *Handbook of test problems in local and global optimization*, volume 33. Springer Science & Business Media, 2013.

[47] E. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32(4):755–761, October 1985.

[48] D. R. Fulkerson and O. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965.

[49] Dimitris Gatzouras, Apostolos Giannopoulos, and Nilolaos Markoulakis. Lower bound for the maximal number of facets of a 0/1 polytope. *Discrete Comput. Geometry*, 34:331–349, 2005.

[50] Bissan Ghaddar, Juan C Vera, and Miguel F Anjos. A dynamic inequality generation scheme for polynomial programming. *Mathematical Programming*, pages 1–37, 2011.

[51] F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[52] J. Gondzio. Splitting dense columns of constraint matrix in interior point methods for large scale linear programming. *Optimization*, 24:285–297, 1992.

[53] D. Grimm, T. Netzer, and M. Schweighofer. A note on the representation of positive polynomials with structured sparsity. *Archiv der Mathematik*, 89:399–403, 2007.

[54] A. Gupte, S. Ahmed, M.S. Cheon, and S.S. Dey. Solving mixed integer bilinear problems using MIP formulations. *SIAM Journal on Optimization*, 23:721–744, 2013.

[55] K.W. Hedman, S.S. Oren, and R.P. O'Neill. A review of transmission switching and network topology optimization. 2011.

[56] M. Hewitt, G. L. Nemhauser, and M. W. P. Savelsbergh. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing*, 22(2):314–325, 2010.

[57] Hassan Hijazi, Carleton Coffrin, and Pascal Van Hentenryck. Convex quadratic relaxations for mixed-integer nonlinear programs in power systems. *Mathematical Programming Computation*, pages 1–47, 2016.

[58] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2016.

[59] R. Jabr. Exploiting sparsity in sdp relaxations of the opf problem. *IEEE Transactions on Power Systems*, 27:1138–1139, 2012.

[60] T. Koch, B. Hiller, M. Pfetsch, and L. Schewe, editors. *Evaluating Gas Network Capacities*. SIAM, Philadelphia, 2015.

[61] Burak Kocuk, Santanu S Dey, and X Andy Sun. Strong socp relaxations for the optimal power flow problem. *Operations Research*, 64(6):1177–1196, 2016.

[62] Burak Kocuk, Santanu S Dey, and Xu Andy Sun. Inexactness of sdp relaxation and valid inequalities for optimal power flow. *IEEE Transactions on Power Systems*, 31(1):642–651, 2016.

[63] M. Kojima, S. Kim, and H. Waki. Sparsity in sums of squares of polynomials. *Mathematical Programming*, 103:4562, 2005.

[64] Ulrich H. Kortenkamp, Jürgen Richter-Gebert, A. Sarangarajan, and Günter M. Ziegler. Extremal properties of 0/1-polytopes. *Discrete & Computational Geometry*, 17:439–448, 1997.

[65] J. Lasserre. Convergent SDP relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, 17(3):822–843, 2006.

[66] J. B. Lasserre. An explicit exact SDP relaxation for nonlinear 0-1 programs. In Karen Aardal and Bert Gerards, editors, *Integer Programming and Combinatorial Optimization*, volume 2081 of *Lecture Notes in Computer Science*, pages 293–303. 2001.

[67] J B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11:796–817, 2001.

[68] J.B. Lasserre, K.-C. Toh, and S. Yang. A bounded degree SOS hierarchy for polynomial optimization, January 2015. arXiv:1501.006126.

[69] M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28:470–496, 2001.

[70] M. Laurent. Sum of squares, moment matrices and optimization over polynomials. *IMA*, pages 1–147, 2010.

[71] Monique Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.

[72] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[73] J. Lavaei and S. H. Low. Zero duality gap in optimal power flow problem. *IEEE Trans. Power Systems*, 27(1):92–107, 2012.

[74] Karsten Lehmann, Alban Grastien, and Pascal Van Hentenryck. Ac-feasibility on tree networks is np-hard. *IEEE Transactions on Power Systems*, 31(1):798–801, 2016.

[75] Bi Li, Fatima Zahra Moataz, Nicolas Nisse, and Karol Suchan. Minimum size tree-decompositions. *Electronic Notes in Discrete Mathematics*, 50:21–27, 2015.

[76] M Locatelli and F Schoen. On convex envelopes for bivariate functions over polytopes. *Mathematical Programming*, pages 1–27, 2013.

[77] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM J. on Optimization*, 1:166–190, 1991.

[78] L Lovász and A Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

[79] James Luedtke, Mahdi Namazifar, and Jeff Linderoth. Some results on the strength of relaxations of multilinear functions. *Mathematical programming*, 136(2):325–351, 2012.

[80] R. Madani, M. Ashraphijuo, and J. Lavaei. OPF Solver. `http://www.ee.columbia.edu/lavaei/Software.html`, 2014.

[81] Ramtin Madani, Morteza Ashraphijuo, and Javad Lavaei. Promises of conic relaxation for contingency-constrained optimal power flow problem. *IEEE Transactions on Power Systems*, 31(2):1297–1307, 2016.

[82] A. Meeraus. *GLOBALLib*. `http:/http://www.gamsworld.org/global/globallib.htm`.

[83] Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59, 1960.

[84] Ruth Misener and Christodoulos A Floudas. Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied and Computational Mathematics*, 8(1):3–22, 2009.

[85] Ruth Misener and Christodoulos A Floudas. Global optimization of mixed-integer quadratically-constrained quadratic programs (miqcqp) through piecewise-linear and edge-concave relaxations. *Mathematical Programming*, 136(1):155–182, 2012.

[86] D.K. Molzahn and I.A. Hiskens. Sparsity-exploiting moment-based relaxations of the optimal power flow problem. *IEEE Transactions on Power Systems*, 30:3168–3180, 2014.

[87] D.K. Molzahn, J.T. Holzer, B.C. Lesieutre, and C.L. DeMarco. Implementation of a large-scale optimal power flow solver based on semidefinite programming. *IEEE Transactions on Power Systems*, 28(4):3987–3998, 2013.

[88] G.L. Nemhauser. *Introduction to Dynamic Programming*. Wiley, 1966.

[89] J. Nie. Optimality conditions and finite convergence of Lasserre's hierarchy. *Mathematical Programming*, 146:97–121, 2014.

[90] Jiawang Nie. Optimality conditions and finite convergence of lasserres hierarchy. *Mathematical programming*, 146(1-2):97–121, 2014.

[91] Narayana Prasad Padhy. Unit commitment-a bibliographical survey. *IEEE Transactions on power systems*, 19(2):1196–1205, 2004.

[92] J. Pearl. Reverend bayes on inference engines: a distributed hierarchical approach. In *in Proceedings of the National Conference on Artificial Intelligence*, pages 133–136, 1982.

[93] Dzung T. Phan. Lagrangian duality and branch-and-bound algorithms for optimal power flow. *Operations Research*, 60(2):275–285, 2012.

[94] Anatoliy D Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10(4):425–437, 1997.

[95] N. Robertson and P.D. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49 – 64, 1984.

[96] N. Robertson and P.D. Seymour. Graph minors II: Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309 – 322, 1986.

[97] A Saxena, P Bonami, and J Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. *Mathematical programming*, 124:383–411, 2010.

[98] A Saxena, P Bonami, and J Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. *Mathematical programming*, 130:359–413, 2011.

[99] H. Sherali and W. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[100] H.D. Sherali, E. Dalkiran, and L. Liberti. Reduced RLT representations for nonconvex polynomial programs. *Journal of Global Optimization*, 52, 2012.

[101] H.D. Sherali and C.H. Tuncbilek. New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Operations Research Letters*, 21:1–9, 1997.

[102] N Z Shor. Quadratic optimization problems. *Soviet Journal of Circuits and Systems Sciences*, 25:6, 1987.

[103] F Tardella. Existence and sum decomposition of vertex polyhedral convex envelopes. *Optimization Letters*, 2:363–375, 2008.

[104] M Tawarmalani, J-P P Richard, and C Xiong. Explicit convex and concave envelopes through polyhedral subdivisions. *Mathematical Programming*, pages 1–47, 2013.

[105] Mohit Tawarmalani and Nikolaos V Sahinidis. Convex extensions and envelopes of lower semi-continuous functions. *Mathematical Programming*, 93:247–263, 2002.

[106] D. Vandenbussche and G. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102(3):559–575, 2005.

[107] A. Verma. *Power Grid Security Analysis: an Optimization Approach.* dissertation, Columbia University, 2007. `http://www.columbia.edu/~dano/theses/theses.html/`.

[108] M.J. Wainwright and M.I. Jordan. Treewidth-Based conditions for exactness of the Sherali-Adams and Lasserre relaxations. Technical Report 671, University of California, September 2004.

[109] M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1:1–305, 2008.

[110] H. Waki, S. Kim, M. Kojima, and M. Muramatsu. Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17:218–242, 2006.

[111] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on Power Systems*, 26:12–19, 2011.

[112] M. Zuckerberg. *A Set Theoretic Approch to Lifting Procedures for 0,1 Integer Programming*. Ph.D. dissertation, Columbia University, 2009. `http://www.columbia.edu/~dano/theses/theses.html/`.

Appendices

Appendix A

Extra details for Chapter 2

A.1 Derivation of General Case

Here we specify the derivation of the formulas (2.42) and (2.43) for general case of a line with a transformer and/or nonzero shunts. We assume the transformer is on the k side of line $\{k, m\}$ and we drop the subscripts to simplify notation. In this case we have

$$\begin{aligned} I_{km} &= \frac{1}{\tau} y \left[\frac{1}{\tau} V_k - e^{j\sigma} V_m \right] + \frac{1}{2\tau^2} y^{sh} V_k \\ &= \frac{1}{\tau} y \left[\frac{1}{\tau} V_k - (\cos\sigma + j \sin\sigma) V_m \right] + \frac{1}{2\tau^2} (g^{sh} + j b^{sh}) V_k. \end{aligned}$$

In rectangular coordinates this can be further expanded as

$$\begin{aligned} I_{km} &= \frac{1}{\tau}(g + jb) \left[\frac{1}{\tau}(e_k + j f_k) - (\cos\sigma + j \sin\sigma)(e_m + j f_m) \right] \\ &\quad + \frac{1}{2\tau^2}(g^{sh} + j b^{sh})(e_k + j f_k) \\ &= \frac{1}{\tau}(g + jb) \left[\frac{1}{\tau}(e_k + j f_k) - e_m \cos\sigma + f_m \sin\sigma - j(e_m \sin\sigma + f_m \cos\sigma) \right] \\ &\quad + \frac{1}{2\tau^2} \left[g^{sh} e_k - b^{sh} f_k + j(b^{sh} e_k + g^{sh} f_k) \right] \\ &= \frac{1}{\tau}(g + jb) \left[\frac{1}{\tau} e_k - e_m \cos\sigma + f_m \sin\sigma + j(\frac{1}{\tau} f_k - e_m \sin\sigma - f_m \cos\sigma) \right] \\ &\quad + \frac{1}{2\tau^2} \left[g^{sh} e_k - b^{sh} f_k + j(b^{sh} e_k + g^{sh} f_k) \right]. \end{aligned}$$

From this expression we obtain:

$$\begin{aligned}\operatorname{Re} I_{km} = {} & \frac{g}{\tau}\left[\frac{e_k}{\tau} - e_m\cos\sigma + f_m\sin\sigma\right] - \frac{b}{\tau}\left[\frac{1}{\tau}f_k - e_m\sin\sigma - f_m\cos\sigma\right] \\ & + \frac{1}{2\tau^2}\left[g^{sh}e_k - b^{sh}f_k\right], \end{aligned} \tag{A.1}$$

$$\begin{aligned}\operatorname{Im} I_{km} = {} & \frac{b}{\tau}\left[\frac{e_k}{\tau} - e_m\cos\sigma + f_m\sin\sigma\right] + \frac{g}{\tau}\left[\frac{1}{\tau}f_k - e_m\sin\sigma - f_m\cos\sigma\right] \\ & + \frac{1}{2\tau^2}\left[b^{sh}e_k + g^{sh}f_k\right]. \end{aligned} \tag{A.2}$$

We can easily verify that in the "no-transformer" case, i.e. $\tau = 1$ and $\sigma = 0$, (A.1) and (A.2) match the expansion (2.30) for I_{km}, as desired. As a result,

$$\begin{aligned} P_{km} = {} & \operatorname{Re} V_k I_{km}^* \\ = {} & e_k\left\{\frac{g}{\tau}\left[\frac{e_k}{\tau} - e_m\cos\sigma + f_m\sin\sigma\right] - \frac{b}{\tau}\left[\frac{1}{\tau}f_k - e_m\sin\sigma - f_m\cos\sigma\right]\right\} \\ & + f_k\left\{\frac{b}{\tau}\left[\frac{e_k}{\tau} - e_m\cos\sigma + f_m\sin\sigma+\right] + \frac{g}{\tau}\left[\frac{1}{\tau}f_k - e_m\sin\sigma - f_m\cos\sigma\right]\right\} \\ & + \frac{g^{sh}}{2\tau^2}(e_k^2 + f_k^2). \end{aligned}$$

This can be rewritten as

$$\begin{aligned} P_{km} = {} & \frac{1}{\tau}\left[\frac{e_k}{\tau} - e_m\cos\sigma\right](g\,,\,b)\tbinom{e_k}{f_k} + \frac{1}{\tau}\left[\frac{1}{\tau}f_k - f_m\cos\sigma\right](-b\,,\,g)\tbinom{e_k}{f_k} \\ & + \frac{g^{sh}}{2\tau^2}(e_k^2 + f_k^2) \\ & + \frac{ge_kf_m + be_ke_m + bf_kf_m - gf_ke_m}{\tau}\sin\sigma. \end{aligned} \tag{A.3}$$

Note that in (A.3) the third term vanishes when there is no transformer, and the second term vanishes when there is no shunt conductance. Thus, in the no-transformer case this expression evaluates to

$$(e_k - e_m)(g\,,\,b)\tbinom{e_k}{f_k} + (f_k - f_m)(-b\,,\,g)\tbinom{e_k}{f_k} + \frac{g^{sh}}{2}(e_k^2 + f_k^2)$$

which is the same as (2.31), as desired. We can further rewrite (A.3) as

$$\begin{aligned} P_{km} = & \frac{1}{\tau}\left[\frac{e_k}{\tau} - e_m\cos\sigma + f_m\sin\sigma\right](g\,,\,b)\binom{e_k}{f_k} \\ & + \frac{1}{\tau}\left[\frac{1}{\tau}f_k - f_m\cos\sigma - e_m\sin\sigma\right](-b\,,\,g)\binom{e_k}{f_k} \\ & + \frac{g^{sh}}{2\tau^2}(e_k^2+f_k^2). \end{aligned} \tag{A.4}$$

This is equation (2.42) given above.

Next we compute an expression for P_{mk}. If there is a transformer present, the expressions are not symmetric and we first need to compute I_{mk}.

$$\begin{aligned} I_{mk} = & -\frac{1}{\tau e^{j\sigma}}yV_k + \left(y+\frac{y^{sh}}{2}\right)V_m \qquad (A.5)\\ = & -\frac{1}{\tau}(\cos\sigma - j\sin\sigma)(g+jb)(e_k+jf_k) \\ & + \left(g+g^{sh}/2+j(b+b^{sh}/2)\right)(e_m+jf_m) \\ = & -\frac{1}{\tau}(g+jb)\left[e_k\cos\sigma + f_k\sin\sigma + j(-e_k\sin\sigma + f_k\cos\sigma)\right] \\ & + (g+g^{sh}/2)e_m - (b+b^{sh}/2)f_m + j\left[(b+b^{sh}/2)e_m + (g+g^{sh}/2)f_m\right]. \qquad (A.6) \end{aligned}$$

Therefore,

$$\begin{aligned} \operatorname{Re} I_{mk} = & -\frac{g}{\tau}\left[e_k\cos\sigma + f_k\sin\sigma\right] + \frac{b}{\tau}\left[-e_k\sin\sigma + f_k\cos\sigma\right] \\ & + (g+g^{sh}/2)e_m - (b+b^{sh}/2)f_m, \quad \text{and} \qquad (A.7)\\ \operatorname{Im} I_{mk} = & -\frac{g}{\tau}\left[-e_k\sin\sigma + f_k\cos\sigma\right] - \frac{b}{\tau}\left[e_k\cos\sigma + f_k\sin\sigma\right] \\ & + (b+b^{sh}/2)e_m + (g+g^{sh}/2)f_m. \qquad (A.8) \end{aligned}$$

Finally, following the power flow equations we obtain

$$\begin{aligned} P_{mk} = & \operatorname{Re} V_m I_{mk}^* \\ = & e_m\left\{-\frac{g}{\tau}\left[e_k\cos\sigma + f_k\sin\sigma\right] + \frac{b}{\tau}\left[-e_k\sin\sigma + f_k\cos\sigma\right]\right\} \\ & + f_m\left\{-\frac{g}{\tau}\left[-e_k\sin\sigma + f_k\cos\sigma\right] - \frac{b}{\tau}\left[e_k\cos\sigma + f_k\sin\sigma\right]\right\} \\ & + (g+g^{sh}/2)(e_m^2+f_m^2) \end{aligned}$$

$$
\begin{aligned}
&= \left[e_m - \frac{1}{\tau}e_k\cos\sigma\right](g\,,\,b)\tbinom{e_m}{f_m} + \left[f_m - \frac{1}{\tau}f_k\cos\sigma\right](-b\,,\,g)\tbinom{e_m}{f_m} \\
&\quad + \frac{g^{sh}}{2}(e_m^2+f_m^2) \\
&\quad + \frac{-ge_mf_k - bf_mf_k - be_me_k + gf_me_k}{\tau}\sin\sigma \\
&= \left[e_m - \frac{1}{\tau}e_k\cos\sigma - \frac{1}{\tau}f_k\sin\sigma\right](g\,,\,b)\tbinom{e_m}{f_m} + \left[f_m - \frac{1}{\tau}f_k\cos\sigma + \frac{1}{\tau}e_k\sin\sigma\right](-b\,,\,g)\tbinom{e_m}{f_m} \\
&\quad + \frac{g^{sh}}{2}(e_m^2+f_m^2), \qquad \text{(A.9)}
\end{aligned}
$$

which is equation (2.43). We now turn to the representation of P_{km} and P_{mk} using polar coordinates.

$$
\begin{aligned}
S_{km} = V_kI_{km}^* &= V_k\left(\frac{1}{\tau^2}V_k^* - \frac{1}{\tau}e^{-j\sigma}V_m^*\right)y^* + \frac{1}{2\tau^2}(g^{sh}-jb^{sh})V_kV_k^* \qquad \text{(A.10)} \\
&= V_k\left(\frac{1}{\tau^2}V_k^* - \frac{1}{\tau}e^{-j\sigma}V_m^*\right)(g-jb) + \frac{1}{2\tau^2}(g^{sh}-jb^{sh})V_kV_k^* \\
&= |V_k|^2\frac{g}{\tau^2} - |V_k||V_m|\frac{g}{\tau}\cos(\theta_{km}-\sigma) - |V_k||V_m|\frac{b}{\tau}\sin(\theta_{km}-\sigma) + \frac{g^{sh}}{2\tau^2}|V_k|^2 \\
&\quad + j\left[-|V_k|^2\frac{b}{\tau^2} + |V_k||V_m|\frac{b}{\tau}\cos(\theta_{km}-\sigma) - |V_k||V_m|\frac{g}{\tau}\sin(\theta_{km}-\sigma) - \frac{b^{sh}}{2\tau^2}|V_k|^2\right].
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
S_{mk} &= |V_m|^2g - |V_k||V_m|\frac{g}{\tau}\cos(\theta_{mk}+\sigma) - |V_k||V_m|\frac{b}{\tau}\sin(\theta_{mk}+\sigma) + \frac{g^{sh}}{2}|V_m|^2 \\
&\quad + j\left[-|V_m|^2b + |V_k||V_m|\frac{b}{\tau}\cos(\theta_{mk}+\sigma) - |V_k||V_m|\frac{g}{\tau}\sin(\theta_{mk}+\sigma) - \frac{b^{sh}}{2}|V_m|^2\right].
\end{aligned}
$$

Hence the active power loss equals

$$
\begin{aligned}
&P_{km} + P_{mk} \\
&= \left(\frac{|V_k|^2}{\tau^2} + |V_m|^2\right)g - 2g\frac{|V_k|}{\tau}|V_m|\cos(\theta_{km}-\sigma) + \frac{g^{sh}}{2\tau^2}|V_k|^2 \;+\; \frac{g^{sh}}{2}|V_m|^2. \qquad \text{(A.11)}
\end{aligned}
$$

Relaxing this to an inequality, and using the lifted variables $V_k^{(2)}$, we obtain

$$
\begin{aligned}
P_{km} + P_{mk} &\geq \left(\frac{|V_k|^2}{\tau^2} + |V_m|^2\right)g - 2g\frac{|V_k|}{\tau}|V_m|\cos(\theta_{km}-\sigma) + \frac{g^{sh}}{2}\left(\frac{V_k^{(2)}}{\tau^2} + V_m^{(2)}\right) \\
&= \left(\frac{|V_k|^2}{\tau^2} + |V_m|^2\right)g - 2g\frac{|V_k|}{\tau}|V_m|\cos(\theta_{km})\cos(\sigma) \\
&\quad + 2g\frac{|V_k|}{\tau}|V_m|\sin(\theta_{km})\sin(\sigma) + \frac{g^{sh}}{2}\left(\frac{V_k^{(2)}}{\tau^2} + V_m^{(2)}\right)
\end{aligned}
$$

Thus, in order to prove the validity of (2.46) it suffices to show

$$\begin{aligned}&\left(e_m - \frac{1}{\tau}e_k\cos\sigma - \frac{1}{\tau}f_k\sin\sigma\right)^2 + \left(f_m - \frac{1}{\tau}f_k\cos\sigma + \frac{1}{\tau}e_k\sin\sigma\right)^2 \\ =\ &\left(\frac{|V_k|^2}{\tau^2} + |V_m|^2\right) - 2\frac{|V_k|}{\tau}|V_m|\cos(\theta_{km})\cos(\sigma) + 2\frac{|V_k|}{\tau}|V_m|\sin(\theta_{km})\sin(\sigma)\end{aligned} \tag{A.12}$$

As $e_k^2 + f_k^2 = |V_k|^2$, it can be easily seen that the term $\frac{|V_k|^2}{\tau^2} + |V_m|^2$ corresponds to the pure quadratics terms in the expansion of (A.12). Thus, we only need to argue that

$$\begin{aligned}&-\frac{|V_k|}{\tau}|V_m|\cos(\theta_{km})\cos(\sigma) + \frac{|V_k|}{\tau}|V_m|\sin(\theta_{km})\sin(\sigma) \\ =\ &-\frac{1}{\tau}e_m e_k\cos\sigma - \frac{1}{\tau}e_m f_k\sin\sigma + \frac{1}{\tau^2}e_k f_k\cos\sigma\sin\sigma \\ &-\frac{1}{\tau}f_m f_k\cos\sigma + \frac{1}{\tau}f_m e_k\sin\sigma - \frac{1}{\tau}e_k f_k\cos\sigma\sin\sigma\end{aligned}$$

Which can be easily verified to be true. This concludes the proof of (2.46).

The details of the proof mentioned in the main body are as follows: consider point k_1 in Figure 2.1. The power injection into the line, at k_1, is equal to P_{km} (i.e. it equals $\frac{V_k}{N}(N * I_{km})^* = V_k I_{km}^* = P_{km}$). Moreover by construction the voltage magnitude at k_1 equals $|V_k|/\tau$ and the phase angle difference from k_1 to m equals $\theta_{km} - \sigma$. We can now recover (A.11) from (2.20), with the last two terms account for shunts, as when deriving (2.35).

A.2 Proof of Theorem 2.4.4

Here we provide a proof of Theorem 2.4.4. For convenience of the reader we restate the Theorem and the relevant definitions:

Definition 2.4.2 *Let G be an undirected graph. A pseudo-flow is a vector P that assigns to each edge $\{k, m\}$ of G two reals, P_{km} and P_{mk}. For any node k of G we write*

$$\delta_k \doteq \textit{set of nodes of } G \textit{ adjacent to } k$$

and

$$o_k(P) \doteq \sum_{m\in\delta_k} P_{km}.$$

We call $o_k(P)$ the net output of k. We say that k is a **source** *if $o_k(P) > 0$ and is a* **sink** *if $o_k(P) < 0$. Likewise, an edge $\{k, m\}$ is termed a* **sink-edge** *(or* **source-edge***) if*

$$P_{km} + P_{mk} > 0 \quad \textit{or, respectively} \quad P_{km} + P_{mk} < 0.$$

Theorem 2.4.4 *Let G be a graph and P be a pseudo-flow on G. Then*

$$\sum_{k\,:\,o_k(P)>0} o_k(P) \;=\; \sum_{k\,:\,o_k(P)<0} (-o_k(P)) \;+\; \sum_{\{k,m\}} (P_{km} + P_{mk}).$$

Furthermore, P can be decomposed into directed flow paths, each originating at a source or source-edge and terminating at a sink or sink-edge.

PROOF: We obtain a *directed* graph D and a flow f on D as follows. First, we subdivide each edge $e = \{k, m\}$ of G by introducing a new node v_e. Moreover

- If $P_{km} \geq 0$ the edge between k and e is directed from k to e and we write $f_{ke} = P_{km}$.
- If $P_{km} < 0$ we direct the edge from e to k and we set $f_{ek} = -P_{km}$.

It follows that $f \geq 0$. Moreover, for any node v of D write

$$\delta_v^+ \;=\; \text{set of arcs of } D \text{ of the form } (v, v'),$$

$$\delta_v^- \;=\; \text{set of arcs of } D \text{ of the form } (v', v).$$

Then for any node k of G

$$o_k(P) \;=\; \sum_{a\in\delta^+(k)} f_a - \sum_{a\in\delta^-(k)} f_a$$

while for any edge $e = \{k, m\}$ of G,

$$P_{km} + P_{mk} \;=\; \sum_{a\in\delta^+(e)} f_a - \sum_{a\in\delta^-(e)} f_a.$$

Thus k is a source (sink) of P in G if and only if

$$\sum_{a\in\delta^+(k)} f_a - \sum_{a\in\delta^-(k)} f_a > 0 \quad (< 0 \text{ resp.})$$

and edge $e = \{k, m\}$ of G is a sink-edge (source-edge) for P if and only if

$$\sum_{a \in \delta^+(e)} f_a - \sum_{a \in \delta^-(e)} f_a > 0 \quad (< 0 \text{ resp.})$$

The theorem now follows from standard network flow concepts [1].

■

Appendix B

Extra details for Chapter 3

B.1 Proof of part (b) of Theorem 3.2.3

Here we describe a procedure that constructs formulation **LPz** or **LP-GB** which requires $O(2^\omega m)$ oracle queries and with additional workload $\tilde{O}(\omega n 2^\omega (m+\omega))$, as per Theorem 3.2.3 (b). Here, as per the formulation, we have a tree-decomposition (T, Q) of the intersection graph of a problem **GB**, of width ω. The critical element in the procedure is the construction of the sets $\mathcal{F}_t$ of Definition 3.5.1, and we remind the reader that for $1 \leq i \leq m$ constraint i has support K_i and the set of feasible solutions for constraint i is indicated by $S^i \subseteq \{0,1\}^{K_i}$. Note that $|K_i| \leq \omega + 1$ for all i. The procedure operates as follows:

1. For each constraint i, enumerate each partition of K_i. Given a partition (A_1, A_0) if the vector $y \in \{0,1\}^{K_i}$ defined by $y_j = k$ if $j \in A_k$ $(k = 0, 1)$ is such that $y \notin S^i$ (i.e., *not* feasible) then we record the triple (i, A_1, A_0) as a vector of length $|K_i| + 1$ (with some abuse of notation). This process requires $2^{|K_i|}$ oracle queries. This sum of all these quantities is $O(2^\omega m)$ but the more precise estimate will be needed.

2. Let $\mathcal{L}$ be the list of all vectors recorded in Step 1, sorted lexicographically; first by the index i, then by A_1 and then by A_0. After post-processing if necessary, we can assume that $\mathcal{L}$ contains no duplicates. These can be performed in time

$$O(\omega|\mathcal{L}|\log|\mathcal{L}|) = O(2^\omega m(\omega + \log m)).$$

3. For each $t \in V(T)$ construct a list of all constraints i such that $K_i \subseteq Q_t$. This can be done in time $O(\omega mn)$.

4. For each $t \in V(T)$ we form the sub-list of $\mathcal{L}$ consisting of all vectors (i, A_1, A_0) (constructed in Step 1) such that $K_i \subseteq Q_t$. Note that for any such i the total number of such vectors is at most $2^{|K_i|}$. Given a vector (i, A_1, A_0) thus enumerated, we form all vectors of the form (A'_1, A'_0) such that $A'_1 \cup A'_0 = Q_t$ and $A_k \subseteq A'_k$ for $k = 0, 1$. Let $\mathcal{L}^t$ be the list of all vectors obtained this way. Clearly, $|\mathcal{L}^t| \leq 2^{\omega} |\{i : K_i \subseteq Q_t\}|$. We lexicographically sort $\mathcal{L}^t$.

5. For each $t \in V(T)$ we enumerate all vectors $y \in \{0,1\}^{Q_t}$. For any such vector y, we have that $y \in \mathcal{F}_t$ if and only if y is *not* found in the list $\mathcal{L}^t$; and this test can be performed in time $O(\omega \log |\mathcal{L}^t|)$ after lexicographically sorting the list.

The total amount of work entailed in Step 4, using $|Q_t| \leq \omega + 1$ for each $t \in V(T)$, is

$$O\left(\sum_t \omega |\mathcal{L}^t| \log |\mathcal{L}^t|\right) = \tilde{O}\left(2^{\omega}\omega \sum_t |\{i : K_i \subseteq Q_t\}|\right).$$

Likewise, Step 5 requires $O(\omega 2^{\omega} \sum_t \log |\mathcal{L}^t|) = \tilde{O}(n\omega^2 2^{\omega})$. This completes the proof.

Appendix C

Extra details for Chapter 4

C.1 Scaled feasiblity in Theorem 4.1.2

In this section we will prove that the ϵ-scaled characteristic of Theorem 4.1.2 cannot be avoided in general.

Suppose that there is an algorithm $\mathcal{A}$ such that any **PO**, whose intersection graph has tree-width ≤ 2, can be solved in polynomial time to some given feasibility tolerance $\epsilon < 1$. That is to say, the algorithm guarantees $f_i(x) \geq -\epsilon$ for any constraint $f_i(x) \geq 0$ (as opposed to the ϵ-scaled feasibility notion). Note that since ϵ is fixed in this case, the formulation in Theorem 4.1.2 yields an algorithm that runs in polynomial time (see the result on Theorem 3.2.3 for the time it takes to build the LP formulation) but with a weaker approximation guarantee than the hypothetical algorithm $\mathcal{A}$.

We claim that the existence of algorithm $\mathcal{A}$ implies P = NP. Consider the subset-sum problem: given $n \geq 2$ positive integers $a_1, \ldots, a_n$ find $I \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in I} a_j = \sum_{j \notin I} a_j$. Denoting

$$S \doteq \frac{1}{2} \sum_{j=1}^{n} a_j \quad \text{and} \quad M \doteq 4nS,$$

the subset-sum problem can be cast as the following (pure feasibility) **PO**:

$$MSy_1 = Ma_1x_1, \tag{C.1a}$$

$$MSy_i = Ma_ix_i \; + \; MSy_{i-1}, \quad 2 \leq i \leq n, \tag{C.1b}$$

$$My_n = M, \tag{C.1c}$$

$$Mx_j(1-x_j) = 0, \quad 1 \le j \le n, \tag{C.1d}$$

$$0 \ \le \ y_j \le 1, \quad 1 \le j \le n, \tag{C.1e}$$

$$0 \ \le \ x_j \le 1, \quad 1 \le j \le n. \tag{C.1f}$$

Given a solution (x, y) to (C.1) it is clear that $x \in \{0,1\}^n$ and that $\sum_j a_j x_j = \frac{1}{2}\sum_{j=1}^n a_j$. Moreover, the intersection graph of (C.1) has tree-width 2.

By assumption, algorithm $\mathcal{A}$ will produce a solution $(\hat{x}, \hat{y})$ that violates each of the constraints (C.1a)-(C.1d) by at most ϵ and that satisfies (C.1e)-(C.1f). Then adding (C.1a)-(C.1c) yields

$$\left| \sum_{j=1}^{n} a_j \hat{x}_j - S \right| \ \le \ \frac{\epsilon n}{M}. \tag{C.2}$$

Moreover, by (C.1d) and (C.1f) for each $1 \le j \le n$,

$$\text{either} \ \ 0 \le \hat{x}_j \le \tfrac{2\epsilon}{M} \ \ \text{or} \ \ 1 - \tfrac{2\epsilon}{M} \le \hat{x}_j \le 1.$$

This follows from the fact $g(x) = x(1-x)$ is strictly increasing in $[0, 1/2)$, strictly decreasing in $(1/2, 1]$, and $g(2\epsilon/M) > \epsilon/M$. Thus, suppose we round each $\hat{x}_j$ to the nearest integer, obtaining binary values $\tilde{x}_j$ for $1 \le j \le n$. Using (C.2) we obtain

$$\left| \sum_{j=1}^{n} a_j \tilde{x}_j - S \right| \le \frac{\epsilon n}{M} + \left(\sum_{j=1}^{n} a_j \right) \frac{2\epsilon}{M} \ = \frac{\epsilon n}{M} + \frac{4S\epsilon}{M}$$

and therefore

$$\left| \sum_{j=1}^{n} a_j \tilde{x}_j \ - \ S \right| \ < \ 1.$$

Since the left hand side of the inequality must be an integer, we conclude that

$$\sum_{j=1}^{n} a_j \tilde{x}_j = S.$$

This proves that, unless $P = NP$, algorithm $\mathcal{A}$ does not exist.

C.2 LP size dependency on ϵ of Theorem 4.1.2

Now we will prove that the dependency on ϵ^{-1} of the resulting LP size cannot be improved in general. Suppose that there is an algorithm $\mathcal{A}$ that, for any $\epsilon < 1$, solves **PO** problems to

ϵ-scaled tolerance (i.e. the violation of any constraint $f_i(x) \geq 0$ is at most $\epsilon\|f\|_1$) but whose running time is polynomial, i.e. in particular it depends polynomially on $\log \epsilon^{-1}$. This is in contrast with the formulation in Theorem 4.1.2 yields an algorithm that runs time polynomial on n, m and ϵ^{-1}. Consider an unscaled version of the previous formulation of the subset-sum problem, i.e:

$$
\begin{aligned}
Sy_1 &= a_1 x_1, && \text{(C.3a)}\\
Sy_i &= a_i x_i + Sy_{i-1}, \quad 2 \leq i \leq n, && \text{(C.3b)}\\
y_n &= 1, && \text{(C.3c)}\\
x_j(1 - x_j) &= 0, \quad 1 \leq j \leq n, && \text{(C.3d)}\\
0 \leq y_j &\leq 1, \quad 1 \leq j \leq n, && \text{(C.3e)}\\
0 \leq x_j &\leq 1, \quad 1 \leq j \leq n. && \text{(C.3f)}
\end{aligned}
$$

Define $\epsilon = 1/(3SM)$ and use algorithm $\mathcal{A}$ to find a solution $(\hat{x}, \hat{y})$ that is ϵ-scaled feasible. Since the 1-norm of any polynomial in constraints (C.3) is at most $2S + 1$, we get that for each constraint $f_i(x, y) \geq 0$

$$f_i(\hat{x}, \hat{y}) \geq -\epsilon\|f_i\|_1 \geq -\epsilon(2S + 1) \geq \frac{1}{M}$$

This way we can reuse the same argument as before to obtain a solution to the subset-sum problem. Since we assume the running time depends on $\log(\epsilon^{-1})$ we get a running time that depends polynomially on $\log(nS)$ yielding the same contradiction as before.

Appendix D

Supplementary Experiments on Intersection Cuts

Table D.1 details results for GLOBALLib instances using the 2×2+OA configuration; Table D.2 shows results for BoxQP instances. OPT is the best known primal solution. RLT is the standard RLT relaxation optimal value. RLT-BT is the RLT optimal value after applying bound tightening (only applied to GLOBALLib instances). Final LB is the final lower bound obtained by the cutting plane algorithm. OA and 2×2 are the number of outer approximation and 2×2 cuts added in total, respectively. All other columns are as described in Table 6.2.4.

Table D.3 and Table D.1 compare the 2×2+OA configuration with the reported values of algorithm V2 by Saxena, Bonami, and Lee [97, 98]. For certain instances of GLOBALLib we did not obtain the same initial bound and thus excluded these from comparison. For BoxQP we start with a weak RLT relaxation (with optimal values in the wRLT column of Table D.1) to match the setup of Saxena et al. Furthermore, result for V2 were reported only for the smaller instances of BoxQP. V2 uses a RLT relaxation for QCQP problems and applies two types of cuts in standard cutting plane algorithm fashion. The first is an outer-approximation of the PSD constraint of Shor's relaxation; these are convex quadratic cuts unlike the linear cuts we are adding with OA and hence provide stronger but more expensive approximations of SDP+RLT. The second is disjunctive cuts for which the separation procedure involves a mixed-integer linear program.

On GLOBALLib instances our algorithm terminates with slightly smaller gap closed on average, with much smaller running times. In Table D.1 we run the 2×2+OA configuration with both 600 and 3600 second time limits. Due to the strong performance of RLT+SDP on these BoxQP instances, much of the difference could be attributed to differences in the outer-approximation methods used for the PSD constraint.

Instance	OPT	RLT	RLT-BT	Final LB	Initial Gap	End Gap	Gap Closed	OA	2x2	Iters	Time	LPTime
Ex2_1_1	-17.00	-18.90	-18.90	-17.89	10.6%	4.9%	53.2%	8	228	67	0.41	2.4%
Ex2_1_5	-268.01	-269.45	-269.08	-268.01	0.4%	0.0%	99.6%	2	12	4	0.14	0.0%
Ex2_1_6	-39.00	-44.40	-40.94	-39.04	5.1%	0.1%	97.8%	5	132	39	0.64	1.6%
Ex2_1_7	-4150.41	-13698.36	-5820.01	-5085.63	40.2%	22.5%	44.0%	90	127	52	7	1.0%
Ex2_1_8	15639.00	-82460.00	14439.00	15626.96	7.7%	0.1%	99.0%	222	73	59	13.82	7.2%
Ex2_1_9	-0.38	-2.20	-2.20	-1.66	131.9%	92.5%	29.9%	1400	891	474	36.9	69.8%
Ex3_1_1	7049.25	2533.20	2766.73	2788.77	60.7%	60.4%	0.5%	149	141	58	1.11	37.8%
Ex3_1_2	-30665.54	-30802.76	-30709.10	-30665.55	0.1%	0.0%	100.0%	4	8	5	0.03	0.0%
Ex3_1_3	-310.00	-310.00	-310.00	-	-	-	-	-	-	-	-	-
Ex3_1_4	-4.00	-6.00	-6.00	-5.41	40.0%	28.2%	29.5%	39	228	69	0.26	7.7%
Ex5_2_2_case1	-400.00	-599.90	-599.90	-598.97	49.9%	49.6%	0.5%	417	353	154	4.65	47.7%
Ex5_2_2_case2	-600.00	-1200.00	-1200.00	-1200.00	99.8%	99.8%	0.0%	27	28	11	0.3	3.3%
Ex5_2_2_case3	-750.00	-875.00	-874.80	-873.60	16.6%	16.5%	1.0%	188	182	74	1.67	37.7%
Ex5_2_4	-450.00	-2933.33	-2933.33	-2224.97	550.6%	393.6%	28.5%	402	744	256	3.66	15.6%
Ex5_2_5	-3500.00	-9700.00	-9700.00	-9700.00	177.1%	177.1%	0.0%	38	17	11	5.64	2.8%
Ex5_3_2	1.86	1.00	1.00	1.00	30.1%	30.1%	0.0%	38	17	11	1.97	1.5%
Ex5_3_3	3.23	1.63	1.63	1.68	37.9%	36.7%	3.1%	52	3	11	57.97	2.0%
Ex5_4_2	7512.23	2598.25	3010.67	3022.14	59.9%	59.8%	0.3%	94	81	35	0.57	22.8%
Ex8_4_1	0.62	-4.75	-3.84	-0.84	275.1%	90.0%	67.3%	5357	28	1077	600.69	63.8%
Ex9_1_4	-37.00	-63.00	-62.96	-62.85	68.3%	68.0%	0.4%	389	421	162	3.84	21.1%
Ex9_2_1	17.00	-16.00	1.00	1.00	88.9%	88.9%	0.0%	21	34	11	0.25	0.0%
Ex9_2_2	100.00	-50.00	66.67	89.79	33.0%	10.1%	69.4%	297	348	129	2.62	11.8%
Ex9_2_3	0.00	-30.00	-30.00	-30.00	3000.0%	3000.0%	0.0%	35	19	11	0.88	1.1%
Ex9_2_4	0.50	-597.00	-296.50	-296.50	19800.0%	19800.0%	0.0%	25	30	11	0.22	0.0%
Ex9_2_6	-1.00	-406.00	-201.50	-18.31	10025.2%	865.5%	91.4%	623	127	150	14.37	16.8%

Ex9_2_7	17.00	-16.00	1.00	1.00	88.9%	88.9%	0.0%	21	34	11	0.3	0.0%
Ex9_2_8	1.50	0.50	1.50	-	-	-	-	-	-	-	-	-

Table D.1: Detailed results for 2x2 + OA on GLOBALLib instances

Instance	OPT	RLT	Initial Gap	End Gap	Gap Closed	OA	2x2	Iters	Time	LPTime %
spar020-100-1	-706.50	-1066.00	50.81%	0.01%	99.97%	121	63	37	5.64	6.91%
spar020-100-2	-856.50	-1289.00	50.44%	0.10%	99.80%	360	146	102	16.87	14.82%
spar020-100-3	-772.00	-1168.50	51.29%	0.00%	100.00%	30	5	7	1.23	5.69%
spar030-060-1	-706.00	-1454.75	105.91%	1.13%	98.94%	3450	15	693	600.35	58.32%
spar030-060-2	-1377.17	-1699.50	23.39%	0.00%	99.99%	93	77	34	13.41	9.55%
spar030-060-3	-1293.50	-2047.00	58.21%	0.37%	99.36%	2074	470	521	557.87	68.14%
spar030-070-1	-654.00	-1569.00	139.69%	2.87%	97.94%	3402	18	684	601.30	60.23%
spar030-070-2	-1313.00	-1940.25	47.74%	0.00%	99.99%	160	73	47	19.61	11.32%
spar030-070-3	-1657.40	-2302.75	38.91%	0.01%	99.97%	459	203	135	57.93	20.40%
spar030-080-1	-952.73	-2107.50	121.08%	1.31%	98.92%	3274	66	668	601.38	60.68%
spar030-080-2	-1597.00	-2178.25	36.37%	0.00%	100.00%	65	20	17	7.18	6.55%
spar030-080-3	-1809.78	-2403.50	32.79%	0.00%	99.99%	71	26	20	8.42	6.18%
spar030-090-1	-1296.50	-2423.50	86.86%	0.01%	99.99%	355	137	102	42.96	20.25%
spar030-090-2	-1466.84	-2667.00	81.76%	0.01%	99.99%	335	90	89	39.40	20.76%
spar030-090-3	-1494.00	-2538.25	69.85%	0.00%	99.99%	118	74	40	16.25	11.45%
spar030-100-1	-1227.13	-2602.00	111.95%	0.01%	99.99%	756	60	165	79.32	27.84%
spar030-100-2	-1260.50	-2729.25	116.43%	0.02%	99.99%	1808	222	410	257.54	44.54%
spar030-100-3	-1511.05	-2751.75	82.05%	0.15%	99.82%	793	354	234	121.12	33.78%
spar040-030-1	-839.50	-1088.00	29.57%	0.00%	99.99%	945	77	205	270.48	45.74%
spar040-030-2	-1429.00	-1635.00	14.41%	0.00%	99.99%	693	110	165	188.59	37.53%

spar040-030-3	-1086.00	-1303.25	19.99%	0.00%	99.99%	986	114	220	305.45	46.79%
spar040-040-1	-837.00	-1606.25	91.80%	6.16%	93.29%	1669	26	339	600.73	60.30%
spar040-040-2	-1428.00	-1920.75	34.48%	0.00%	99.99%	418	112	111	115.38	32.72%
spar040-040-3	-1173.50	-2039.75	73.75%	2.44%	96.69%	1591	24	323	602.73	62.16%
spar040-050-1	-1154.50	-2146.25	85.83%	2.07%	97.59%	1659	16	335	602.55	60.50%
spar040-050-2	-1430.98	-2357.25	64.68%	0.63%	99.03%	1775	10	357	602.23	57.78%
spar040-050-3	-1653.63	-2616.00	58.16%	0.41%	99.30%	1658	7	333	600.21	61.12%
spar040-060-1	-1322.67	-2872.00	117.05%	4.77%	95.92%	1667	18	337	602.26	61.10%
spar040-060-2	-2004.23	-2917.50	45.54%	0.00%	100.00%	984	48	207	268.05	45.09%
spar040-060-3	-2454.50	-3434.00	39.89%	0.00%	100.00%	165	64	47	42.73	13.95%
spar040-070-1	-1605.00	-3144.00	95.83%	0.00%	100.00%	1353	79	288	426.26	51.88%
spar040-070-2	-1867.50	-3369.25	80.37%	0.00%	100.00%	853	60	187	232.35	41.41%
spar040-070-3	-2436.50	-3760.25	54.31%	0.00%	99.99%	1368	159	313	448.92	50.06%
spar040-080-1	-1838.50	-3846.50	109.16%	0.06%	99.94%	1649	11	332	602.62	61.57%
spar040-080-2	-1952.50	-3833.00	96.26%	0.00%	100.00%	985	85	215	277.94	44.40%
spar040-080-3	-2545.50	-4361.50	71.31%	0.03%	99.96%	1776	179	391	601.69	54.47%
spar040-090-1	-2135.50	-4376.75	104.90%	0.07%	99.93%	1737	8	349	602.56	59.09%
spar040-090-2	-2113.00	-4357.75	106.18%	0.15%	99.86%	1789	1	358	601.22	58.01%
spar040-090-3	-2535.00	-4516.75	78.14%	0.00%	100.00%	441	117	117	118.24	27.11%
spar040-100-1	-2476.38	-5009.75	102.26%	0.00%	100.00%	808	92	183	228.66	42.57%
spar040-100-2	-2102.50	-4902.75	133.12%	0.64%	99.52%	1687	3	338	600.02	60.07%

spar040-100-3	-1866.07	-5075.75	171.91%	4.35%	97.47%	1617	13	326	600.11	61.83%
spar050-030-1	-1324.50	-1858.25	40.27%	2.31%	94.27%	925	15	188	603.07	52.48%
spar050-030-2	-1668.00	-2334.00	39.90%	4.45%	88.84%	936	14	190	601.47	52.07%
spar050-030-3	-1453.61	-2107.25	44.94%	7.22%	83.94%	911	14	185	600.67	53.98%
spar050-040-1	-1411.00	-2632.00	86.47%	7.56%	91.26%	906	19	185	602.20	51.35%
spar050-040-2	-1745.76	-2923.25	67.41%	5.23%	92.24%	919	11	186	602.88	53.03%
spar050-040-3	-2094.50	-3273.50	56.26%	1.09%	98.07%	890	5	179	603.93	50.85%
spar050-050-1	-1198.41	-3536.00	194.89%	50.26%	74.21%	920	65	197	601.70	49.24%
spar050-050-2	-1776.00	-3500.50	97.05%	8.48%	91.26%	968	22	198	602.63	50.32%
spar050-050-3	-2106.10	-4119.75	95.56%	6.83%	92.86%	888	17	181	603.60	54.05%
spar060-020-1	-1212.00	-1757.25	44.95%	44.95%	0.00%	54	1	11	43.39	2.72%
spar060-020-2	-1925.50	-2238.25	16.23%	16.23%	0.00%	52	3	11	42.36	3.05%
spar060-020-3	-1483.00	-2098.75	41.49%	16.44%	60.38%	551	24	115	600.17	45.57%
spar070-025-1	-2538.91	-3832.75	50.94%	17.53%	65.58%	354	36	78	603.14	31.70%
spar070-025-2	-1888.00	-3248.00	72.00%	34.35%	52.29%	355	35	78	600.63	30.92%
spar070-025-3	-2812.28	-4167.25	48.16%	13.91%	71.12%	346	39	77	609.45	32.52%
spar070-050-1	-3252.50	-7210.75	121.66%	18.93%	84.44%	350	35	77	609.62	30.01%
spar070-050-2	-3296.00	-6620.00	100.82%	14.28%	85.84%	355	25	76	607.55	30.33%
spar070-050-3	-4306.50	-7522.00	74.65%	3.60%	95.17%	364	16	76	605.96	29.81%
spar070-075-1	-4655.50	-11647.75	150.16%	12.75%	91.51%	357	13	74	603.94	30.52%
spar070-075-2	-3865.15	-10884.75	181.57%	19.27%	89.39%	350	15	73	600.17	29.43%

spar070-075-3	-4329.40	-11262.25	160.10%	15.57%	90.27%	343	37	76	601.31	27.61%
spar080-025-1	-3157.00	-4840.75	53.32%	25.75%	51.71%	224	21	49	603.39	20.23%
spar080-025-2	-2312.34	-4378.50	89.32%	54.23%	39.28%	240	15	51	601.91	20.41%
spar080-025-3	-3090.88	-5130.25	65.96%	31.26%	52.60%	237	18	51	610.58	22.29%
spar080-050-1	-3448.10	-9783.25	183.68%	48.85%	73.41%	220	30	50	609.10	17.83%
spar080-050-2	-4449.20	-9270.00	108.33%	15.77%	85.45%	229	16	49	607.95	20.13%
spar080-050-3	-4886.00	-10029.75	105.25%	15.37%	85.39%	229	26	51	608.64	19.83%
spar080-075-1	-5896.00	-15250.75	158.64%	14.65%	90.77%	220	20	48	610.97	18.21%
spar080-075-2	-5341.00	-14246.50	166.71%	17.20%	89.68%	222	18	48	605.51	17.49%
spar080-075-3	-5980.50	-14961.50	150.15%	16.46%	89.04%	217	28	49	604.13	17.64%
spar090-025-1	-3372.50	-6171.50	82.97%	55.94%	32.58%	147	28	35	611.75	13.68%
spar090-025-2	-3500.29	-6015.00	71.82%	50.93%	29.09%	145	30	35	608.43	12.53%
spar090-025-3	-4299.00	-6698.25	55.80%	32.76%	41.29%	144	31	35	616.69	15.60%
spar090-050-1	-5152.00	-12584.00	144.23%	42.44%	70.57%	133	37	34	605.23	13.35%
spar090-050-2	-5386.50	-11920.50	121.28%	33.83%	72.11%	131	44	35	613.62	12.83%
spar090-050-3	-6151.00	-12514.00	103.43%	23.61%	77.17%	137	33	34	619.27	15.37%
spar090-075-1	-6267.45	-19054.25	203.99%	52.30%	74.36%	113	47	32	610.76	9.80%
spar090-075-2	-5647.50	-18245.50	223.03%	56.16%	74.82%	116	44	32	615.87	10.41%
spar090-075-3	-6450.00	-18929.50	193.45%	40.61%	79.01%	130	35	33	608.45	12.21%
spar100-025-1	-4027.50	-7660.75	90.19%	76.91%	14.72%	100	20	24	608.35	7.84%
spar100-025-2	-3892.56	-7338.50	88.50%	76.29%	13.79%	108	12	24	610.50	8.82%

spar100-025-3	-4453.50	-7942.25	78.32%	64.89%	17.15%	109	11	24	604.63	9.01%
spar100-050-1	-5490.00	-15415.75	180.76%	98.98%	45.25%	81	34	23	610.73	6.89%
spar100-050-2	-5866.00	-14920.50	154.33%	79.56%	48.45%	89	26	23	606.93	8.30%
spar100-050-3	-6485.00	-15564.25	139.98%	71.68%	48.79%	87	33	24	620.59	8.42%
spar100-075-1	-7384.20	-23387.50	216.69%	78.70%	63.68%	79	26	21	622.82	6.96%
spar100-075-2	-6755.50	-22440.00	232.14%	93.17%	59.86%	71	24	19	607.02	6.05%
spar100-075-3	-7554.00	-23243.50	207.67%	88.07%	57.59%	70	20	18	607.93	5.51%
spar125-025-1	-5572.00	-12251.00	119.85%	119.76%	0.07%	43	2	9	632.22	0.98%
spar125-025-2	-6156.06	-12732.00	106.80%	106.80%	0.00%	44	1	9	633.90	1.08%
spar125-025-3	-6815.50	-12650.75	85.60%	85.61%	0.00%	45	0	9	640.70	1.07%
spar125-050-1	-9308.38	-24993.00	168.48%	156.65%	7.03%	31	4	7	612.13	2.84%
spar125-050-2	-8395.00	-24810.50	195.52%	190.86%	2.38%	22	8	6	632.91	1.65%
spar125-050-3	-8343.91	-24424.50	192.70%	178.70%	7.27%	31	9	8	648.46	2.89%
spar125-075-1	-12330.00	-38202.00	209.81%	190.70%	9.11%	14	6	4	609.09	1.01%
spar125-075-2	-10382.47	-37466.75	260.84%	244.29%	6.34%	15	0	3	616.68	1.02%
spar125-075-3	-9635.50	-36202.25	275.69%	272.41%	1.19%	7	3	2	607.65	0.46%

Table D.2: Detailed results for 2x2 + OA cuts on BoxQP instances

Instance	V2 Gap	V2 Time	Gap Closed	Time
Ex2_1_1	72.62%	704.40	53.21%	0.41
Ex2_1_5	99.98%	0.17	99.68%	0.13
Ex2_1_6	99.95%	3397.65	93.87%	0.95
Ex2_1_8	84.70%	3632.28	73.23%	19.13
Ex2_1_9	98.79%	1587.94	29.87%	36.9
Ex3_1_1	15.94%	3600.27	0.34%	0.55
Ex3_1_2	99.99%	0.08	99.98%	0.04
Ex3_1_4	86.31%	21.26	29.49%	0.26
Ex5_2_2_case1	0.00%	0.02	2.05%	0.47
Ex5_2_2_case2	0.00%	0.05	0.00%	0.26
Ex5_2_2_case3	0.36%	0.36	0.00%	0.16
Ex5_2_4	79.31%	68.93	29.04%	5.69
Ex5_3_2	7.27%	245.82	0.00%	2.33
Ex5_4_2	27.57%	3614.38	0.24%	0.59
Ex9_1_4	0.00%	0.60	0.00%	0.34
Ex9_2_1	60.04%	2372.64	54.17%	28.37
Ex9_2_2	88.29%	3606.36	77.90%	30.84
Ex9_2_6	87.93%	2619.02	90.45%	0.12
Ex9_2_8	-	-	83.27%	0.12

Table D.3: Comparison with V2 on GLOBALLib instances

	OPT	wRLT	V2 Gap	V2 Time	Gap Closed	Time	Gap Closed	Time
spar020-100-1	-706.5	-1137	95.40%	3638.2	78.32%	600.45	78.34%	3600.97
spar020-100-2	-856.5	-1328.5	93.08%	3636.665	78.61%	600.68	78.61%	1083.09
spar020-100-3	-772	-1224	97.47%	3632.56	83.68%	600.52	83.76%	3600.58
spar030-060-1	-706	-1472.5	60.00%	3823.051	58.38%	600.97	58.51%	3603.02
spar030-060-2	-1377.17	-1741	91.16%	3715.979	88.55%	601.22	88.76%	3601.78
spar030-060-3	-1293.5	-2073.5	77.41%	3696.495	77.14%	600.98	77.22%	3602.11
spar030-070-1	-654	-1647	57.39%	3786.025	53.10%	600.64	53.29%	3610.13
spar030-070-2	-1313	-1989.5	86.60%	3708.212	81.52%	601.14	81.62%	3600.68
spar030-070-3	-1657.4	-2367.5	88.66%	3744.044	86.29%	601.01	86.38%	3600.67
spar030-080-1	-952.729	-2189	69.67%	3600.777	56.96%	600.4	57.04%	3602.02
spar030-080-2	-1597	-2316	86.25%	3627.132	73.92%	601.11	73.97%	3600.21
spar030-080-3	-1809.78	-2504.5	91.42%	3666.392	85.14%	601.34	85.28%	3601.93
spar030-090-1	-1296.5	-2521	81.15%	3676.815	70.02%	600.85	70.09%	3600.91
spar030-090-2	-1466.84	-2755	82.66%	3646.756	71.56%	601.03	71.62%	3600.09
spar030-090-3	-1494	-2619.5	86.37%	3701.849	75.33%	600.57	75.56%	3602.01
spar030-100-1	-1227.13	-2683.5	81.10%	3692.504	69.68%	601.17	69.78%	3602.26
spar030-100-2	-1260.5	-2870.5	72.87%	3697.329	63.60%	600.05	63.66%	3601.47
spar030-100-3	-1511.05	-2831.5	84.10%	3606.496	75.06%	600.42	75.13%	3602.89
spar040-030-1	-839.5	-1162	31.05%	3719.223	50.80%	600.67	57.21%	3605.46
spar040-030-2	-1429	-1695	27.74%	3937.898	17.58%	15.57	17.58%	15.49

spar040-030-3	-1086	-1322	28.00%	3798.683	7.73%	15.88	7.73%	16.97
spar040-040-1	-837	-1641	33.31%	3817.844	39.77%	600.53	42.76%	3602.06
spar040-040-2	-1428	-1967.5	35.19%	3968.111	56.73%	600.08	61.35%	3600.24
spar040-040-3	-1173.5	-2089	26.71%	3972.902	43.30%	600.42	47.36%	3603.8
spar040-050-1	-1154.5	-2204	36.72%	3819.72	46.05%	600.54	49.94%	3602.77
spar040-050-2	-1430.98	-2403.5	40.87%	3610.64	50.39%	601.65	53.90%	3602.41
spar040-050-3	-1653.63	-2715	33.95%	3639.977	46.20%	600.35	48.93%	3604.85
spar040-060-1	-1322.67	-2934	47.75%	3760.964	51.73%	601.64	54.33%	3604.92
spar040-060-2	-2004.23	-3011	55.79%	3707.992	68.04%	601.1	71.04%	3600.39
spar040-060-3	-2454.5	-3532	72.63%	3764.079	74.36%	601.57	76.45%	3600.44
spar040-070-1	-1605	-3194.5	64.03%	3642.681	67.37%	601.47	69.39%	3602
spar040-070-2	-1867.5	-3446.5	57.91%	3756.377	62.79%	601.51	64.77%	3605.11
spar040-070-3	-2436.5	-3833.5	62.94%	3693.666	68.40%	600.45	70.59%	3604.16
spar040-080-1	-1838.5	-3969	58.37%	3808.258	59.04%	601.59	61.09%	3600.31
spar040-080-2	-1952.5	-3902.5	66.96%	4062.433	63.85%	600.69	65.25%	3602.2
spar040-080-3	-2545.5	-4440	72.31%	4057.149	73.35%	600.42	74.94%	3601.66
spar040-090-1	-2135.5	-4490	66.64%	3781.044	66.03%	600.92	67.48%	3603.5
spar040-090-2	-2113	-4474	66.46%	3931.349	65.74%	600.33	67.12%	3603.76
spar040-090-3	-2535	-4641	73.49%	4003.706	73.80%	601.32	75.18%	3600.81
spar040-100-1	-2476.38	-5118	76.24%	3853.573	74.81%	601.45	76.31%	3605.11
spar040-100-2	-2102.5	-5043	63.89%	3658.261	64.80%	601.7	66.14%	3603.14

spar040-100-3	-1866.07	-5196.5	59.92%	3842.685	61.46%	602.01	63.58%	3600.51

Table D.4: Comparison with V2 on BoxQP instances