Mario Henrique Alves Souto Neto

Semidefinite Programming via Generalized Proximal Point Algorithm

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Elétrica.

Advisor: Prof. Álvaro de Lima Veiga Filho

Rio de Janeiro
December 2018

Pontifícia Universidade Católica do Rio de Janeiro

Mario Henrique Alves Souto Neto

Semidefinite Programming via Generalized Proximal Point Algorithm

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Engenharia Elétrica. Approved by the undersigned Examination Committee.

Rio de Janeiro, December 12th, 2018

Mario Henrique Alves Souto Neto

Mario Henrique Alves Souto Neto received in 2012 his B.Sc. degree in Industrial Engineering from the Pontifical Catholic University, Rio de Janeiro – Brazil (PUC-Rio). In 2014, he received his M.Sc. degree in Electrical Engineering also from PUC-Rio university. During his years at academia, he has been actively participating on research projects in the Laboratory of Applied Mathematical Programming and Statistics (LAMPS) at the Electrical Engineering Department of PUC-Rio.

Bibliographic data

Souto Neto, Mario Henrique Alves

Semidefinite Programming via Generalized Proximal Point Algorithm / Mario Henrique Alves Souto Neto; advisor: Álvaro de Lima Veiga Filho. – Rio de Janeiro: PUC-Rio, Departamento de Engenharia Elétrica, 2018.

v., 58 f.: il. ; 29,7 cm

1. Tese (doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica.

Inclui referências bibliográfias.

1. Engenharia Elétrica – Teses. 2. Programação Semidefinida; 3. Otimização Convexa; 4. Operadores Monotônicos; 5. Algoritmos Proximais. I. Veiga Filho, Álvaro de Lima. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

Acknowledgments

I would like to dedicate this thesis in the memory of my grandmother Alda de Lourdes Alves Souto. A very special thanks goes out to my parents, Mario José Alves Souto and Léa Maria da Silva, and my sister Jéssica Silva Lustosa, for the support they provided me.

I am grateful to my advisor Álvaro Veiga and the professors Alexandre Street and Carlos Kubrusly for their excellent technical assistance and inspiring lectures. I would like to express my gratitude to all members of LAMPS (Laboratory of Applied Mathematical Programming and Statistics) for the daily support and fruitful discussions. I would like to acknowledge the collaboration with Joaquim Garcia, which was essential to the development of this work. A special thanks to Thuener Silva for the support regarding the software development of the SDP solver.

Since all code used in this work was developed in the Julia language, I would like to thank all developers involved in this open source project. In particular, I would like to acknowledge the developers of the package JuMP. The use of the domain-specific modeling language for mathematical optimization, JuMP, was fundamental for the development of this work. A special thanks to Benoît Legat for helping with `ProxSDP`'s MOI interface.

Finally, I would also like to thank the Brazilian agency CNPq and PUC-Rio university for financial support.

Abstract

Souto, Mario; Veiga, Álvaro (Advisor). **Semidefinite Programming via Generalized Proximal Point Algorithm**. Rio de Janeiro, 2018. 58p. PhD Thesis – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Many problems of interest can be solved by means of *Semidefinite Programming* (SDP). The potential applications range from telecommunications, electrical power systems, game theory and many more fields. Additionally, the fact that SDP is a subclass of convex optimization brings a set of theoretical guarantees that makes SDP very appealing. However, among all sub-classes of convex optimization, SDP remains one of the most challenging in practice. State-of-the-art semidefinite programming solvers still do not efficiently solve large scale instances. In this regard, this thesis proposes a novel algorithm for solving SDP problems. The main contribution of this novel algorithm is to achieve a substantial speedup by exploiting the low-rank property inherent to several SDP problems. The convergence of the new methodology is proved by showing that the novel algorithm reduces to a particular case of the *Approximated Proximal Point Algorithm*. Along with the theoretical contributions, an open source numerical solver, called `ProxSDP`, is made available with this work. The performance of `ProxSDP` in comparison to state-of-the-art SDP solvers is evaluated on three case studies.

Keywords

Semidefinite Programming; Convex Optimization; Monotone Operators; Proximal Algorithms.

Resumo

Souto, Mario; Veiga, Álvaro. **Programação Semidefinida via Algoritmo de Ponto Proximal Generalizado**. Rio de Janeiro, 2018. 58p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Diversos problemas em engenharia, aprendizado de máquina e economia podem ser resolvidos através de *Programação Semidefinida* (SDP). Potenciais aplicações podem ser encontradas em telecomunicações, fluxo de potência e teoria dos jogos. Além disso, como SDP é uma subclasse de otimização convexa, temos uma série de propriedades e garantias que fazem da SDP uma tecnologia muito poderosa. Entretanto, dentre as diferentes subclasses de otimização convexa, SDP ainda permanece como uma das mais desafiadoras. Instancias de larga escala ainda não podem ser resolvidas pelos atuais softwares disponíveis. Nesse sentido, esta tese porpõe um novo algoritmo para resolver problemas de SDP. A principal contribuição deste novo algoritmo é explorar a propriedade de posto baixo presente em diversas instancias. A convergência desta nova metodologia é provada ao mostrar que o algoritmo proposto é um caso particular do *Approximate Proximal Point Algorithm*. Adicionalmente, as variáveis ótimas duais são disponibilizadas como uma consequência do algoritmo proposto. Além disso, disponibilizamos um software para resolver problemas de SDP, chamado `ProxSDP`. Três estudos de caso são utilizados para avaliar a performance do algoritmo proposto.

Palavras-chave

Programação Semidefinida; Otimização Convexa; Operadores Monotônicos; Algoritmos Proximais.

Table of contents

List of figures

List of tables

1
Introduction

Semidefinite programming (SDP) plays an important role in the field of convex optimization and subsumes several classes of optimization problems such as linear programming (LP), quadratic programming (QP) and second-order cone programming (SOCP) [1]. As a consequence, the range of applications that SDP can be applied to is wide and constantly expanding. Besides being a general framework for convex problems, SDP is also a powerful tool for building tight convex relaxations of NP-hard problems. This property has practical consequences to the approximation of a wide range of combinatorial optimization problems and potentially to all constraint satisfaction problems [2].

In practice, if one is interested in solving an SDP problem, it is crucial to have a fast, reliable and memory-efficient numerical solver. Unfortunately, in comparison to other convex optimization classes, currently available SDP solvers are not as efficient as its counterparts.

1.1
Contributions

This thesis relies mostly on the publication "*Exploiting Low-Rank Structure in Semidefinite Programming by Approximate Operator Splitting*"[3]. With that being said, the main contributions of the paper [3] are the following:

- A first-order proximal algorithm for general SDP based on the *primal-dual hybrid gradient* (PDHG) [4] is proposed. The main advantage of this methodology, in comparison to other operator splitting techniques, is that it computes the optimal dual variables along with the optimal primal solution. Additionally, the algorithm does not require solving a linear system at every iteration and it allows the presence of linear inequalities without the need of introducing additional variables to the problem;
- A modified version of the PDHG that can exploit the low-rank property of SDP is proposed. For several problems of interest, this modification makes PDHG competitive with interior-point methods, in some cases providing a speedup of an order of magnitude. For some problems, with

low-rank structure, the proposed algorithm is able to solve instances with dimensions that were still unattainable to interior-points methods in less than ten minutes, i.e. up to $5,000 \times 5,000$ sized semidefinite matrices;

- An open source SDP solver, called `ProxSDP`, is made publicly available. The goal of developing and providing this software is to both make the results of this thesis reproducible and to foster the use of semidefinite programming on different fields.

This thesis is organized to be a self contained document. In this sense, several concepts are explained in more detail than in the associated publication [3]. In particular, the connection between the proposed methodology and a generalized version of the proximal point point algorithm [5] is built from first principles.

1.2 Outline

This thesis is organized as follows. In Chapter 2, we give a brief introduction to convex analysis containing the basic concepts necessary to understanding all developments presented in this work. Chapter 3 presents the concepts of proximal operators and the proximal point meta-algorithm. Chapter 4 presents semidefinite programming and does a short historical review of the topic. These three initial Chapters provide the necessary foundation that will be used in the design of the proposed primal-dual algorithm. The main contributions of this work are presented in Chapters 5 and 6. Chapter 5 proposes a primal-dual algorithm for solving SDPs based on the proximal point algorithm framework. In Chapter 6, the primal-dual method is improved in order to exploit the low-rank structure presented in several SDP problems. The associated open source numerical SDP solver developed along this work is presented in Chapter 7. Case studies comparing the proposed solver with state-of-the-art solvers are evaluated in Chapter 8. Finally, concluding remarks and future work are discussed in Chapter 9.

2
Basic convex analysis

In this chapter we make a very brief review of some concepts and notations from convex analysis. The main goal of this chapter is to make this thesis self-contained and set the path for some results that will be established in the subsequent chapters. If the reader is looking for a more in depth exposition of convex analysis, it should refer to [6–9].

2.1
Convexity

Definition 1 *A set $\mathcal{C}$ is convex if any convex combination between two points in $\mathcal{C}$ also lies in $\mathcal{C}$. In other words, if $\theta \in [0,1]$, we have*

$$\forall\, x_1, x_2 \in \mathcal{C}, \quad \theta x_1 + (1-\theta)x_2 \in \mathcal{C}.$$

It is important to highlight that the intersection between convex sets is also a convex set [9].

Definition 2 *A set $\mathcal{K}$ is a cone if for any $\theta \geq 0$ we have that*

$$\forall\, x \in \mathcal{K}, \quad \theta x \in \mathcal{K}.$$

Definition 3 *A set $\mathcal{K}$ is a convex cone if it is both a convex set and a cone. This two properties can be translated into*

$$\forall\, x_1, x_2 \in \mathcal{K}, \theta_1, \theta_2 \geq 0, \quad \theta_1 x_1 + \theta_2 x_2 \in \mathcal{K}.$$

Definition 4 *A function $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ is called convex if it obeys Jensen's inequality*

$$\forall\, x_1, x_2 \in \mathbb{R}^n, \theta \in [0,1], \quad f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2).$$

Definition 5 *A convex optimization problem is the problem of finding a solution that optimizes a convex function under convex constraints. There are several ways of representing convex optimization problems, let's consider the generic form*

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{C}. \end{aligned} \tag{2-1}$$

where f is a convex function and $\mathcal{C}$ is a convex set.

2.2 Subgradients

Definition 6 *Given a convex function $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$, the associated subdifferential operator is defined as*

$$\partial f = \{(x, g) \in \mathbb{R}^n \times \mathbb{R}^n : \forall y \in \mathbb{R}^n,\ f(y) \geq f(x) + g^T(y - x)\}.$$

Definition 7 *The inverse of the subdifferential operator, denoted by $(\partial f)^{-1}$, is defined as*

$$(\partial f)^{-1} = \{(g, x) : (x, g) \in \partial f\}.$$

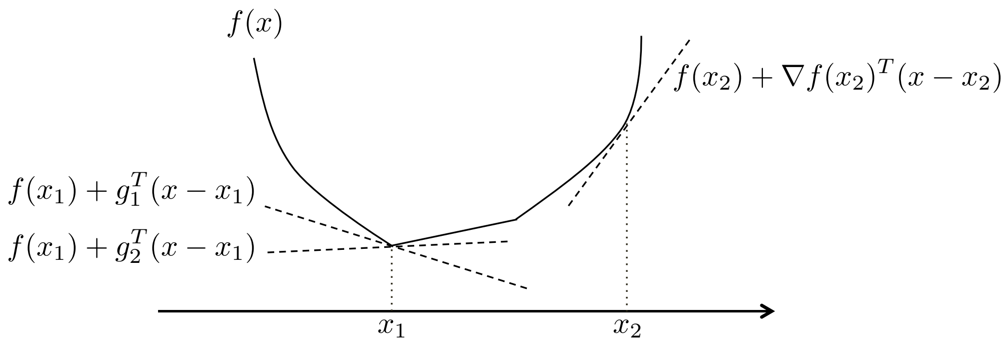
Definition 8 *The subdifferential operator evaluated at the point $x \in \mathbb{R}^n$ is a convex set, denoted by $\partial f(x)$, which is called the subdifferential of f at x. It is given by*

$$\partial f(x) = \{g \in \mathbb{R}^n : \forall\, y \in \mathbb{R}^n,\ f(y) \geq f(x) + g^T(y - x)\}.$$

Definition 9 *An element g of the subdifferential of f at x is called a subgradient of f at x, i.e. $g \in \partial f(x)$.*

Definition 10 *When the subdifferential of f at x contains only one element, i.e. $\partial f(x) = \{\nabla f(x)\}$, it is called the gradient of f. This occurs when the function f is differentiable at the point x.*

As it can be seen in Figure 2.1, the subdifferential $\partial f(x)$ has an infinite number of subgradients if the point x corresponds to a *kink* in the function, i.e. the first derivative of f is not continuous at x.



$f(x)$
$f(x_2) + \nabla f(x_2)^T(x - x_2)$
$f(x_1) + g_1^T(x - x_1)$
$f(x_1) + g_2^T(x - x_1)$
x_1
x_2

Figure 2.1: Projection of v onto the convex set $\mathcal{C}$.

2.3 Optimality conditions

In this section, we state two first-order optimality conditions for constrained convex optimization problems.

Lemma 1 (Minimum principle) *Let f be a convex function and $\mathcal{C}$ be a nonempty convex set. The point $x^* \in \mathcal{C}$ is a optimal solution of (2-1) if and only if*

$$\forall\, y \in \mathcal{C}, \quad g^T(y - x^*) \geq 0, \tag{2-2}$$

for any $g \in \partial f(x^*)$.

To give a geometric intuition consider the case where f is differentiable as in Figure 2.2. One can see that the minimum principle does provide a necessary condition for optimality and a sufficient condition under convexity. If you take the point x_1 in Figure 2.2 and try to move towards any other feasible point in $\mathcal{C}$ the value of f will increase. In contrast, the point x_2 does not satisfy the minimum principle since, for all the area of $\mathcal{C}$ shaded in red, the inequality (2-2) does not hold.

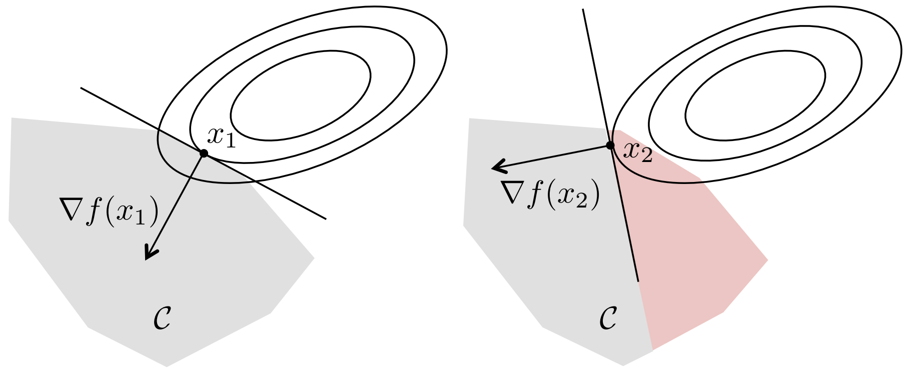


Figure 2.2: Illustration of the minimum principle optimality condition.

Proof:

See Theorem 3.4.3 in [10]. □

Lemma 2 *Let f and h be convex functions. The point $x^* \in \mathcal{C}$ is an optimal solution of* $\underset{x \in \mathcal{C}}{\text{minimize}}\Big(f(x) + h(x)\Big)$ *if and only if there exists $g_h \in \partial h(x^*)$ such that*

$$\forall\, x \in \mathcal{C}, \quad f(x) - f(x^*) + g_h^T(x - x^*) \geq 0. \tag{2-3}$$

Proof:

Let's first prove that if there is x^* such that the inequality (2-3) holds true, then x^* will be an optimal solution. From the convexity of h we have that

$$\forall\, x \in \mathcal{C}, \quad h(x) \geq h(x^*) + g_h^T(x - x^*),$$

where $g_h \in \partial h(x^*)$. As a consequence, the following holds

$$\forall\, x \in \mathcal{C}, \quad f(x) - f(x^*) + h(x) - h(x^*) \geq f(x) - f(x^*) + g_h^T(x - x^*),$$

which implies that x^* is optimal by noticing

$$\forall\, x \in \mathcal{C}, \quad (f(x) + h(x)) - (f(x^*) + h(x^*)) \geq 0,$$

Now lets show that if x^* is optimal the inequality (2-3) holds. From Lemma 1, we know that if x^* is optimal then

$$\forall\, x \in \mathcal{C}, \quad g_{f+h}^T(x - x^*) \geq 0,$$

where g_{f+h} denotes a subgradient of $(f + h)$ evaluated at the point x^*, i.e. $g_{f+h} \in \partial(f + h)(x^*)$. By the distributive quality of subgradients we have that

$$\forall\, x \in \mathcal{C}, \quad g_f^T(x - x^*) + g_h^T(x - x^*) \geq 0,$$

where $g_f \in \partial f(x^*)$ and $g_h \in \partial h(x^*)$. From the convexity of f we have that

$$\forall\, x \in \mathcal{C}, \quad f(x) \geq f(x^*) + g_f^T(x - x^*),$$

and consequently

$$\forall\, x \in \mathcal{C}, \quad f(x) - f(x^*) + g_h^T(x - x^*) \geq 0,$$

□

2.4 Convex conjugate

Definition 11 *The convex conjugate of f is given by the convex function*

$$f^*(s) = \sup_{x \,\in\, \mathrm{dom}\, f} (s^T x - f(x)) \quad \forall\, s \in \mathbb{R}^n.$$

The convex conjugate assumes the maximum difference between the original function and the linear function $s^T x$. Each pair $(s, f^*(s))$ corresponds to a tangent line of f. For instance in Figure 2.3, given a slope $s_1 \in \mathbb{R}^n$ the convex conjugate returns how far the affine function $s_1^T x$ is from a parallel supporting hyperplane of f.

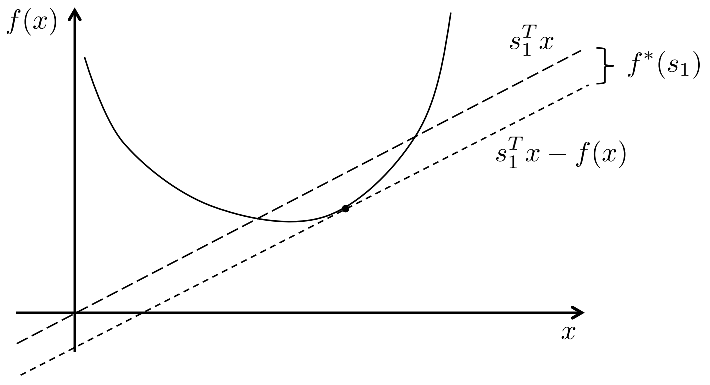


Figure 2.3: Convex conjugate of f evaluated at the point s_1.

In this work, the convex conjugate function will be used solely as a theoretical tool. In different contexts, the convex conjugate can bring insightful perspectives to convex optimization problems [9].

Lemma 3 *If f is a closed convex function, then $f^{**} = f$. In other words, the biconjugate f^{**} given by the convex conjugate of the convex conjugate is the original function f.*

Proof:

See [11] Theorem 12.2.

Lemma 4 *Let $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ be a closed convex function. The inverse of the subdifferential operator is given by the subdifferential operator of the convex conjugate of f.*

Proof:

Let the tuple (s, x) be a an element of the inverse subdifferential operator set, i.e. $(s, x) \in (\partial f)^{-1}$. By definition, we known that the tuple (x, s) will belong to the set defined by ∂f. Which implies that

$$\begin{aligned} s \in \partial f(x) &\iff 0 \in \partial f(x) - s, \\ &\iff x \in \operatorname*{argmin}_z \Big(f(z) - s^T z \Big), \\ &\iff x \in \operatorname*{argmax}_z \Big(s^T z - f(z) \Big). \end{aligned}$$

Given that x is a maximizer of $s^T z - f(z)$, by the definition of the convex conjugate we have that $f^*(s) = s^T x - f(x)$. So far we have established that

$$s \in \partial f(x) \iff f^*(s) = s^T x - f(x).$$

From lemma (3), we know that since f is convex $f^{**} = f$. Substituting above we have that

$$\begin{aligned} s \in \partial f^{**}(x) &\iff f^*(s) = s^T x - f^{**}(x), \\ &\iff (x, s) \in \partial (f^*)^{-1}, \\ &\iff (s, x) \in \partial f^*. \end{aligned}$$

□

3 Proximal point framework

In this chapter we are going to introduce the concept of proximal operators and a related algorithm for optimizing convex functions. It is important to highlight that a wide range of algorithms based on proximal operators have been proposed in the past years and the content of this chapter is only a brief review regarding some concepts that will be required in subsequent chapters. More precisely, proximal operators will be the main building blocks of the proposed algorithm for solving SDPs in chapter 5.

Methods based on proximal operators, such as the proximal gradient descent [12], are particularly interesting when the proximal operator has a known closed-form solution. Also, when the objective function is not differentiable, proximal algorithms are generally a good alternative to subgradient based methods, e.g. ISTA for linear inverse problems [13].

3.1 Proximal operator

Definition 12 *Let $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ be a convex function. The proximal operator of f is given by*

$$\mathbf{prox}_{\alpha f}(v) = \underset{x}{\operatorname{argmin}}\Big(f(x) + \tfrac{1}{2\alpha} \|x - v\|_2^2 \Big),$$

where the constant $\alpha > 0$ is a parameter that controls the trade-off between moving towards the minimizer of f and shifting in the direction of v.

The use of proximal operators is particularly interesting when a closed-form solution to the $\mathbf{prox}_{\alpha f}(v)$ problem is known. This is the case of several convex functions of interest. For instance, if $f = \|\cdot\|_1$ we have

$$\mathbf{prox}_{\alpha\|\cdot\|_1}(v) = \max\{v - \alpha, 0\} - \max\{-v - \alpha, 0\},$$

which is also know as the *soft-thresholding* associated with the ℓ_1-norm [14].

Another case of interest occurs when f is the indicator function of a convex set, i.e.

$$I_{\mathcal{C}} := \begin{cases} 0, & \text{if } x \in \mathcal{C} \\ +\infty, & \text{if } x \notin \mathcal{C}, \end{cases}$$

where $\mathcal{C}$ denotes a closed convex set. In such case we have,

$$\begin{aligned}\mathbf{prox}_{\alpha I_{\mathcal{C}}}(v) &= \underset{x}{\operatorname{argmin}}\Big\{I_{\mathcal{C}}(x) + \tfrac{1}{2\alpha}||x-v||_2^2\Big\},\\ &= \underset{x\in\mathcal{C}}{\operatorname{argmin}}\Big\{||x-v||_2^2\Big\},\\ &= \mathbf{proj}_{\mathcal{C}}(v).\end{aligned}$$

where $\mathbf{proj}_{\mathcal{C}}$ is the Euclidean projection onto the convex set $\mathcal{C}$.

Lemma 5 *The resolvent of the subdifferential operator ∂f, defined as the operator $(I+\alpha\partial f)^{-1}$, is given by the proximal operator associated with f.*

Proof:

$$\begin{aligned}z = (I+\alpha\partial f)^{-1}(v) &\iff 0 \in \partial f(z) + \tfrac{1}{\alpha}(z-v),\\ &\iff z = \underset{x}{\operatorname{argmin}}\Big\{f(x) + \tfrac{1}{2\alpha}||x-v||_2^2\Big\}.\end{aligned}$$

□

Lemma 6 *(Moreau decomposition) Let f be a convex function and f^* be it's convex conjugate. Given $\alpha > 0$, the following decomposition holds*

$$\forall\, x \in \mathbb{R}^n, \quad x = \alpha\mathbf{prox}_{f/\alpha}(x/\alpha) + \mathbf{prox}_{\alpha f^*}(x),$$

Proof: Let $u = \mathbf{prox}_{\alpha f^*}(x)$, by Lemma 5 we have that

$$\begin{aligned}0 \in \partial f^*(u) + \tfrac{1}{\alpha}(u-x) &\iff x \in (I+\alpha\partial f^*)(u),\\ &\iff \tfrac{x-u}{\alpha} \in \partial f^*(u).\end{aligned}$$

By Lemma (4) we know that $\partial f^* = (\partial f)^{-1}$ and therefore

$$\begin{aligned}0 \in \partial f(\tfrac{x-u}{\alpha}) - u &\iff x \in \partial f(\tfrac{x-u}{\alpha}) + x - u,\\ &\iff \tfrac{x}{\alpha} \in (I + \tfrac{1}{\alpha}\partial f)(\tfrac{x-u}{\alpha}),\\ &\iff \tfrac{x-u}{\alpha} \in (I + \tfrac{1}{\alpha}\partial f)^{-1}(\tfrac{x}{\alpha}),\\ &\iff x = \alpha\mathbf{prox}_{f/\alpha}(x/\alpha) + u.\end{aligned}$$

By replacing u, we have that $x = \alpha\mathbf{prox}_{f/\alpha}(x/\alpha) + \mathbf{prox}_{\alpha f^*}(x)$.

□

3.2 Generalized proximal operator

In this work we are also going to consider an extended version of the proximal operator where the Euclidean norm is replaced by a norm induced by a positive semidefinite matrix P. The P-norm will be denoted by $\|\cdot\|_P^2$, where $\|x\|_P^2 = x^T P x$. In this sense, consider the following definition and property.

Definition 13 *Let f be a convex function, P be a positive definite matrix and $\alpha > 0$. The generalized proximal operator of f with respect to P is given by*

$$\mathbf{prox}_{\alpha f}^{P}(v) = \underset{x}{\operatorname{argmin}}\Big(f(x) + \tfrac{1}{2\alpha}\|x - v\|_P^2 \Big).$$

Theorem 7 *Let f be a convex function, P be a positive definite matrix and $\mathbf{prox}_{\alpha f}^{P}$ its associated generalized proximal operator. A fixed point of $\mathbf{prox}_{\alpha f}^{P}$ is the optimal solution of f.*

Proof:

The first order optimality condition of $\mathbf{prox}_{\alpha f}^{P}(v)$ is given by

$$0 \in \partial f(x) + \tfrac{1}{\alpha}(x - v)^T P.$$

If the solution of the system above is a fixed point, i.e. $x = v$, we have that $0 \in \partial f(x)$. Therefore, a fixed point of $\mathbf{prox}_{\alpha f}^{P}$ will be the optimal solution of f. $\square$

3.3 Proximal point algorithm

The result of Lemma (7), yet very simple, suggests a very powerful strategy to optimize the convex function f. Instead of optimizing f directly, one can try to find a fixed point of its associated proximal operator. This meta-algorithm, know as *Proximal Point Algorithm* (PPA), starts with any initial iterate x^0 and successively applies the proximal operator. PPA was first proposed in the seminal work of Rockafellar [15] and quickly became a widely used framework in several fields such as image processing [16] and solving partial differential equations [17].

In the case of an unconstrained convex optimization problem, we want to find the root of the subdifferential operator, i.e. $0 \in \partial f(x)$. Given $\alpha > 0$ and starting from x_0, PPA can be viewed as successively finding a solution for

$$0 \in \partial f(x^{k+1}) + \tfrac{1}{\alpha}(x^{k+1} - x^k) \iff x^{k+1} = \underset{x}{\operatorname{argmin}}\Big(f(x) + \tfrac{1}{2\alpha}\left\|x - x^k\right\|_2^2 \Big),$$

until a fixed point, i.e. $x^{k+1} = x^k$, is found. If f is convex and has a minimum point, then the sequence $\{x^k\}$ will converge to a point belonging to the set of minimizers of f [15]. Convergence guarantees can be derived from *monotone operator theory* [8].

In this work we will refer to *Generalized Proximal Point Algorithm* (GPPA) as the iterative scheme

$$x^{k+1} \leftarrow \mathbf{prox}^P_{\alpha f}(x^k). \tag{3-1}$$

This scheme is similar to PPA but convergence will be achieved with respect to the P-norm [5]. In the remaining of this chapter, we are going to show that GPPA converges to a fixed point if one exists.

Lemma 8 *Let f be a convex function and $\mathbf{prox}^P_{\alpha f}$ its associated generalized proximal operator. If x^* is a fixed point of $\mathbf{prox}^P_{\alpha f}$, then the sequence $\{x^k\}$ generated by GPPA obeys*

$$\forall\, x^* = \mathbf{prox}^P_{\alpha f}(x^*),\quad (x^* - x^{k+1})^T P(x^{k+1} - x^k) \geq 0,$$

Proof:

Let x^{k+1} be an iteration from GPPA algorithm, i.e. $x^{k+1} = \mathbf{prox}^P_{\alpha f}(x^k)$, we have from Lemma 2 that

$$\forall\, x \in \mathbb{R}^n.,\quad f(x) - f(x^{k+1}) + \tfrac{1}{\alpha}(x - x^{k+1})^T P(x^{k+1} - x^k) \geq 0.$$

Let x^* be a fixed point of $\mathbf{prox}^P_{\alpha f}$, according to Theorem 7 we know that $f(x^*) \leq f(x)\,\forall\, x \in \mathbb{R}^n$. Therefore we have that

$$\forall\, x \in \mathbb{R}^n,\quad (x - x^{k+1})^T P(x^{k+1} - x^k) \geq 0,$$

consequently $\forall\, x^* = \mathbf{prox}^P_{\alpha f}(x^*),\quad (x^* - x^{k+1})^T P(x^{k+1} - x^k) \geq 0.$ □

Definition 14 *A sequence $\{x^k\}$ is called $\mathcal{X}$-Fejér monotone with respect to the P-norm if*

$$\forall k \in \mathbb{N}, x \in \mathcal{X},\quad \left\|x^{k+1} - x\right\|^2_P \leq \left\|x^k - x\right\|^2_P.$$

Theorem 9 *The sequence $\{x^k\}$ generated by GPPA is $\mathcal{X}^*$-Fejér monotone with respect to the P-norm, where $\mathcal{X}^*$ is the set of fixed-points of (3-1).*

Proof:

Let $x^* \in \mathcal{X}^*$,

$$\begin{aligned}\left\|x^k - x^*\right\|_P^2 &= \left\|(x^{k+1} - x^*) + (x^k - x^{k+1})\right\|_P^2, \\ &= \left\|x^{k+1} - x^*\right\|_P^2 + \left\|x^k - x^{k+1}\right\|_P^2 + 2(x^{k+1} - x^*)^T P(x^k - x^{k+1}).\end{aligned}$$

From lemma (4) we know that $(x^* - x^{k+1})^T P(x^{k+1} - x^k) \geq 0$ and therefore

$$\begin{aligned}\left\|x^{k+1} - x^*\right\|_P^2 &\leq \left\|x^k - x^*\right\|_P^2 - \left\|x^k - x^{k+1}\right\|_P^2, \\ &\leq \left\|x^k - x^*\right\|_P^2.\end{aligned}$$

□

Theorem 10 *The sequence $\{x^k\}$ generated by GPPA being $\mathcal{X}^*$-Fejér monotone implies that*

$$\mathbf{dist}(x^k, \mathcal{X}^*) = \inf_{x \in \mathcal{X}^*} \left\|x^k - x\right\|_P \to 0 \text{ as } k \to \infty,$$

monotonically.

In other words, $\mathbf{dist}(x^k, \mathcal{X}^*)$ decreases at each step of GPPA. This means that GPPA will converge to a point arbitrarily close to a fix point of $\mathbf{prox}_{\alpha f}^P$ with respect to the P-norm [18, 19].

Proof:

See Lemma 3.1 of the work of Chen and Teboulle [20].

4
Semidefinite programming

In this work we are going to consider a formulation of semidefinite programming where the inequalities are explicitly stated, avoiding the use of slacks variables. The formulation referred to as *general* SDP form is defined as the following

$$\begin{aligned} \underset{X\in\mathbb{S}^n}{\text{minimize}} \quad & \mathbf{tr}(CX) \\ \text{subject to} \quad & \mathcal{A}(X) = b, \\ & \mathcal{G}(X) \leq h, \\ & X \succeq 0. \end{aligned} \tag{4-1}$$

where the operators $\mathcal{A} : \mathbb{S}^n_+ \to \mathbb{R}^m$ and $\mathcal{G} : \mathbb{S}^n_+ \to \mathbb{R}^p$ are given by

$$\mathcal{A}(X) = \begin{bmatrix} \mathbf{tr}(A_1X) \\ \mathbf{tr}(A_2X) \\ \vdots \\ \mathbf{tr}(A_mX) \end{bmatrix}, \quad \mathcal{G}(X) = \begin{bmatrix} \mathbf{tr}(G_1X) \\ \mathbf{tr}(G_2X) \\ \vdots \\ \mathbf{tr}(G_pX) \end{bmatrix}.$$

and the problem data are the symmetric matrices $A_1, \dots, A_m, G_1, \dots, G_p, C \in \mathbb{S}^n$, the vectors $b \in \mathbb{R}^m$ and $h \in \mathbb{R}^p$. In this semidefinite programming formulation, one wants to minimize a linear function subject to a set of m linear equality constraints and p linear inequality constraints, where the decision variable is an $n\times n$ symmetric matrix constrained to be on the positive semidefinite (p.s.d.) cone.

The dual problem takes the form

$$\begin{aligned} \underset{y\in\mathbb{R}^{m+p}}{\text{maximize}} \quad & [b^T \; h^T]\, y \\ \text{subject to} \quad & \mathcal{A}^T(y) + \mathcal{G}^T(y) \preceq C, \\ & y_j \leq 0, \;\; \forall\, j = m+1, \dots, p. \end{aligned}$$

where the transpose operators $\mathcal{A}^T : \mathbb{R}^m \to \mathbb{S}^n_+$ and $\mathcal{G}^T : \mathbb{R}^p \to \mathbb{S}^n_+$ are given by $\mathcal{A}^T(y) = \sum_{i=1}^m y_iA_i$ and $\mathcal{G}^T(y) = \sum_{j=1}^p y_{j+m}G_j$ respectively. Despite being very similar to linear programming, strong duality does not always hold for SDP. For a comprehensive analysis on the duality of semidefinite programming the reader should refer to the work of Boyd and Vandenberghe [1].

4.1 Applications

Besides the theoretical motivation, several problems of practical interest lie precisely in the SDP class. As more applications are found, an efficient method for solving large scale SDP problems is required. This section aims to briefly cover some representative applications that have been successfully solved by SDP. For a more complete list of applications and SDP problems, the reader can refer to [21, 22].

Traditionally, semidefinite programming has been widely used in control theory. Classic applications such as stability of dynamic systems and stochastic control problems [23] have motivated the development of SDP over decades. Modern applications such as motion for humanoid robots [24] use SDP as a core element for control. In several cases, SDP problems in control can be formulated in the form of *linear matrix inequalities* (LMI) [25]. In this sense, for some particular applications in control, there is no need to use general SDP algorithms since some LMI problems have closed form solutions, although adding little complexity to the problem might render the closed form solutions useless. As a consequence, interior-point methods particularly developed to solve LMI problems were proposed [26]. An extensive literature on LMI can be found on [27, 28].

Subsequently, several fields of study have found in SDP a powerful tool for solving complex problems. For instance, the use of SDP in power systems allowed deriving tight bounds and solutions for more realistic optimal power flow models with alternating current networks [29]. In chip design, transistor sizing also was optimized with the use of SDP [30, 31]. In the field of structural truss layout, the use of SDP has been popularized after the seminal work of Ben-Tal and Nemirovski [32]. This latter work also presented the first ideas that subsequently lead to the expanding field of optimization under uncertainty [33], where SDP is also used to approximate chance constraints [34].

A remarkable property of SDP is the ability to build tight convex relaxations to NP-hard problems. This technique, also known as semidefinite relaxation (SDR), has been a powerful tool bridging convex and combinatorial optimization. In the early nineties, Lovász and Schrijver developed a SDR for optimization problems with the presence of boolean variables [35]. Soon after, SDR gained momentum after the celebrated Goemans-Williamson randomized rounding method for the max-cut problem [36]. Eventually, similar SDR approaches were proposed for other combinatorial problems, such as the max-3-sat [37] and the traveling salesman problem [38]. Recently, Candès et al. proposed an SDR approach to the phase retrieval problem [39]. Such relaxations generally square the original number of decision variables through a technique called *lifting* [40]. From the algorithmic perspective, this can quickly make moderate size instances computationally challenging.

In the last decade, applications in machine learning have challenged traditional SDP methods at solving remarkably large scale instances [41]. The problem size n is usually associated with the sample size, which can easily reach millions. One of the most celebrated application is the *matrix completion* problem [42] which became very popular with the *Netflix prize* [43]. In graphical models, the problem of covariance selection, which is a powerful tool for modeling dependences between random variables, can also be formulated as an SDP problem [44]. In statistical learning, finding the best model that combines different positive definite kernels, also known as kernel learning, can be achieved by solving an SDP problem [45, 46].

For *constraint satisfaction problems* (CSP), i.e. problems where one tries to satisfy as many constraints as possible, semidefinite programming also plays an essential role. In the work of Raghavendra [2], it was shown that, if the *Unique Games* conjecture [47] is true, then semidefinite programming achieves the best approximation for every CSP. Even though there is no consensus on the legitimacy of the Unique Games conjecture, recent work by Khot et al. [48, 49] provides new results suggesting that the Unique Games conjecture may be true. Recent developments bring semidefinite programming back to the spotlight and imply that this class of convex optimization algorithms do have singular properties worth exploiting.

4.2 SDP solution methods

In the early days, general SDP problems were solved by the ellipsoid method [50, 51] and subsequently by bundle methods [52]. After the advent of the first polynomial-time interior-point algorithm for linear programming by Karmarkar et al. [53, 54], Nesterov and Nemirovski extended interior-points methods for other classes of convex optimization problems [55]. Shortly afterwards, a range of interior-points methods to solve SDPs were proposed [56–58]. The solvers CSDP [59] and MOSEK [60] use state-of-the-art interior point methods for solving general SDP problems. Up to medium size problems, this class of methods is preferable due to the fast convergence and high precision. However, as it is inherent to all second-order methods, the use of interior-points algorithms may be prohibitive for solving large-scale instances. The main bottleneck is due to the cumbersome effort for computing and storing the Hessian at each iteration.

More recently, first-order methods have been widely used for applications in machine learning and signal processing. Even though first-order methods generally have slower convergence rate, their cost per iteration is usually small and they require less memory allocation [61]. These characteristics make first-order methods very appealing for large scale problems and, consequently, it has been an intense area of research in several fields, e.g. image processing [62]. A great example of the use of first order methods is the conic solver SCS developed by O'Donoghue et al. [63], which can efficiently solve general conic optimization problems to modest accuracy. More recently, the use of the *alternating direction method of multipliers* (ADMM) for specifically solving SDP has been proposed by [64].

For most of the above mentioned algorithms, exploiting sparsity patterns in the decision variable is not as straightforward as it is for other classes of convex optimization problems. In this sense, a major recent contribution has been made by showing that sparsity can be exploited by means of chordal decomposition techniques [65–67]. This approach has enabled parallel implementations that can solve larger instances with the use of supercomputers [68]. In a series of works, Zhang and Lavaei have presented SDP algorithms that can properly take advantage of the problem's sparsity [69, 70].

5 Primal-dual operator splitting for SDP

In this chapter, the proposed primal-dual algorithm will be built from the first-order optimality conditions of the SDP in general form (4-1). The strategy adopted to derive the algorithm is to translate the problem of finding a solution that satisfies the optimality conditions into a problem of finding a fixed point of a related monotone operator. This approach has been previously adopted with the purpose of designing new algorithms and developing alternative proofs for existing ones [19]. Further information on the use of monotone operators in the context of convex optimization can be found at [12, 71–73].

Consider the general SDP form (4-1) where the problem constraints are encoded by indicator functions

$$\underset{X \in \mathbb{S}^n}{\text{minimize}} \quad \mathbf{tr}(CX) + I_{\mathbb{S}^n_+}(X) + I_{\substack{=b \\ \leq h}}(\mathcal{M}(X)), \quad \mathcal{M} = \begin{bmatrix} \mathcal{A} \\ \mathcal{G} \end{bmatrix} \tag{5-1}$$

and the indicator functions are defined as follows

$$I_{\mathbb{S}^n_+}(X) = \begin{cases} 0, & \text{if } X \succeq 0, \\ \infty, & \text{otherwise,} \end{cases}$$

encodes the positive semidefinite cone constraint and

$$I_{\substack{=b \\ \leq h}}(u) = I_{=b}(u_1) + I_{\leq h}(u_2), \qquad u = [u_1 \;\; u_2]^T,$$

encodes the linear constraints right-hand side for any $u \in \mathbb{R}^{m+p}$ such that

$$I_{=b}(u_1) = \begin{cases} 0, & \text{if } u_1 = b, \\ \infty, & \text{otherwise,} \end{cases} \qquad I_{\leq h}(u_2) = \begin{cases} 0, & \text{if } u_2 \leq h, \\ \infty, & \text{otherwise.} \end{cases}$$

for any $u_1 \in \mathbb{R}^m$ and $u_2 \in \mathbb{R}^p$.

5.1 Optimality condition

The first order optimality condition (5-2) for the optimization problem (4-1) can be expressed as

$$0 \in \partial\, \mathbf{tr}(CX) + \partial\, I_{\mathbb{S}^n_+}(X) + \mathcal{M}^T(\partial\, I_{\substack{=b \\ \leq h}}(\mathcal{M}(X))). \tag{5-2}$$

By introducing an auxiliary variable $y \in \mathbb{R}^{m+p}$, the optimality condition can be recast as the following system of inclusions

$$\begin{aligned} &0 \in \partial\,\mathbf{tr}(CX) + \partial\, I_{\mathbb{S}^n_+}(X) + \mathcal{M}^T(y), \\ &y \in \partial\, I_{\substack{=b\\ \leq h}}(\mathcal{M}(X)). \end{aligned}$$

By definition, the auxiliary variable y represents the dual variable associated with the problem constraints. This statement is easily verifiable since $y \in \partial\, I_{\substack{=b\\ \leq h}}(\mathcal{M}(X))$, i.e. y is a subgradient of $I_{\substack{=b\\ \leq h}}(\mathcal{M}(X))$ at X. Since problem (4-1) is convex, finding a pair (X^*, y^*) satisfying (5-2) is equivalent to finding an optimal primal-dual pair for (4-1), as long as strong duality holds [8].

From Lemma 4 we know that $(\partial f)^{-1} = \partial f^*$. One can manipulate the second equation as follows

$$\begin{aligned} y \in \partial\, I_{\substack{=b\\ \leq h}}(\mathcal{M}(X)) &\iff \partial\, I^*_{\substack{=b\\ \leq h}}(y) \ni \mathcal{M}(X), \\ &\iff 0 \in \partial\, I^*_{\substack{=b\\ \leq h}}(y) - \mathcal{M}(X). \end{aligned}$$

Using this new expression, the system (5-2) can be recast as $0 \in F(X, y)$, where $F : \mathbb{S}^n \times \mathbb{R}^{m+p} \to \mathbb{S}^n \times \mathbb{R}^{m+p}$ is the following monotone operator

$$F(X, y) = \Big(\partial\,\mathbf{tr}(CX) + \partial\, I_{\mathbb{S}^n_+}(X) + \mathcal{M}^T(y)\,,\ \partial\, I^*_{\substack{=b\\ \leq h}}(y) - \mathcal{M}(X)\Big). \tag{5-3}$$

One can verify that F is a monotone operator by noticing that F is the sum of a subdifferential operator and a monotone affine operator [74]

This new formulation of the system (5-2) implies that finding a zero of the underlying monotone operator F is equivalent to finding an optimal primal-dual pair for the semidefinite programming problem (4-1). On the remainder of this section, a method for finding a zero for the operator F will be established.

5.2 Fixed-point algorithm

Finding a zero of the monotone operator F can be translated into finding a fixed point for the following system

$$P(X, y) \in \alpha F(X, y) + P(X, y),$$

by simply adding $P(X, y)$ on both sides. Where P is a positive definite operator that works as preconditioner for the fixed-point inclusion [8]. This formulation induces the following fixed point iteration

$$\Big(X^k, y^k\Big) \leftarrow \Big(P + \alpha F\Big)^{-1} P(X^{k-1}, y^{k-1}). \tag{5-4}$$

By choosing P as

$$P = \begin{bmatrix} I & -\alpha\mathcal{M}^T \\ -\alpha\mathcal{M} & I \end{bmatrix}, \tag{5-5}$$

the fixed-point inclusions can be expressed as

$$\begin{aligned}&\Big(X^{k-1} - \alpha\mathcal{M}^T(y^{k-1}),\ y^{k-1} - \alpha\mathcal{M}(X^{k-1})\Big) \in \\ &\alpha F(X^k, y^k) + \Big(X^k - \alpha\mathcal{M}^T(y^k),\ y^k - \alpha\mathcal{M}(X^k)\Big).\end{aligned}$$

By substituting the operator $F(X^k, y^k)$, we have that

$$\begin{aligned}&\Big(X^{k-1} - \alpha\mathcal{M}^T(y^{k-1}),\ y^{k-1} - \alpha\mathcal{M}(X^{k-1})\Big) \in \\ &\alpha\Big(\partial\,\mathbf{tr}(CX^k) + \partial\, I_{\mathbb{S}^n_+}(X^k) + \mathcal{M}^T(y^k)\,,\ \partial\, I^*_{\substack{=b \\ \leq h}}(y^k) - \mathcal{M}(X^k)\Big) + \\ &\Big(X^k - \alpha\mathcal{M}^T(y^k),\ y^k - \alpha\mathcal{M}(X^k)\Big).\end{aligned}$$

Further manipulation leads to the system

$$\begin{aligned}&\Big(X^{k-1} - \alpha\mathcal{M}^T(y^{k-1}),\ y^{k-1} + \alpha\mathcal{M}(2X^k - X^{k-1})\Big) \in \\ &\alpha\Big(\partial\,\mathbf{tr}(CX^k) + \partial\, I_{\mathbb{S}^n_+}(X^k),\ \partial\, I^*_{\substack{=b \\ \leq h}}(y^k)\Big) + \Big(X^k,\ y^k\Big)\end{aligned}$$

which induces the following fixed-point iteration:

$$\begin{aligned}X^k &\leftarrow \Big(I + \alpha\partial\left(\mathbf{tr}(C\cdot) + I_{\mathbb{S}^n_+}\right)\Big)^{-1}\Big(X^{k-1} - \alpha\mathcal{M}^T(y^{k-1})\Big), \\ y^k &\leftarrow \Big(I + \alpha\partial\, I^*_{\substack{=b \\ \leq h}}\Big)^{-1}\Big(y^{k-1} + \alpha\mathcal{M}(2X^k - X^{k-1})\Big).\end{aligned}$$

This scheme is a particular case of the primal-dual hybrid gradient (PDHG) proposed by Chambolle and Pock [4,75] which has been successfully applied to a wide range of image processing problems, such as image denoising and deconvolution [62,76,77].

5.3 Convergence

Lemma 11 *The matrix P is positive definite if and only if* $0 < \alpha < 1/\left\|\mathcal{M}\right\|_2$.

Proof:

By definition, P is positive definite if $\langle P(X,y),(X,y)\rangle > 0\,\forall\,(X,y) \in \mathbb{S}^n \times \mathbb{R}^{m+p}$. Lets pick an arbitrary pair (X,y), we have that

$$\langle P(X,y),(X,y)\rangle = \|X\|_F^2 + \|y\|_2^2 - \alpha 2\langle\mathcal{M}(X), y\rangle.$$

In this sense, P will be positive definite if

$$\forall\, (X,y) \in \mathbb{S}^n \times \mathbb{R}^{m+p}, \quad \|X\|_F^2 + \|y\|_2^2 > \alpha 2 \langle \mathcal{M}(X), y\rangle.$$

From the Cauchy-Schwarz inequality [78] we know that

$$\forall\, (X,y) \in \mathbb{S}^n \times \mathbb{R}^{m+p}, \quad \alpha 2 \langle \mathcal{M}(X), y\rangle \le \alpha 2 \left\|\mathcal{M}\right\|_2 \left\|X\right\|_F \left\|y\right\|_2 .$$

Applying Young's inequality [78] to the right hand side gives

$$\forall\, (X,y) \in \mathbb{S}^n \times \mathbb{R}^{m+p}, \quad \alpha 2 \left\|\mathcal{M}\right\|_2 \left\|X\right\|_F \left\|y\right\|_2 \le \alpha 2 \left\|\mathcal{M}\right\|_2 \Big(\frac{\|X\|_F^2}{2} + \frac{\|y\|_2^2}{2}\Big).$$

This implies that

$$\forall\, (X,y) \in \mathbb{S}^n \times \mathbb{R}^{m+p}, \quad \alpha 2 \langle \mathcal{M}(X), y\rangle \le \alpha \left\|\mathcal{M}\right\|_2 \Big(\, \|X\|_F^2 + \|y\|_2^2 \Big).$$

Therefore, if $\alpha < \frac{1}{\|\mathcal{M}\|_2}$ then P will be positive definite.

□

Theorem 12 *If* $0 < \alpha < 1/\left\|\mathcal{M}\right\|_2$*, the iterative process (5-4) is equivalent to the generalized proximal point method (3-1).*

Proof:

Given that under the condition the $0 < \alpha < 1/\left\|\mathcal{M}\right\|_2$ the matrix P is positive definite, we can consider the associated generalized proximal operator $\mathbf{prox}_{\alpha F}^P$. Leading to the following iterative scheme

$$(X^{k+1}, y^{k+1}) \leftarrow \mathbf{prox}_{\alpha F}^P(X^k, y^k),$$

which is equivalent to successively finding a pair (X^{k+1}, y^{k+1}) that satisfies the first order optimality condition

$$0 \in F(X^{k+1}, y^{k+1}) + \tfrac{1}{\alpha}\Big((X^{k+1}, y^{k+1}) - (X^k, y^k)\Big)^T P,$$

which can be rearranged as

$$P(X^k, y^k) \in \alpha F(X^{k+1}, y^{k+1}) + P(X^{k+1}, y^{k+1}),$$

leading to the inclusion

$$\Big(X^{k+1}, y^{k+1}\Big) \in \Big(P + \alpha F\Big)^{-1} P(X^k, y^k).$$

□

Given that the iterative scheme (5-4) is equivalent to GPPA, we know from Theorem 10 that the sequence $\{(X^k, y^K)\}$ will monotonically converge to a fixed point (X^*, y^*) of $P(X, y) \in \alpha F(X, y) + P(X, y)$ if one exists. Additionally, from Theorem 7, we know that the same point (X^*, y^*) will be the optimal solution of (5-1).

5.4 Termination criteria

In practice, the progress of the algorithm can be measured by the primal, dual and combined residuals respectively defined as follows

$$\begin{aligned}
\epsilon^k_{\text{primal}} &= \left\| \tfrac{1}{\alpha}(X^k - X^{k-1}) - \mathcal{M}^T(y^k - y^{k-1}) \right\|_F, \\
\epsilon^k_{\text{dual}} &= \left\| \tfrac{1}{\alpha}(y^k - y^{k-1}) - \mathcal{M}(X^k - X^{k-1}) \right\|_2, \\
\epsilon^k_{\text{comb}} &= \epsilon^k_{\text{primal}} + \epsilon^k_{\text{dual}}.
\end{aligned}$$

This stopping criterion is equivalent to the criterion proposed by [79] and can be derived from the optimality conditions of (5-1). One can stop the iterative scheme (5-4) as soon as the combined residual is less than a predetermined tolerance.

5.5 Proximal operators in SDP

In order to employ the fixed-point iteration (5-4) one needs to compute both resolvent operators $(I+\alpha\partial\,(\mathbf{tr}(C\cdot)+I_{\mathbb{S}^n_+}))^{-1}$ and $(I+\alpha\partial\, I^*_{\substack{=b\\ \leq h}})^{-1}$. As it was stated in Lemma (5) in chapter 3, the resolvent of a subdifferential operator is given by its associated proximal operator. In this sense, in this section, the proximal operators associated with (5-4) are going to be analyzed in more detail.

5.5.1 Box constraints

The resolvent associated with $\partial I_{\substack{=b\\ \leq h}}$ is simply given by the projection onto the box constraints

$$\left(I + \alpha\partial I_{\substack{=b\\ \leq h}}\right)^{-1}(u) = \mathbf{proj}_{\substack{=b\\ \leq h}}(u) = \begin{bmatrix} \mathbf{proj}_{=b}(u_1) \\ \mathbf{proj}_{\leq h}(u_2) \end{bmatrix} = \begin{bmatrix} b \\ \min\{u_2, h\} \end{bmatrix},$$

where $u_1 \in \mathbb{R}^p$ and $u_2 \in \mathbb{R}^m$ and $\min\{u_2, h\}$ is the point-wise minimum. Additionally, Lemma 6 gives the Moreau identity

$$u = \Big(I + \alpha\partial f^*\Big)^{-1}(u) + \alpha\Big(I + \partial f/\alpha\Big)^{-1}(u/\alpha).$$

Therefore one concludes that

$$\Big(I + \alpha\partial I^*_{\substack{=b\\ \leq h}}\Big)^{-1}(u) = u - \alpha\,\mathbf{proj}_{\substack{=b\\ \leq h}}(u/\alpha). \tag{5-6}$$

5.5.2 Positive semidefinite cone

Similarly, the resolvent associated with the positive semidefinite constraint is given by the Euclidean projection onto the positive semidefinite cone. Let $S \in \mathbb{S}^n$, the projection onto the set $\{X : X \succeq 0\}$ has the closed form

$$\Big(I + \alpha\partial I_{\mathbb{S}^n_+}\Big)^{-1}(S) = \mathbf{proj}_{\mathbb{S}^n_+}(S) = \sum_{i=1}^{n} \max\{0, \lambda_i\} u_i u_i^T,$$

where $S = \sum_{i=1}^n \lambda_i u_i u_i^T$ is the eigenvalue decomposition of the symmetric matrix S [9].

5.5.3 Trace

Given the symmetric matrices C and S, the resolvent associated to the trace function is given by the formula

$$\Big(I + \alpha\partial\mathbf{tr}(C\cdot)\Big)^{-1}(S) = S - \alpha C.$$

Unlikely the majority of cases, the resolvent associated with the trace function plus any convex function g is given by the left composition as in

$$\begin{aligned}\Big(I + \alpha\partial(g + \mathbf{tr}(C\cdot)\Big)^{-1}(S) &= \Big(I + \alpha\partial g\Big)^{-1} \circ \Big(I + \alpha\partial\mathbf{tr}(C\cdot)\Big)^{-1}(S),\\ &= \Big(I + \alpha\partial g\Big)^{-1}\Big(S - \alpha C\Big).\end{aligned}$$

Consequently,

$$\Big(I + \alpha\partial\left(\mathbf{tr}(C\cdot) + I_{\mathbb{S}^n_+}\right)\Big)^{-1}(S) = \mathbf{proj}_{\mathbb{S}^n_+}(S - \alpha C). \tag{5-7}$$

5.6 PD-SDP

Algorithm 1, referred to as PD-SDP, as in Primal-Dual SemiDefinite Programming, matches the fixed-point iteration (5-4) and the resolvents in its closed forms (5-6, 5-7). In this particular setting, the primal-dual method turns out to be a very simple routine. As it is illustrated in Algorithm 1, the method avoids explicitly solving a linear system or a convex optimization problem at each iteration. One only needs a subroutine to evaluate the resolvents and access to an abstract linear operator for $\mathcal{M}$ and its adjoint.

Algorithm 1 PD-SDP

```
Given: $\mathcal{M}$, $b \in \mathbb{R}^m$, $h \in \mathbb{R}^p$ and $C \in \mathbb{S}^n$.
while $\epsilon^k_{\text{comb}} > \epsilon_{\text{tol}}$ do
    $X^{k+1} \leftarrow \mathbf{proj}_{\mathbb{S}^n_+}(X^k - \alpha(\mathcal{M}^T(y^k) + C))$        ▷ Primal step
    $y^{k+1/2} \leftarrow y^k + \alpha\mathcal{M}(2X^{k+1} - X^k)$        ▷ Dual step part 1
    $y^{k+1} \leftarrow y^{k+1/2} - \alpha\,\mathbf{proj}_{\substack{=b \\ \leq h}}(y^{k+1/2}/\alpha)$        ▷ Dual step part 2
end while
return $\left(X^{k+1}, y^{k+1}\right)$
```

In [4], it was shown that PDHG achieves $\mathcal{O}(1/k)$ convergence rate for non-smooth problems, where k is the number of iterations. In this sense, in terms of convergence rate, the PD-SDP is optimal among all the possible first-order methods. Similar convergence rates can be achieved by other operator splitting methods such as Tseng's ADM [80] or ADMM [81,82]. However, the PD-SDP has the advantage of offering the optimal dual variable as a by-product of the algorithm.

The computational complexity of each loop is dominated by the projection onto the positive semidefinite cone. In the most naive implementation, each iteration will cost $\mathcal{O}(n^3)$ operations, where n is the dimension of the p.s.d. matrix. If one knew the number of positive eigenvalues r, at each iteration, the computational cost could be reduced to $\mathcal{O}(n^2r)$. In this work we are going to refer to r as the *target-rank* of a particular iteration. One should keep in mind that the rank of the matrix will be smaller than or equal to r. Unfortunately, in practice, one does not have access to the *target-rank*. However, as it will be show in the next section, there is no need to know the *target-rank* in advance. Even more surprisingly, faster running times can be achieved by underestimating the target rank to some extent.

6
Exploiting low-rank structure

So far we have proposed a first-order method for solving general SDP problems. However, the projection onto the positive semidefinite cone is an obstacle to make the algorithm scalable for larger instances. In this section, we are going to explore how to take advantage of a low-rank structure, even if the target rank is unknown.

6.1
Low-rank approximation

It is well-known that SDP solutions very often exhibit a low-rank structure. More precisely, as shown by Barvinok [83] and Pataki [84], any SDP with m equality constraints admits an optimal solution with rank at most $\sqrt{2m}$. In practice, for several SDP problems, it is frequently observed that the optimal solution has an even smaller rank. This phenomenon is notably present in SDPs generated by a semidefinite relaxation, where the solution has ideally low rank. However, even if the relaxation is inexact, the rank of the solution is usually substantially small.

This property has motivated a series of nonconvex methods aiming to exploit the low-rank structure of the problem [85–87]. For instance, one can encode the positive semidefinite constraint as a matrix factorization of the type $X = V^T V$ where $V \in \mathbb{R}^{r \times n}$ and r is the target rank. This technique was proposed a decade ago by Burer and Monteiro [88] and since then it has been one of the main tools for tackling the scalability of low-rank SDPs. This matrix factorization approach has been successfully applied to large-scale computer vision [89] and combinatorial optimization problems [90]. Unfortunately, by resorting to this approach, one loses convexity and all the associated guarantees.

The main bottleneck of `PD-SDP` and any other convex optimization methods for solving SDPs is computing the eigenvalue decomposition. A natural approach to overcome this issue is to make use of low-rank matrix approximation techniques in place of computing the full matrix decomposition. Recent work by Udell, Tropp et al. [91] uses matrix sketching methods [92, 93] to successfully find approximate solutions to low-rank convex problems. While their methodology possesses several advantages, such as optimal storage, it does require all solutions to be low-rank in order to guarantee convergence to an optimal solution. In contrast, the methodology proposed in this paper exploits the low-rank structure of the problem whenever possible but also converges to an optimal solution even in the presence of a full-rank structure.

As it was shown by Eckart and Young [94], the best rank-r approximation of symmetric matrices, for both the Frobenius and the spectral norms, is given by the truncated eigenvalue decomposition. Inspired by this result, the *approximate* projection onto the positive semidefinite cone is given by

$$\mathbf{aproj}_{\mathbb{S}^n_+}(X, r) = \sum_{i=1}^{r} \max\{0, \lambda_i\} u_i u_i^T, \tag{6-1}$$

where X is a symmetric matrix, r is its *target-rank* and $\lambda_1 \geq \cdots \geq \lambda_r$ are the eigenvalues with the r largest real values. It is important to notice that, despite being different from the Euclidean projection, $\mathbf{aproj}_{\mathbb{S}^n_+}(X, r)$ does project the matrix X onto the p.s.d. cone. In other words, the truncated projection maps into the p.s.d. cone but not necessarily the closest point, according to the Frobenius norm, as it is illustrated in Figure 6.1.

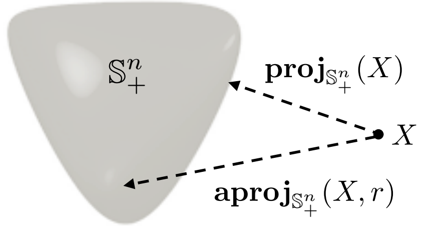

$\mathbb{S}^n_+$
$\mathbf{proj}_{\mathbb{S}^n_+}(X)$
X
$\mathbf{aproj}_{\mathbb{S}^n_+}(X, r)$

Figure 6.1: Comparison of Euclidean projection onto the positive semidefinite cone, denoted by $\mathbf{proj}_{\mathbb{S}^n_+}(X)$, and the truncated projection given by $\mathbf{aproj}_{\mathbb{S}^n_+}(X, r)$.

If the target-rank r equals the number of nonzero eigenvalues, both the the truncated and the full projection will be equivalent. Otherwise, if the target-rank r is smaller than the number of nonzero eigenvalues, the truncated projection will be only an approximation of the exact projection. In this case, according to the Eckart–Young–Mirsky theorem, the approximation error can be expressed as the sum of the eigenvalues that were left out by the truncated projection:

$$\left\| \mathbf{proj}_{\mathbb{S}^n_+}(X) - \mathbf{aproj}_{\mathbb{S}^n_+}(X, r) \right\|_F^2 = \sum_{i=r+1}^{n} \max\{\lambda_i, 0\}.$$

For practical purposes, the approximation error can be bounded in terms of the smaller eigenvalue computed by the truncated projection as the following

$$\left\|\mathbf{proj}_{\mathbb{S}^n_+}(X) - \mathbf{aproj}_{\mathbb{S}^n_+}(X, r)\right\|_F^2 \leq (n - r)\max\{\lambda_r, 0\}. \tag{6-2}$$

The partial eigenvalue decomposition (6-1) can be efficiently computed via power iteration algorithms or Krylov subspace methods [95,96]. Computational routines are freely available on almost every programming language for numerical computing [97,98].

6.2 Target rank update

As previously noted, the `PD-SDP` method can be seen as a fixed-point iteration of a monotone operator [99]. In this sense, replacing the exact projection onto the positive semidefinite cone by its approximation (6-1) will result in an inexact iteration as the following

$$\left(X^{k+1}, u^{k+1}\right) \leftarrow \left(P + \alpha F\right)^{-1} P(X^k, u^k) + \varepsilon^k, \tag{6-3}$$

where F and P are the ones defined in (5-3) and (5-5) and ε is an error component. In the literature, this methodology can be found by the name of *inexact solves* or *approximate proximal point* [15]. In the work of Eckstein and Bertsekas [82], they have shown that the approximate scheme (6-3) converges as long as the error component is summable, i.e.,

$$\sum_{k=1}^{\infty} \left\|\varepsilon^k\right\|_2 < \infty. \tag{6-4}$$

In the context of the approximate projection onto the p.s.d. cone, condition (6-4) can be expressed in terms of the smallest eigenvalue of the truncated decomposition for each iterate. Let λ_r^k denote the smallest eigenvalue computed at the k^{th} iteration of the algorithm (6-3), which corresponds to the r^{th} largest eigenvalue at that iteration. Analogously to [82], given a target-rank r, the fixed-point iteration (6-3) will converge to a fixed point as long as

$$(n - r)\sum_{k=1}^{\infty} \left\|\max\{\lambda_r^k, 0\}\right\|_2 < \infty \tag{6-5}$$

and a fixed-point exists. It is easily verifiable that for an arbitrarily fixed target-rank r, the iteration (6-3) will never converge. For instance, if one fixes the target-rank to a value smaller than the rank of the optimal solution, the error component will remain above a threshold and the sequence of errors will not be summable. We refer to the target-rank as *sufficient* if it satisfies the condition (6-5).

Since the minimal sufficient target-rank is not known *a priori*, it is necessary to use an update mechanism that can guarantee the convergence of (6-3). The strategy adopted in this paper is to start the algorithm with a small target-rank and increase its value whenever necessary. The combined residual (5.4) will be used to describe the *state* of the algorithm and trigger the updates of the target-rank. Given an initial target-rank, the sum of the subsequent combined residuals can either converge or diverge. Even if the sequence converges it will not necessarily be monotonic. In this regard, instead of checking convergence of sub-sequential iterates we are going to evaluate the residuals (5.4) within a window of size ℓ.

In case the combined residuals converge according to a given tolerance, we need to examine the approximation error (6-2). It follows from (6-2) that if the approximation error is zero, the smallest eigenvalue of X^* is smaller or equals to zero and the truncated projection is no longer an approximation. Therefore, the inexact iteration (6-3) has also converged to a fixed-point of (5-4) and consequently an optimal primal-dual solution for the SDP problem of interest has been found.

If the combined residual has converged with a target rank r but the approximation error is greater than the tolerance, the target-rank needs to be updated. In this case, it is interesting to notice that even though the current iterate pair is not an optimal point, it does give a feasible primal solution, under the assumption of strong duality. By characterizing a fixed-point of (6-2), the current iterate will be satisfying the linear constraints of the original SDP problem. Additionally, as it was previously pointed out, the truncated projection maps onto the positive semidefinite cone. Therefore, given a target rank r, if a fixed-point of (6-2) is found, one has a feasible point designated by $(X_{[r]}, y_{[r]})$.

The last possible case occurs when the combined residuals either stay stationary or diverge within the last ℓ iterates. In this case, the target-rank also needs to be updated. After updating the target-rank, the process is repeated. The combination of `PD-SDP` and the target rank updating scheme is described in Algorithm 2 and will be referred to as `LR-PD-SDP`. In the worst case scenario, the target-rank will be updated until r equals n and the subsequent iterations of the algorithm will be equivalent to the ones in `PD-SDP`. Consequently, in this setting, `LR-PD-SDP` will converge to a fixed point of (5-3) under the same conditions of `PD-SDP`.

Algorithm 2 LR-PD-SDP

```
Given: $\mathcal{M}$, $b \in \mathbb{R}^p$, $h \in \mathbb{R}^q$, $C \in \mathbb{S}^n$ and $r = 1$.
while $(n-r)\lambda_r > \varepsilon_\lambda$ do
    while $\epsilon^k_{\text{comb}} > \epsilon_{\text{tol}}$ and $\epsilon^k_{\text{comb}} < \epsilon^{k-\ell}_{\text{comb}}$ do
        $X^{k+1} \leftarrow \mathbf{aproj}_{\mathbb{S}^n_+}(X^k - \alpha(\mathcal{M}^T(y^k) + C), r)$   ▷Approximate primal step
        $y^{k+1/2} \leftarrow y^k + \alpha\mathcal{M}(2X^{k+1} - X^k)$   ▷ Dual step part 1
        $y^{k+1} \leftarrow y^{k+1/2} - \alpha\,\mathbf{proj}_{\substack{=b\\ \leq h}}(y^{k+1/2}/\alpha)$   ▷ Dual step part 2
    end while
    if $\epsilon^k_{\text{comb}} < \epsilon_{\text{tol}}$ then
        $(X_{[r]}, y_{[r]}) \leftarrow (X^{k+1}, y^{k+1})$   ▷ Save feasible solution
    end if
    $r \leftarrow 2r$   ▷ *Target-rank* update
end while
return $(X^{k+1}, y^{k+1})$
```

Each iteration of Algorithm 2 has a computational complexity of $\mathcal{O}(n^2 r)$ as opposed to $\mathcal{O}(n^3)$ of the previous version. Additionally, if one doubles the target-rank whenever necessary, the updating procedure can be carried out $\mathcal{O}(\log(n))$ times. Usually, LR-PD-SDP will require more iterations to reach convergence than PD-SDP. On the other hand, LR-PD-SDP induces the rank of the iterates X^k to remain small. As it is illustrated in Figure 2, LR-PD-SDP avoids the presence of high rank iterates, as happens with PD-SDP. Consequently, if the problem of interest has a low-rank solution, the LR-PD-SDP will terminate much faster than PD-SDP.

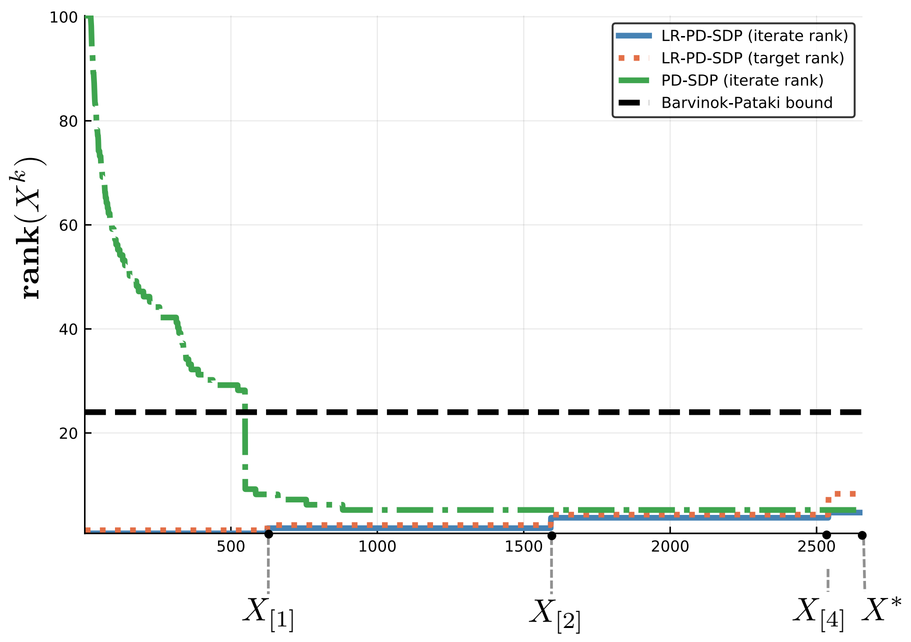


Figure 6.2: Comparison of the rank path of the iterates X^k for both `PD-SDP` and `LR-PD-SDP` methods. Additionally, the sequence of primal intermediate feasible solution found by `LR-PD-SDP` are represented as the points $X_{[1]}, X_{[2]}$ and $X_{[4]}$.

7
ProxSDP numerical solver

The complete implementation of the algorithm is available online at

`https://github.com/mariohsouto/ProxSDP.jl`

It is not only possible to examine the implementation of the solver but also it can be used as a general purpose solver. The project includes usage examples and all the data and scripts needed for next sections' benchmarks.

The solver was completely written in the Julia language [100], making extensive usage of its linear algebra capabilities. Sparse matrix operations were crucial to achieve good performance on manipulations involving the linear constraints. Also, dense linear algebra routines relying on BLAS [101] were heavily used, just like multiple in-place operation to avoid unnecessary memory allocations. The built-in wrappers over LAPACK [102] and BLAS made it very easy to write high performance code. In particular, the ARPACK wrapper, used to efficiently compute the largest eigenvalues, was modified to maximize in-place operations and avoid the unnecessary allocations.

Instead of writing a solver interface from scratch, we used the package MathOptInterface.jl (MOI) that abstracts solver interfaces. In doing that, we were able to write problems only once and test them in all available solvers. Moreover, having a MOI based interface means that the solve is available to through the modeling language JuMP [103].

8
Case studies

In the following sections we will present three SDP problems to serve as background for comparisons between SDP solvers. The main goal of the experiments is to show how `LR-PD-SDP` outperforms the state-of-the-art solvers in the low-rank setting. In this sense, the numerical experiments are focused on semidefinite relaxation problems. On such SDP relaxations, the original problem one is interested in solving is nonconvex and it can be formulated as a SDP plus a rank constraint of the form $\mathbf{rank}(X) = d$, where d usually assumes a small value. Unfortunately, the rank constraint makes the problem extremely hard to solve and any exact algorithm has doubly exponential complexity [104]. The SDR avoids this problem by simply dropping the rank constraint and solving the remaining problem via semidefinite programming. Usually SDRs admit low-rank solutions, even without the presence of the rank constraint, making this the ideal case for testing the `LR-PD-SDP` algorithm.

In the presented numerical experiments, we are going to consider a default numerical tolerance of $\epsilon_{\text{tol}} = 10^{-3}$. As any first-order method, both `PD-SDP` and `LR-PD-SDP` may require a large number of iterations to converge to a higher accuracy [61, 105]. All the following tests were run in a Intel(R) Core(TM) i7-5820K CPU 3.30GHz (12 cores) Linux workstation with 62 Gb of RAM. In the following benchmarks for `PD-SDP` and `LR-PD-SDP` the Julia version used was compiled with Intel's MKL [106]. The maximum running time for all experiments was set to 1200s.

8.1
Graph equipartition

Consider the undirected graph $G = (V, E)$ where V is the set of vertices, E is a set of edges, n is the total number of edges and a cut (S, S') is a disjoint partition of V. Let $x \in \{-1, +1\}^n$ such that

$$x_i = \begin{cases} +1, & \text{if } x_i \in S, \\ -1, & \text{if } x_i \in S', \end{cases} \quad \forall\, i = 1, \cdots, n.$$

Given a set of weights w, the quantity $\frac{1}{4}\sum_{(i,j)\in E} w_{ij}(1 - x_i x_j)$ is called the weight of the cut (S, S'). The *graph equipartition* problem, as illustrated in

Figure (8.1), aims to find the cut with maximum weight on a given graph such that both partitions of the graph have the same cardinality.

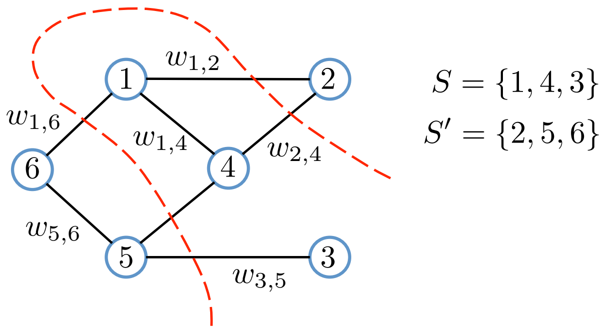


Figure 8.1: Graph equipartition problem

This problem can be formulated as the combinatorial optimization problem (8-1).

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \tfrac{1}{4} \sum_{(i,j)\in E} w_{ij}(1 - x_i x_j) \\ \text{subject to} \quad & x_i \in \{-1,+1\}, \quad \forall\, i = 1,\cdots,n, \\ & \sum_{i=1}^{n} x_i = 0. \end{aligned} \tag{8-1}$$

The binary constraints $x \in \{-1,+1\}^n$, can be expressed as a nonconvex equality constraints of the form $x_i^2 = 1 \,\forall\, i = 1,\cdots,n$. By lifting the decision variables to the space of symmetric matrices $X \in \mathbb{S}^n_+$ and introducing a rank-one constraint, the graph equipartition problem can be formulated as follows

$$\begin{aligned} \underset{X\in\mathbb{S}^n_+}{\text{minimize}} \quad & \mathbf{tr}(WX) \\ \text{subject to} \quad & \mathbf{tr}(1_{n\times n}X) = 0, \\ & \mathbf{diag}(X) = 1, \\ & X \succeq 0, \\ & \mathbf{rank}(X) = 1, \end{aligned}$$

where the symmetric matrix W is composed by the original weights w and $1_{n\times n}$ denotes a $n \times n$ matrix filled with ones. By dropping the rank constraint, one obtains an SDP relaxation. For more details on graph partition problems, the reader should refer to [107].

Problem instances: Graph equipartition instances from the SDPLIB [108] problem set are used to evaluate the performance of the proposed methods. As Table 1 shows, for smaller instances Mosek solver is slightly faster. For larger instances such as equalG11 and equalG51, `LR-PD-SDP` outperforms all other considered methods with a considerable margin. Furthermore, without

n	sdplib	SCS	CSDP	MOSEK	PD-SDP	LR-PD-SDP
124	gpp124-1	1.6	0.4	**0.2**	0.7	0.9
124	gpp124-2	1.5	0.4	0.3	0.5	**0.2**
124	gpp124-3	1.6	0.3	**0.2**	0.6	**0.2**
124	gpp124-4	1.7	0.5	0.3	0.6	**0.2**
250	gpp250-1	21.4	2.9	**0.9**	3.7	1.4
250	gpp250-2	7.8	2.2	**1.1**	4.1	1.2
250	gpp250-3	12.6	2.1	**0.9**	3.4	**0.9**
250	gpp250-4	16.4	2.2	0.9	3.8	**0.6**
500	gpp500-1	134.2	59.1	8.2	22.7	**5.6**
500	gpp500-2	97.4	12.2	8.6	21.5	**6.1**
500	gpp500-3	64.4	12.1	8.9	15.5	**4.4**
500	gpp500-4	71.4	13.4	8.7	15.4	**6.5**
801	equalG11	324.2	47.3	32.4	84.3	**11.3**
1001	equalG51	425.1	98.7	83.4	113.5	**22.5**

Table 8.1: Comparison of running times (seconds) for the SDPLIB's graph equipartition problem instances.

exploiting the low-rank structure of the problem, the primal-dual method fails to scale to the larger instances.

8.2 Sensor network localization

Now consider the problem of estimating the position of a set of sensors on a d-dimensional plane [109]. Let $a_1, \dots, a_m \in \mathbb{R}^d$ be a set of anchor points in which the positions are known and let $x_1, \dots, x_n \in \mathbb{R}^d$ be a set of sensor points that are the decision variables corresponding to the location of each sensor. Additionally, as illustrated in Figure (8.2), some Euclidean distances between sensors and between sensors and anchors are also given.

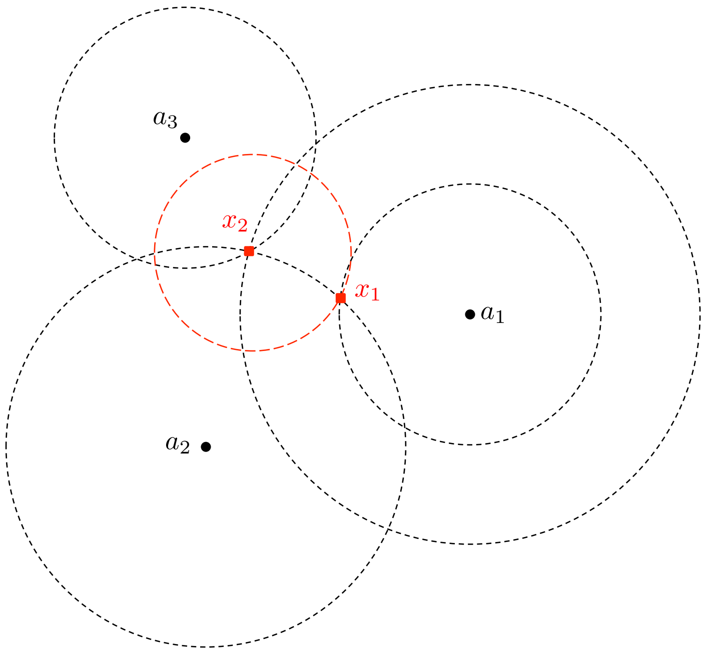


Figure 8.2: Sensor localization problem in two dimensions.

The sensor network localization problem can be originally formulated as the quadratic constrained program (8-2), where the indexes of the distances that are known are either in the set Ω_s or in the set Ω_a. Unfortunately, solving (8-2) is NP-hard [9].

$$\begin{aligned} \underset{x_1,\cdots,x_n\in\mathbb{R}^d}{\text{find}} \quad & x_1,\cdots,x_n \\ \text{subject to} \quad & \|x_i - x_j\|_2^2 = w_{ij}^2, \quad \forall\,(i,j)\in\Omega_s, \\ & \|a_k - x_j\|_2^2 = \tilde{w}_{kj}^2, \quad \forall\,(k,j)\in\Omega_a. \end{aligned} \tag{8-2}$$

One alternative is to formulate (8-2) as the SDR discussed in [110]. In order to start building the SDR, consider the matrices $X \in \mathbb{R}^{d\times n}$ and $Y \in \mathbb{S}^n$ as

$$X = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix} \text{ and } Y = X^TX = \begin{bmatrix} x_1^Tx_1 & x_1^Tx_2 & \dots & x_1^Tx_n \\ x_2^Tx_1 & x_2^Tx_2 & \dots & x_2^Tx_n \\ \vdots & \vdots & \ddots & \vdots \\ x_n^Tx_1 & x_n^Tx_2 & \dots & x_n^Tx_n \end{bmatrix}.$$

Now let $E^{(i,j)} \in \mathbb{S}^n$ be filled with zeros except for the following entries: $E^{(i,j)}_{i,i} = 1$, $E^{(i,j)}_{j,j} = 1$, $E^{(i,j)}_{i,j} = -1$ and $E^{(i,j)}_{j,i} = -1$. With this setting, the constraints that represent the distance between sensors and anchors can be formulated as

$$\mathbf{tr}(E^{(i,j)}Y) = \omega_{ij}^2, \quad \forall\,(i,j)\in\Omega_s. \tag{8-3}$$

Similarly, let $Z \in \mathbb{S}^{d+n}$ be the matrix

$$Z = \begin{bmatrix} I_{d\times d} & X \\ X^T & Y \end{bmatrix}.$$

Additionally, let $U^{(k,j)} \in \mathbb{S}^{d+n}$ be filled with zeros except for the entries: $U^{(k,j)}_{1,1} = a_k^T a_k$, $U^{(k,j)}_{d+j,d+j} = 1$, $U^{(k,j)}_{1:d,d+j} = -a_k$ and $U^{(k,j)}_{d+j,1:d} = -a_k^T$. The constraints regarding the distances between sensors and anchors can be formulated as

$$\mathbf{tr}(U^{(k,j)} Z) = \tilde{\omega}^2_{kj}, \quad \forall\, (k,j) \in \Omega_a. \tag{8-4}$$

Using (8-3), (8-4) and the Schur complement of $Y \succeq X^T X$ [110], the network localization problem can formulated as

$$\begin{aligned} \underset{Z\in\mathbb{S}^{d+n}_+}{\text{find}} \quad & Z \\ \text{subject to} \quad & Z = \begin{bmatrix} I_{d\times d} & X \\ X^T & Y \end{bmatrix} \succeq 0, \\ & \mathbf{tr}(E^{(i,j)} Y) = \omega^2_{ij}, \quad \forall\, (i,j) \in \Omega_s, \\ & \mathbf{tr}(U^{(k,j)} Z) = \tilde{\omega}^2_{kj}, \quad \forall\, (k,j) \in \Omega_a, \\ & \mathbf{rank}(Y) = d. \end{aligned} \tag{8-5}$$

If a unique solution for the given set of distances exists, the SDR obtained by dropping the rank constraint in (8-5) will be exact [110].

Problem instances: In a set of numerical simulation, we randomly generate anchor points and distances measurements. Each anchor and sensor has its position in the two-dimensional Euclidean plane, i.e. $d = 2$. In this sense, if the relaxation is exact the optimal solution Y^* must have a rank of two. This property justifies the `LR-PD-SDP` outperforming other solvers when the number of sensors grow, as it can be seen in Table 2. This claim can be verified by observing that the same operator splitting method, as in `PD-SDP`, does not efficiently scale as n increases.

n	SCS	CSDP	MOSEK	PD-SDP	LR-PD-SDP
50	0.2	0.2	**0.1**	0.5	0.6
100	**0.8**	4.5	0.9	6.1	1.6
150	**2.6**	28.1	3.2	14.4	3.6
200	6.4	89.8	11.2	32.3	**6.1**
250	12.1	239.2	36.4	52.9	**7.9**
300	28.7	timeout	85.2	96.6	**13.5**

Table 8.2: Comparison of running times (seconds) for randomized network localization problem instances.

8.3 MIMO detection

Consider an application in the field of wireless communication known in the literature as binary multiple-input multiple-output (MIMO) [111,112]. As in several MIMO applications, one needs to send and receive multiple data signals over the same channel with the presence of additive noise. The binary MIMO can be modeled as:

$$y = Hx + \varepsilon,$$

where $y \in \mathbb{R}^m$ is the received signal, $H \in \mathbb{R}^{m\times n}$ is the channel and $\varepsilon \in \mathbb{R}^m$ is an i.i.d. Gaussian noise with variance σ^2. The signal, which is unknown to the receiver, is represented by $x \in \{-1,+1\}^n$.

Assuming the noise distribution is known to be Gaussian, a natural approach is to compute the maximum likelihood estimate of the signal by solving the optimization problem:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \|Hx - y\|_2^2 \\ \text{subject to} & x \in \{-1,+1\}^n. \end{array} \tag{8-6}$$

At first sight, the structure of the problem is very similar to a standard least squares problem. However, the unknown signal is constrained to be binary, which makes the problem nonconvex and dramatically changes the problem's complexity. More precisely, solving (8-6) is known to be NP-hard [113].

By using a similar technique as in the graph equipartition problem, one can reformulate (8-6) as the following rank constrained semidefinite problem:

$$\begin{array}{ll} \underset{X\in\mathbb{S}_+^{n+1}}{\text{minimize}} & \mathbf{tr}(LX) \\ \text{subject to} & \mathbf{diag}(X) = 1, \\ & X_{n+1,n+1} = 1, \\ & -1 \le X \le 1, \\ & X \succeq 0, \\ & \mathbf{rank}(X) = 1. \end{array}$$

where the decision variable X is an $n+1 \times n+1$ symmetric matrix and L is given by:

$$L = \begin{bmatrix} H^T H & -H^T y \\ -y^T H & y^T y \end{bmatrix}$$

Given the optimal solution X^* for the relaxation, the solution for the original binary MIMO is obtained by slicing the last column as $x^* = X^*_{1:n,n+1}$. For this particular problem, the SDR is known to be exact if the signal to noise ratio, σ^{-1}, is sufficiently large [111]. In other words, the rank of the optimal solution X^* is guaranteed to be equal to one even without the rank constraint. This extreme low-rank structure makes the ideal case study for the techniques proposed in this paper.

Problem instances: In order to measure the performance of the different methods, problem instances with large signal to noise ratio were randomly generated. For each instance, the channel matrix A is designed as a $n \times n$ matrix with i.i.d. standardized Gaussian entries. The true signal x^* was drawn from a discrete uniform distribution. Since a high signal to noise ratio was used to build the instances, all recovered optimal solutions are rank-one solutions. In this setting, as it is illustrated in Table 3, `LR-PD-SDP` outperforms all other methods as the signal length increases. More surprisingly, `LR-PD-SDP` was able to solve large scale instances with 5000×5000 p.s.d. matrices. The bottleneck found while trying to optimize even larger instances was the amount of memory required by the `ProxSDP` solver.

n	**SCS**	**CSDP***	**MOSEK**	**PD-SDP**	**LR-PD-SDP**
100	1.5	1.2	**0.1**	**0.1**	**0.1**
500	277.8	27.4	2.3	3.1	**1.1**
1000	timeout	97.2	15.6	16.5	**4.7**
2000	timeout	473.6	117.5	115.9	**38.9**
3000	timeout	timeout	418.2	350.6	**122.1**
4000	timeout	timeout	976.8	906.5	**258.3**
5000	timeout	timeout	timeout	timeout	**472.4**

Table 8.3: Running times (seconds) for MIMO detection with high SNR.

9
Conclusions and future work

As a concluding remark, this work has proposed a novel primal-dual method that can efficiently exploit the low-rank structure of semidefinite programming problems. As it was illustrated by the case studies, the proposed technique can achieve up to one order of magnitude faster solving times in comparison to existing algorithms. Additionally, an open source solver, `ProxSDP`, for general SDP problems was made available. We hope that the results and tools contemplated in this work foster the use of semidefinite programming on new applications and fields of study.

One aspect of the proposed methodology not fully explored in this paper, is the value of the intermediate solutions found by `LR-PD-SDP`. For several applications, a suboptimal feasible solution may already be useful. Particularly if one is interested in solving a semidefinite relaxation, a suboptimal solution can be almost as useful as the optimal solution, with the advantage of requiring less computing time to be discovered. For instance, a branch-and-bound search method can benefit from lower bounds that a feasible semidefinite relaxation provides [114]. This ability of quickly generating high quality lower bounds via intermediate feasible solutions can enhance the already well known property of SDP to approximate hard problems.

Another promising future line of work is the combination of chordal decomposition methods with the low-rank approximation presented in this work. If successful, this match would allow the exploitation of both sparsity and low-rank structure simultaneously. Furthermore, it is possible to incorporate other cones to the primal step of `LR-PD-SDP`.

References

[1] VANDENBERGHE, L.; BOYD, S. Semidefinite programming. *SIAM review*, v. 38, n. 1, p. 49–95, 1996.

[2] RAGHAVENDRA, P. Optimal algorithms and inapproximability results for every csp? In: . c2008. p. 245–254.

[3] SOUTO, M.; GARCIA, J. D.; VEIGA, Á. Exploiting low-rank structure in semidefinite programming by approximate operator splitting. *arXiv preprint arXiv:1810.05231*, 2018.

[4] CHAMBOLLE, A.; POCK, T. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, v. 40, n. 1, p. 120–145, 2011.

[5] HE, B.-S. Ppa-like contraction methods for convex optimization: a framework using variational inequality approach. *Journal of the Operations Research Society of China*, v. 3, n. 4, p. 391–420, 2015.

[6] ROCKAFELLAR, R. T. *Convex analysis*. Princeton university press, 2015.

[7] BERTSEKAS, D. P.; NEDI, A.; OZDAGLAR, A. E. et al. Convex analysis and optimization. 2003.

[8] BAUSCHKE, H. H.; COMBETTES, P. L. et al. *Convex analysis and monotone operator theory in hilbert spaces*. Springer, 2011. v. 408.

[9] BOYD, S.; VANDENBERGHE, L. *Convex optimization*. Cambridge university press, 2004.

[10] BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[11] ROCKAFELLAR, R. T. Convex analysis. princeton landmarks in mathematics, 1997.

[12] COMBETTES, P. L.; PESQUET, J.-C. Proximal splitting methods in signal processing. In: *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011. p. 185–212.

[13] BECK, A.; TEBOULLE, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, v. 2, n. 1, p. 183–202, 2009.

[14] DONOHO, D. L.; JOHNSTONE, J. M. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, v. 81, n. 3, p. 425–455, 1994.

[15] ROCKAFELLAR, R. T. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, v. 14, n. 5, p. 877–898, 1976.

[16] RUDIN, L. I.; OSHER, S.; FATEMI, E. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, v. 60, n. 1-4, p. 259–268, 1992.

[17] DOUGLAS, J.; RACHFORD, H. H. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, v. 82, n. 2, p. 421–439, 1956.

[18] COMBETTES, P. L. Fejér monotonicity in convex optimization. In: *Encyclopedia of Optimization*. Springer, 2008. p. 1016–1024.

[19] RYU, E. K.; BOYD, S. Primer on monotone operator methods. *Appl. Comput. Math*, v. 15, n. 1, p. 3–43, 2016.

[20] CHEN, G.; TEBOULLE, M. Convergence analysis of a proximal-like minimization algorithm using bregman functions. *SIAM Journal on Optimization*, v. 3, n. 3, p. 538–543, 1993.

[21] WOLKOWICZ, H.; SAIGAL, R.; VANDENBERGHE, L. *Handbook of semidefinite programming: theory, algorithms, and applications*. Springer Science & Business Media, 2012. v. 27.

[22] VANDENBERGHE, L.; BOYD, S. Applications of semidefinite programming. *Applied Numerical Mathematics*, v. 29, n. 3, p. 283–299, 1999.

[23] LUR'E, A. I. *Some non-linear problems in the theory of automatic control*. Her Majesty's stationery office, 1957.

[24] KUINDERSMA, S.; DEITS, R.; FALLON, M.; VALENZUELA, A.; DAI, H.; PERMENTER, F.; KOOLEN, T.; MARION, P.; TEDRAKE, R. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, v. 40, n. 3, p. 429–455, 2016.

[25] BELLMAN, R.; FAN, K. On systems of linear inequalities in hermitian matrix variables. *Convexity*, v. 7, p. 1–11, 1963.

[26] VANDENBERGHE, L.; BALAKRISHNAN, V. R.; WALLIN, R.; HANSSON, A.; ROH, T. Interior-point algorithms for semidefinite programming problems derived from the kyp lemma. *Positive polynomials in control*, p. 579–579, 2005.

[27] BOYD, S.; EL GHAOUI, L.; FERON, E.; BALAKRISHNAN, V. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[28] SCHERER, C.; WEILAND, S. Linear matrix inequalities in control. *Lecture Notes, Dutch Institute for Systems and Control, Delft, The Netherlands*, v. 3, 2000.

[29] LAVAEI, J.; LOW, S. H. Zero duality gap in optimal power flow problem. *IEEE Transactions on Power Systems*, v. 27, n. 1, p. 92–107, 2012.

[30] VANDENBERGHE, L.; BOYD, S.; EL GAMAL, A. Optimizing dominant time constant in rc circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 17, n. 2, p. 110–125, 1998.

[31] VANDENBERGHE, L.; BOYD, S.; EL GAMAL, A. Optimal wire and transistor sizing for circuits with non-tree topology. In: . c1997. p. 252–259.

[32] BEN-TAL, A.; NEMIROVSKI, A. Robust truss topology design via semidefinite programming. *SIAM journal on optimization*, v. 7, n. 4, p. 991–1016, 1997.

[33] BEN-TAL, A.; NEMIROVSKI, A. Robust convex optimization. *Mathematics of operations research*, v. 23, n. 4, p. 769–805, 1998.

[34] ZYMLER, S.; KUHN, D.; RUSTEM, B. Distributionally robust joint chance constraints with second-order moment information. *Mathematical Programming*, p. 1–32, 2013.

[35] LOVÁSZ, L.; SCHRIJVER, A. Cones of matrices and set-functions and 0–1 optimization. *SIAM Journal on Optimization*, v. 1, n. 2, p. 166–190, 1991.

[36] GOEMANS, M. X.; WILLIAMSON, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, v. 42, n. 6, p. 1115–1145, 1995.

[37] KARLOFF, H.; ZWICK, U. A 7/8-approximation algorithm for max 3sat? In: . c1997. p. 406–415.

[38] CVETKOVIĆ, D.; ČANGALOVIĆ, M.; KOVAČEVIĆ-VUJČIĆ, V. Semidefinite programming methods for the symmetric traveling salesman problem. In: . c1999. p. 126–136.

[39] CANDES, E. J.; ELDAR, Y. C.; STROHMER, T.; VORONINSKI, V. Phase retrieval via matrix completion. *SIAM review*, v. 57, n. 2, p. 225–251, 2015.

[40] LOVÁSZ, L. Semidefinite programs and combinatorial optimization. In: *Recent advances in algorithms and combinatorics*. Springer, 2003. p. 137–194.

[41] DE BIE, T. Deploying sdp for machine learning. In: . c2007. p. 205–210.

[42] CANDES, E.; RECHT, B. Exact matrix completion via convex optimization. *Communications of the ACM*, v. 55, n. 6, p. 111–119, 2012.

[43] BENNETT, J.; LANNING, S. et al. The netflix prize. In: . c2007. v. 2007. p. 35.

[44] WAINWRIGHT, M. J.; JORDAN, M. I. Log-determinant relaxation for approximate inference in discrete markov random fields. *IEEE Transactions on Signal Processing*, v. 54, n. 6, p. 2099–2109, 2006.

[45] LANCKRIET, G. R.; CRISTIANINI, N.; BARTLETT, P.; GHAOUI, L. E.; JORDAN, M. I. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, v. 5, n. Jan, p. 27–72, 2004.

[46] LANCKRIET, G. R.; DE BIE, T.; CRISTIANINI, N.; JORDAN, M. I.; NOBLE, W. S. A statistical framework for genomic data fusion. *Bioinformatics*, v. 20, n. 16, p. 2626–2635, 2004.

[47] KHOT, S. On the power of unique 2-prover 1-round games. In: . c2002. p. 767–775.

[48] KHOT, S.; MINZER, D.; SAFRA, M. Pseudorandom sets in grassmann graph have near-perfect expansion. In: . c2018. v. 25.

[49] DINUR, I.; KHOT, S.; KINDLER, G.; MINZER, D.; SAFRA, M. Towards a proof of the 2-to-1 games conjecture? In: . c2018. p. 376–389.

[50] SHOR, N. Z. Cut-off method with space extension in convex programming problems. *Cybernetics and systems analysis*, v. 13, n. 1, p. 94–96, 1977.

[51] IUDIN, D.; NEMIROVSKII, A. S. Informational complexity and efficient methods for solving complex extremal problems. *Matekon*, v. 13, n. 3, p. 25–45, 1977.

[52] HIRIART-URRUTY, J.-B.; LEMARÉCHAL, C. *Convex analysis and minimization algorithms i: Fundamentals*. Springer science & business media, 2013. v. 305.

[53] KARMARKAR, N. A new polynomial-time algorithm for linear programming. In: . c1984. p. 302–311.

[54] ADLER, I.; RESENDE, M. G.; VEIGA, G.; KARMARKAR, N. An implementation of karmarkar's algorithm for linear programming. *Mathematical programming*, v. 44, n. 1, p. 297–335, 1989.

[55] NESTEROV, Y.; NEMIROVSKII, A. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[56] ALIZADEH, F. Optimization over the positive-definite cone: interior point methods and combinatorial applications. *Advances in optimization and parallel computing*, 1992.

[57] HELMBERG, C.; RENDL, F.; VANDERBEI, R. J.; WOLKOWICZ, H. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, v. 6, n. 2, p. 342–361, 1996.

[58] ALIZADEH, F.; HAEBERLY, J.-P. A.; OVERTON, M. L. Primal-dual interior-point methods for semidefinite programming: convergence rates, stability and numerical results. *SIAM Journal on Optimization*, v. 8, n. 3, p. 746–768, 1998.

[59] BORCHERS, B. Csdp, ac library for semidefinite programming. *Optimization methods and Software*, v. 11, n. 1-4, p. 613–623, 1999.

[60] MOSEK, A. The mosek optimization software. *Online at http://www. mosek. com*, v. 54, p. 2–1, 2010.

[61] BOYD, S.; PARIKH, N.; CHU, E.; PELEATO, B.; ECKSTEIN, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, v. 3, n. 1, p. 1–122, 2011.

[62] HEIDE, F.; DIAMOND, S.; NIESSNER, M.; RAGAN-KELLEY, J.; HEIDRICH, W.; WETZSTEIN, G. Proximal: Efficient image optimization using proximal algorithms. *ACM Transactions on Graphics (TOG)*, v. 35, n. 4, p. 84, 2016.

[63] O'DONOGHUE, B.; CHU, E.; PARIKH, N.; BOYD, S. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, v. 169, n. 3, p. 1042–1068, 2016.

[64] MADANI, R.; KALBAT, A.; LAVAEI, J. Admm for sparse semidefinite programming with applications to optimal power flow problem. In: . c2015. p. 5932–5939.

[65] FUKUDA, M.; KOJIMA, M.; MUROTA, K.; NAKATA, K. Exploiting sparsity in semidefinite programming via matrix completion i: General framework. *SIAM Journal on Optimization*, v. 11, n. 3, p. 647–674, 2001.

[66] NAKATA, K.; FUJISAWA, K.; FUKUDA, M.; KOJIMA, M.; MUROTA, K. Exploiting sparsity in semidefinite programming via matrix completion ii: Implementation and numerical results. *Mathematical Programming*, v. 95, n. 2, p. 303–327, 2003.

[67] VANDENBERGHE, L.; ANDERSEN, M. S. et al. Chordal graphs and semidefinite optimization. *Foundations and Trends® in Optimization*, v. 1, n. 4, p. 241–433, 2015.

[68] FUJISAWA, K.; SATO, H.; MATSUOKA, S.; ENDO, T.; YAMASHITA, M.; NAKATA, M. High-performance general solver for extremely large-scale semidefinite programming problems. In: . c2012. p. 1–11.

[69] ZHANG, R. Y.; LAVAEI, J. Sparse semidefinite programs with near-linear time complexity. *arXiv preprint arXiv:1710.03475*, 2017.

[70] ZHANG, R. Y.; LAVAEI, J. Modified interior-point method for large-and-sparse low-rank semidefinite programs. *arXiv preprint arXiv:1703.10973*, 2017.

[71] ECKSTEIN, J. *Splitting methods for monotone operators with applications to parallel optimization*. 1989. Tese (Doutorado em Física) - Massachusetts Institute of Technology, 1989.

[72] COMBETTES*, P. L. Solving monotone inclusions via compositions of nonexpansive averaged operators. *Optimization*, v. 53, n. 5-6, p. 475–504, 2004.

[73] COMBETTES, P. L.; WAJS, V. R. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, v. 4, n. 4, p. 1168–1200, 2005.

[74] BAUSCHKE, H. H.; COMBETTES, P. L. et al. *Convex analysis and monotone operator theory in hilbert spaces*. Springer, 2017. v. 2011.

[75] POCK, T.; CREMERS, D.; BISCHOF, H.; CHAMBOLLE, A. An algorithm for minimizing the mumford-shah functional. In: . c2009. p. 1133–1140.

[76] SIDKY, E. Y.; JØRGENSEN, J. H.; PAN, X. Convex optimization problem prototyping for image reconstruction in computed tomography with the chambolle–pock algorithm. *Physics in medicine and biology*, v. 57, n. 10, p. 3065, 2012.

[77] VAITER, S.; PEYRÉ, G.; DOSSAL, C.; FADILI, J. Robust sparse analysis regularization. *IEEE Transactions on information theory*, v. 59, n. 4, p. 2001–2016, 2013.

[78] HARDY, G. H.; LITTLEWOOD, J. E.; PÓLYA, G. et al. *Inequalities*. Cambridge university press, 1988.

[79] GOLDSTEIN, T.; LI, M.; YUAN, X.; ESSER, E.; BARANIUK, R. Adaptive primal-dual hybrid gradient methods for saddle-point problems. *arXiv preprint arXiv:1305.0546*, 2013.

[80] TSENG, P. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization*, v. 29, n. 1, p. 119–138, 1991.

[81] GLOWINSKI, R.; MARROCO, A. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, v. 9, n. R2, p. 41–76, 1975.

[82] ECKSTEIN, J.; BERTSEKAS, D. P. On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, v. 55, n. 1-3, p. 293–318, 1992.

[83] BARVINOK, A. I. Problems of distance geometry and convex properties of quadratic maps. *Discrete & Computational Geometry*, v. 13, n. 2, p. 189–202, 1995.

[84] PATAKI, G. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, v. 23, n. 2, p. 339–358, 1998.

[85] ZHANG, Y.; LU, Z. Penalty decomposition methods for rank minimization. In: . c2011. p. 46–54.

[86] ZHAO, Y.-B. An approximation theory of matrix rank minimization and its application to quadratic equations. *Linear Algebra and its Applications*, v. 437, n. 1, p. 77–93, 2012.

[87] YUAN, G.; GHANEM, B. A proximal alternating direction method for semi-definite rank minimization. In: . c2016. p. 2300–2308.

[88] BURER, S.; MONTEIRO, R. D. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, v. 95, n. 2, p. 329–357, 2003.

[89] SHAH, S.; YADAV, A. K.; CASTILLO, C. D.; JACOBS, D. W.; STUDER, C.; GOLDSTEIN, T. Biconvex relaxation for semidefinite programming in computer vision. In: . c2016. p. 717–735.

[90] WANG, P.-W.; CHANG, W.-C.; KOLTER, J. Z. The mixing method: coordinate descent for low-rank semidefinite programming. *arXiv preprint arXiv:1706.00476*, 2017.

[91] YURTSEVER, A.; UDELL, M.; TROPP, J. A.; CEVHER, V. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. *arXiv preprint arXiv:1702.06838*, 2017.

[92] HALKO, N.; MARTINSSON, P.-G.; TROPP, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, v. 53, n. 2, p. 217–288, 2011.

[93] TROPP, J. A.; YURTSEVER, A.; UDELL, M.; CEVHER, V. Practical sketching algorithms for low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, v. 38, n. 4, p. 1454–1485, 2017.

[94] ECKART, C.; YOUNG, G. The approximation of one matrix by another of lower rank. *Psychometrika*, v. 1, n. 3, p. 211–218, 1936.

[95] GOLUB, G. H.; VAN LOAN, C. F. *Matrix computations*. JHU Press, 2012. v. 3.

[96] HIGHAM, N. J. *Matrix nearness problems and applications*. University of Manchester. Department of Mathematics, 1988.

[97] LEHOUCQ, R. B.; SORENSEN, D. C.; YANG, C. *Arpack users' guide: solution of large-scale eigenvalue problems with implicitly restarted arnoldi methods*. Siam, 1998. v. 6.

[98] ANDERSON, E.; BAI, Z.; BISCHOF, C.; BLACKFORD, L. S.; DEMMEL, J.; DONGARRA, J.; DU CROZ, J.; GREENBAUM, A.; HAMMARLING, S.; MCKENNEY, A. et al. *Lapack users' guide*. SIAM, 1999.

[99] HE, B.; YUAN, X. Convergence analysis of primal-dual algorithms for total variation image restoration. *Rapport technique, Citeseer*, 2010.

[100] BEZANSON, J.; EDELMAN, A.; KARPINSKI, S.; SHAH, V. B. Julia: A fresh approach to numerical computing. *SIAM review*, v. 59, n. 1, p. 65–98, 2017.

[101] LAWSON, C. L.; HANSON, R. J.; KINCAID, D. R.; KROGH, F. T. Basic linear algebra subprograms for fortran usage. *ACM Transactions on Mathematical Software (TOMS)*, v. 5, n. 3, p. 308–323, 1979.

[102] ANDERSON, E.; BAI, Z.; DONGARRA, J.; GREENBAUM, A.; MCKENNEY, A.; DU CROZ, J.; HAMMARLING, S.; DEMMEL, J.; BISCHOF, C.; SORENSEN, D. Lapack: A portable linear algebra library for high-performance computers. In: . c1990. p. 2–11.

[103] DUNNING, I.; HUCHETTE, J.; LUBIN, M. Jump: A modeling language for mathematical optimization. *SIAM Review*, v. 59, n. 2, p. 295–320, 2017.

[104] CHISTOV, A. L.; GRIGOR'EV, D. Y. Complexity of quantifier elimination in the theory of algebraically closed fields. In: . c1984. p. 17–31.

[105] ECKSTEIN, J.; FERRIS, M. C. Operator-splitting methods for monotone affine variational inequalities, with a parallel application to optimal control. *INFORMS Journal on Computing*, v. 10, n. 2, p. 218–235, 1998.

[106] WANG, E.; ZHANG, Q.; SHEN, B.; ZHANG, G.; LU, X.; WU, Q.; WANG, Y. Intel math kernel library. In: *High-Performance Computing on the Intel® Xeon Phi™*. Springer, 2014. p. 167–188.

[107] KARISCH, S. E.; RENDL, F.; CLAUSEN, J. Solving graph bisection problems with semidefinite programming. *INFORMS Journal on Computing*, v. 12, n. 3, p. 177–191, 2000.

[108] BORCHERS, B. Sdplib 1.2, a library of semidefinite programming test problems. *Optimization Methods and Software*, v. 11, n. 1-4, p. 683–690, 1999.

[109] ALFAKIH, A. Y.; KHANDANI, A.; WOLKOWICZ, H. Solving euclidean distance matrix completion problems via semidefinite programming. *Computational optimization and applications*, v. 12, n. 1-3, p. 13–30, 1999.

[110] SO, A. M.-C.; YE, Y. Theory of semidefinite programming for sensor network localization. *Mathematical Programming*, v. 109, n. 2-3, p. 367–384, 2007.

[111] JALDÉN, J.; MARTIN, C.; OTTERSTEN, B. Semidefinite programming for detection in linear systems-optimality conditions and space-time decoding. In: . c2003. v. 4. p. IV–9.

[112] JALDÉN, J.; OTTERSTEN, B. The diversity order of the semidefinite relaxation detector. *IEEE Transactions on Information Theory*, v. 54, n. 4, p. 1406–1422, 2008.

[113] VERDÚ, S. Computational complexity of optimum multiuser detection. *Algorithmica*, v. 4, n. 1, p. 303–312, 1989.

[114] DONG, H. Relaxing nonconvex quadratic functions by multiple adaptive diagonal perturbations. *SIAM Journal on Optimization*, v. 26, n. 3, p. 1962–1985, 2016.