FIRST-ORDER METHODS FOR SEMIDEFINITE PROGRAMMING

Zaiwen Wen

Advisor: Professor Donald Goldfarb

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2009

ABSTRACT

First-order Methods For Semidefinite Programming

Zaiwen Wen

Semidefinite programming (SDP) problems are concerned with minimizing a linear function of a symmetric positive definite matrix subject to linear equality constraints. These convex problems are solvable in polynomial time by interior point methods. However, if the number of constraints m in an SDP is of order $O(n^2)$ when the unknown positive semidefinite matrix is $n \times n$, interior point methods become impractical both in terms of the time ($O(n^6)$) and the amount of memory ($O(m^2)$) required at each iteration to form the $m \times m$ positive definite Schur complement matrix M and compute the search direction by finding the Cholesky factorization of M. This significantly limits the application of interior-point methods. In comparison, the computational cost of each iteration of first-order optimization approaches is much cheaper, particularly, if any sparsity in the SDP constraints or other special structure is exploited. This dissertation is devoted to the development, analysis and evaluation of two first-order approaches that are able to solve large SDP problems which have been challenging for interior point methods.

In chapter 2, we present a row-by-row (RBR) method based on solving a sequence of problems obtained by restricting the n-dimensional positive semidefinite constraint on the matrix X. By fixing any $(n-1)$-dimensional principal submatrix of X and using its (generalized) Schur complement, the positive semidefinite constraint is reduced to a simple second-order cone constraint. When the RBR method is applied to solve the max-

cut SDP relaxation, the optimal solution of the RBR subproblem only involves a single matrix-vector product which leads to a simple and very efficient method. To handle linear constraints in generic SDP problems, we use an augmented Lagrangian approach. Specialized versions are presented for the maxcut SDP relaxation and the minimum nuclear norm matrix completion problem since closed-form solutions for the RBR subproblems are available. Numerical results on the maxcut SDP relaxation and matrix completion problems are presented to demonstrate the robustness and efficiency of our algorithm.

In chapter 3, we present an alternating direction method based on the augmented Lagrangian framework for solving SDP problems in standard form. At each iteration, the algorithm, also known as a two-splitting scheme, minimizes the dual augmented Lagrangian function sequentially with respect to the Lagrange multipliers corresponding to the linear constraints, then the dual slack variables and finally the primal variables, while in each minimization keeping the other variables fixed. Convergence is proved by using a fixed-point argument. A multiple-splitting algorithm is then proposed to handle SDPs with inequality constraints and positivity constraints directly without transforming them to the equality constraints in standard form. Numerical results on frequency assignment, maximum stable set and binary integer quadratic programming problems, show that our algorithm is very promising.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my deep and sincere gratitude to my advisor Professor Don Goldfarb for his generous support of my study and research, immense patience, understanding, warm encouragement and inspiration. His expertise in mathematical programming and operations research greatly improved my research skills and prepared me for future challenges. His optimistic personality and dedication to research has had great and positive influence on my personality and attitude. His pursuit of simplicity pulled me out of many traps of confusion and chaos encountered in cracking and presenting complicated topics. I am also grateful for his correction of my English pronunciation and his revisions of my usage of the English language in my papers. Working with Don has been a stimulating, fun and incredibly enriching experience.

My keen appreciation goes to Professors Cliff S. Stein, Daniel Bienstock, Michael L. Overton and Farid Alizadeh for serving on my thesis committee and their valuable comments on my dissertation. I would like to thank Professors Daniel Bienstock and Garud Iyengar for the their excellent guidance in integer and robust optimization. I wish to thank Professor Rama Cont for teaching me inverse problems in financial engineering. My sincere thanks goes to Professors Ward Whitt and David Yao for their consistent encouragement. I am indebted to Professor Yin Zhang for his valuable advise in compressive sensing and semidefinite programming.

I would like to acknowledge many previous and current members of Don's research group, including Katya Scheinberg, Wotao Yin, Lifeng Chen and Shiqian Ma, who collab-

orated with Don and myself on a broad range of research topics, and thank them for many insightful and fruitful discussions on optimization. I wish to extend my heartfelt thanks to Wotao for sharing ideas and providing support during the ups and downs.

Special thanks are due to the Department staff members Jenny Mak, Michael Mostow, Jaya Mohanty, Donella Crosgnach, Risa Cho, Ufei Chan and Maria Casuscelli for producing a great atmosphere at work. I am very grateful to doctoral students/friends Wotao, Nan, Wanmo, Lin, Zhiyong, Ming, Lifeng, Ning, Anuj, Xianhua, Kachun, Zongjian, Thiam Hui, Abhinav, Ohad, Kun Soo, Masoud, Rouba, Rishi, Vijay, Ruxian, Shiqian, Yunan, Yiping, Yu Hang, Guodong, Yixi, Pingkai, John and many others too numerous to mention, for pleasant and fruitful discussions, and for sharing different ideas.

Going back to my time at the Institute of Computational Mathematics and Scientific / Engineering Computing of the Chinese Academy of Sciences, I am grateful to my lifelong advisor Professor Yaxiang Yuan, who introduced the broad topic of nonlinear programming to me and provided continuous help throughout the course of my thesis work. I also wish to thank Professor Yanfei Wang, for teaching me many topics in inverse problems. They were very generous with their time, and were instrumental in the development of my interest in research.

I am indebted to my family. My parents Shaoshu Wen and Qizhi Tan are pure and simple farmers in a small village close to the beautiful Dongting Lake. Their hardworking spirit always reminds me to conquer any difficulty patiently and peacefully. My brother Shouwen Wen has been the major source of support of the family over the years. Without his help and guidance, I would not have been able to finish my undergraduate and graduate

study in China and had the opportunity to come to Columbia to pursue my professional goal.

I owe my deepest gratitude to my wife Keqi for sacrificing her job as a lecturer in a prestigious college in Beijing three years ago and for her love, unwavering support and delightful company in New York City. She helped me to concentrate on completing this dissertation. Without her help and encouragement, this study would not have been completed. Finally, I would like to dedicate my dissertation to our new-born baby Yanyue and wish joy and happiness will always be with her.

Zaiwen Wen

August 12, 2009

To my family

Chapter 1

Introduction

1.1 Background and Motivation

Semidefinite programming (SDP) problems are concerned with minimizing a linear function of a symmetric positive semidefinite matrix subject to linear equality constraints. Although the positive semidefinite constraint is nonlinear and nonsmooth, these convex problems are solvable in polynomial time by interior point methods [52, 56, 59]. Consequently, SDPs have been a very successful and powerful tool for modeling many applications arising from combinatorial optimization, nonconvex quadratic programming, eigenvalue and nonconvex optimization, systems and control theory, structural design, matrix completion problems, and problems in statistics and engineering since the late 1980s and 1990s.

Unfortunately, however, in practice large scale SDPs are quite difficult to solve because of the very large amount of work required by each iteration of an interior point method. Most of these methods form a positive definite $m \times m$ matrix M, where m is the number

of constraints in the SDP, and then compute the search direction by finding the Cholesky factorization of M. Since m can be $O(n^2)$ when the unknown positive semidefinite matrix is $n \times n$, it can take $O(n^6)$ arithmetic operations to do this. Consequently, this becomes impractical both in terms of the time and the amount of memory required ($O(m^2)$) when n is much larger than one hundred and m is much larger than a few thousand. Moreover forming M itself can be prohibitively expensive unless m is not too large or the constraints in the SDP are very sparse [23].

On the contrary, although the computational complexity of the first-order optimization methods is not polynomial, each of their iterations can be executed much more cheaply than in an interior point algorithm. This enables them to solve very large SDPs efficiently. In particular, first-order augmented Lagrangian approaches have been proposed for both the primal and dual formulations of SDPs, and different ways have been used to minimize the augmented Lagrangian function depending on how the positive semidefinite constraints are handled. In [11, 12], the positive definite variable X is replaced by $RR^\top$ in the primal augmented Lagrangian function, where R is a low rank matrix, and then nonlinear programming approaches are used. In [64], the positive semidefinite constraint is represented implicitly by using a projection operator and a semismooth Newton approach combined with the conjugate gradient method is proposed to minimize the dual augmented Lagrangian function. The regularization methods [42] (and the related boundary point method [46]) are also based on a dual augmented Lagrangian approach and they use an eigenvalue decomposition to maintain complementarity. In [9, 14], a coordinate descent method and eigenvalue decomposition are used to minimize the primal augmented Lagrangian function. A gen-

eralized version of the augmented Lagrangian method is used in [37] by constructing a special penalty function for matrix inequalities.

This dissertation is devoted to the development, analysis and evaluation of two first-order approaches based on an augmented Lagrangian framework. In chapter 2, we present a row-by-row (RBR) method. By fixing any $(n-1)$-dimensional principal submatrix of X and using its Schur complement, the positive semidefinite constraint is reduced to a simple second-order cone constraint and then a sequence of second-order cone programming problems constructed from the primal augmented Lagrangian function are minimized. Specialized versions are presented for the maxcut SDP relaxation and the minimum nuclear norm matrix completion problem since closed-form solutions for the RBR subproblems are available. In particular, when the RBR method is applied to solve the maxcut SDP relaxation, the optimal solution of the RBR subproblem only involves a single matrix-vector product which leads to a simple and very efficient method.

In chapter 3, we first present an alternating direction method for solving SDP problems in standard form. At each iteration, the algorithm, also known as a two-splitting scheme, minimizes the dual augmented Lagrangian function sequentially, first with respect to the Lagrange multipliers corresponding to the linear constraints, and then with respect to the dual slack variables while keeping the other variables fixed, after which it updates the primal variables as in a standard augmented Lagrangian method. A multiple-splitting algorithm is then proposed to handle SDPs with inequality constraints and positivity constraints directly without transforming them to the equality constraints in standard form.

1.2 Preliminaries

1.2.1 Basic Notations

We denote the set of real numbers by $\mathbb{R}$ and the set of vectors with n components in $\mathbb{R}$ by $\mathbb{R}^n$. The superscript "$\top$" denotes the transpose operation. The inner product between vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ is defined as

$$\langle x, y \rangle := x^\top y = \sum_{j=1}^{n} x_j y_j.$$

The ℓ_2-norm of a vector x is given by $\|x\|_2 = \sqrt{x^\top x}$. The vector $\begin{pmatrix} x \\ y \end{pmatrix}$ obtained by stacking the vector $x \in \mathbb{R}^p$ on the top of the vector $y \in \mathbb{R}^q$ is also denoted by $[x; y] \in \mathbb{R}^{p+q}$.

The set of real matrices with m rows and n columns is denoted by $\mathbb{R}^{m \times n}$. The operator norm $\|A\|_2$ of $A \in \mathbb{R}^{m \times n}$ is defined as

$$\|A\|_2 := \max_{0 \neq x \in \mathbb{R}^n} \frac{\|Ax\|_2}{\|x\|_2}.$$

The notation $X \geq 0$ ($X > 0$) means that every component of the matrix X is nonnegative (positive). A matrix $X \in \mathbb{R}^{n \times n}$ is said to be positive definite (semidefinite), denoted by $X \succ 0$ ($X \succeq 0$), if $y^\top X y > 0$ ($y^\top X y \geq 0$), for all $0 \neq y \in \mathbb{R}^n$.

The set of $n \times n$ symmetric matrices is denoted by S^n and the set of $n \times n$ symmetric positive semidefinite (positive definite) matrices is denoted by S^n_+ (S^n_{++}). The trace of X,

i.e., the sum of the diagonal elements of X, is denoted by $\mathbf{Tr}(X)$. The inner product between two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times n}$ is defined as

$$\langle A, B\rangle := \sum_{jk} A_{j,k}B_{j,k} = \mathbf{Tr}(A^\top B).$$

The Frobenius norm of a matrix $A \in \mathbb{R}^{m\times n}$ is defined as $\|A\|_F := \sqrt{\sum_{i,j} A_{i,j}^2}$. The identity matrix is denoted by I. We write $\mathrm{diag}(X)$ for the vector of diagonal entries of $X \in S^n$, and $\mathrm{diag}(x)$ for the diagonal matrix with the vector $x \in \mathbb{R}^n$ on its diagonal. Let $\mathrm{vec}(X)$ be a vector that contains the columns of the matrix X, stacked each on top of the next in the order that they appear in the matrix, and $\mathrm{mat}(x)$ be a matrix X such that $x = \mathrm{vec}(X)$.

1.2.2 Semidefinite Programming

In this dissertation, we mainly consider the standard form SDP:

$$\begin{aligned} \min_{X\in S^n} \quad & \langle C, X\rangle \\ \text{s.t.} \quad & \mathcal{A}(X) = b, \quad X \succeq 0, \end{aligned} \tag{1.2.1}$$

where the linear map $\mathcal{A}(\cdot) : S^n \to \mathbb{R}^m$ is defined as

$$\mathcal{A}(X) := \left(\left\langle A^{(1)}, X\right\rangle, \cdots, \left\langle A^{(m)}, X\right\rangle\right)^\top, \tag{1.2.2}$$

the matrices $C, A^{(i)} \in S^n$, $i = 1, \cdots, m$, and the vector $b \in \mathbb{R}^m$ are given, and the unknown matrix $X \in S^n_+$. Note that the equation $\mathcal{A}(X) = b$ is equivalent to $A\,\mathrm{vec}(X) = b$, where

$$A := \left(\mathrm{vec}(A^{(1)}), \cdots, \mathrm{vec}(A^{(m)}) \right)^\top \in \mathbb{R}^{m \times n^2}.$$

Furthermore, we introduce the following three operators related to $\mathcal{A}(\cdot)$. The adjoint operator $\mathcal{A}^* : \mathbb{R}^m \to S^n$ of $\mathcal{A}$ is defined as

$$\mathcal{A}^*(y) := \sum_{i=1}^{m} y_i A^{(i)} = \mathrm{mat}(A^\top y).$$

The operator $\mathcal{A}\mathcal{A}^* : \mathbb{R}^m \to \mathbb{R}^m$ is defined as

$$\mathcal{A}(\mathcal{A}^*(y)) = (AA^\top)y$$

and the operator $\mathcal{A}^*\mathcal{A} : S^n \to S^n$ is defined as

$$\mathcal{A}^*(\mathcal{A}(X)) = \mathrm{mat}\left(\left(A^\top A \right) \mathrm{vec}(X) \right).$$

The dual problem associated with (1.2.1) is

$$\begin{aligned} \max_{y \in \mathbb{R}^m, S \in S^n} \quad & b^\top y \\ \text{s.t.} \quad & \mathcal{A}^*(y) + S = C, \quad S \succeq 0. \end{aligned} \tag{1.2.3}$$

We denote the feasible sets for the primal (1.2.1) and dual (1.2.3) by $\mathcal{F}_p$ and $\mathcal{F}_d$, respec-

tively; that is,

$$\begin{aligned} \mathcal{F}_p &:= \{X \in S^n \mid \mathcal{A}(X) = b, X \succeq 0\}, \\ \mathcal{F}_d &:= \{(y,S) \mid \mathcal{A}^*(y) + S = C, S \succeq 0\}. \end{aligned}$$

The duality gap, which is defined as the difference between the primal and dual objective values at the feasible solutions of (1.2.1) and (1.2.3), is nonnegative.

Lemma 1.2.1. *(weak duality, [17,52,59]) For any $X \in \mathcal{F}_p$ and $(y,S) \in \mathcal{F}_d$,*

$$\mathbf{Tr}(CX) - b^\top y = \mathbf{Tr}(XS) \geq 0.$$

Corollary 1.2.2. *Let $X \in \mathcal{F}_p$, $(y,S) \in \mathcal{F}_d$, and $\langle X,S \rangle = 0$. Then (X,y,S) is optimal for problems* (1.2.1) *and* (1.2.3)*.*

The optimal objective values of (1.2.1) and (1.2.3) are not necessary equal for every SDP problem, and even if they are, the optimal solution of either (1.2.1) or (1.2.3) may not be attainable. The following theorem states sufficient conditions for strong duality to hold.

Theorem 1.2.3. *(Strong duality, [17, 52, 59]) Let $\mathcal{F}_p$ and $\mathcal{F}_d$ be nonempty and have an interior feasible solution. Then X is optimal for* (1.2.1) *if and only if $X \in \mathcal{F}_p$ and there exists $(y,S) \in \mathcal{F}_d$ such that $\langle C,X \rangle = b^\top y$ or $\mathbf{Tr}(XS) = 0$.*

We conclude this subsection by summarizing the SDP duality as follows.

Theorem 1.2.4. *(SDP duality therorem, [17,52,59])*

i) *If* (1.2.1) *and* (1.2.3) *both have feasible solutions, then both problems have optimal solutions and the optimal objective values may have a duality gap;*

ii) *If one of* (1.2.1) *and* (1.2.3) *has a strictly or interior feasible solution and has a finite optimal value, then the other is feasible and has the same optimal value;*

iii) *If one of* (1.2.1) *and* (1.2.3) *is unbounded, then other has no feasible solution.*

1.2.3 Augmented Lagrangian method

The augmented Lagrangian method (or the method of multipliers) was proposed originally for solving nonlinear programming problems [43, 50]. By introducing explicit Lagrange multiplier estimates to a quadratic penalty function, these methods combine both the properties of the Lagrangian function and the quadratic penalty function and often yield less ill-conditioned subproblems than the quadratic penalty method.

We now briefly introduce an augmented Lagrangian method for solving the primal SDP (1.2.1). By taking into consideration only the general linear constraints $\mathcal{A}(X) = b$, the augmented Lagrangian function is defined as:

$$\mathcal{L}(X, \pi, \mu) := \langle C, X \rangle - \pi^\top (\mathcal{A}(X) - b) + \frac{1}{2\mu} \|\mathcal{A}(X) - b\|_2^2, \tag{1.2.4}$$

where $\pi \in \mathbb{R}^m$ is the vector of Lagrangian multipliers and $\mu > 0$ is the penalty parameter. Starting from $\pi^1 = \mathbf{0}$ and $\mu^1 \in (0, +\infty)$ at the first iteration, our augmented Lagrangian

method iteratively solves at the k-th iteration

$$X^k := \arg\min_X \mathcal{L}(X, \pi^k, \mu^k), \quad \text{s.t. } X \succeq 0, \tag{1.2.5}$$

chooses $\mu^{k+1} \in [\gamma\mu^k, \mu^k]$, where $\gamma \in (0,1)$, and then updates the vector of Lagrange multipliers by

$$\pi^{k+1} := \pi^k - \frac{\mathcal{A}(X^k) - b}{\mu^k}. \tag{1.2.6}$$

The following Slater condition for (1.2.1) is assumed to hold throughout this dissertation.

Assumption 1.2.5. *Problem* (1.2.1) *satisfies the Slater condition:*

$$\begin{cases} \mathcal{A} : S^n \to \mathbb{R}^m \text{ is onto }, \\ \exists X^1 \in S^n_{++} \text{ such that } \mathcal{A}(X^1) = b. \end{cases} \tag{1.2.7}$$

The convergence of our augmented Lagrangian framework follows from the standard theory for the augmented Lagrangian method for minimizing a strictly convex function subject to linear equality constraints [4, 5, 48]. For completeness of this dissertation, we present the following convergence result and provide its proof in Appendix A1.

Theorem 1.2.6. *Suppose that the Slater condition* (1.2.7) *holds, then any limit point of the sequence* $\{X^k\}$ *generated by the iterative procedure* (1.2.5)-(1.2.6) *is a global minimizer of problem* (1.2.1).

Chapter 2

Row by row methods for SDP

2.1 Introduction

In this chapter we present a new first-order method for solving semidefinite programming problems. Although the computational complexity of the new method presented here is not polynomial, each of its iterations can be executed much more cheaply than in an interior point algorithm. This enables it to solve very large SDPs efficiently. Preliminary numerical testing verifies this. For example, variants of our new method produce highly accurate solutions to maxcut SDP relaxation problems involving matrices of size 4000×4000 in less than 5.25 minutes and nuclear norm matrix completion SDPs involving matrices of size 1000×1000 in less than 1 minute on a 3.4 GHZ workstation. If only moderately accurate solutions are required (i.e., a relative accuracy of the order of 10^{-3}) then less than 45 and 10 seconds, respectively, is needed.

Our method is based upon the well known relationship between the positive semidefi-

niteness of a symmetric matrix and properties of the Schur complement of a sub-matrix of that matrix [62]. We note that Schur complements play an important role in semidefinite programming and related optimization problems. For example, they are often used to formulate problems as SDPs [8, 56]. In [1, 26, 27] they are used to reformulate certain SDP problems as second-order cone programs (SOCPs). More recently, they have been used by Banerjee, El Ghaoui and d'Aspremont [2] to develop a method for solving a maximum likelihood estimation problem whose objective function involves the log determinant of a positive semidefinite matrix. That method is closely related to the method proposed here.

As in the method proposed in Banerjee et. al [2], we use Schur complements to develop an *overlapping* block coordinate descent method. The coordinates (i.e., variables) in each iteration of our method correspond to the components of a single row (column) of the unknown semidefinite matrix. Since every row (column) of a symmetric matrix contains one component of each of the other rows (columns), the blocks in our method overlap. The convergence properties of block coordinate descent methods [6, 28, 29, 40, 55], which are also referred to as nonlinear Gauss-Seidel methods and subspace correction methods [51], have been intensively studied. In the case of (single) coordinate descent, convergence is not guaranteed for nonconvex functions [47]. Convergence can be proved under fairly mild conditions when the objective function is convex. As we shall see below, the convergence result in [6] can be extended to the case of overlapping blocks. However, they do not apply to the case where constraints couple the variables between different blocks.

2.1.1 Preliminaries

Our row-by-row (**RBR**) method is motivated by the following fact about the Schur complement of a positive definite matrix. Assume the matrix $X \in S^n$ can be partitioned as

$$X := \begin{pmatrix} \xi & y^\top \\ y & B \end{pmatrix}, \tag{2.1.1}$$

where $\xi \in \mathbb{R}$, $y \in \mathbb{R}^{n-1}$ and $B \in S^{n-1}$ is nonsingular. Since X can be factorized as

$$X = \begin{pmatrix} 1 & y^\top B^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} \xi - y^\top B^{-1} y & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} 1 & 0 \\ B^{-1} y & I \end{pmatrix}, \tag{2.1.2}$$

the positive definite constraint $X \succ 0$ is equivalent to

$$X \succ 0 \Longleftrightarrow B \succ 0 \text{ and } (X/B) := \xi - y^\top B^{-1} y > 0, \tag{2.1.3}$$

where (X/B) is called the Schur complement of X with respect to B. If the matrix B is singular, the generalized Schur complement of B in X is defined as

$$(X/B) := \xi - y^\top B^\dagger y, \tag{2.1.4}$$

where $B^\dagger$ is the Moore-Penrose pseudo-inverse of B and the following theorem generalizes (2.1.3).

Theorem 2.1.1. *([62], Theorems 1.12 and 1.20) Let $X \in S^n$ be a symmetric matrix parti-*

tioned as (2.1.1) *in which* ξ *is a scalar and* $B \in S^{n-1}$. *The generalized Schur complement of* B *in* X *is defined as* $(X/B) := \xi - y^\top B^\dagger y$, *where* $B^\dagger$ *is the pseudo-inverse of* B. *Then the following holds.*

1) If B *is nonsingular, then* $X \succ 0$ *if and only if both* $B \succ 0$ *and* $(X/B) > 0$.

2) If B *is nonsingular, then* $X \succeq 0$ *if and only if both* $B \succ 0$ *and* $(X/B) \geq 0$.

3) $X \succeq 0$ *if and only if* $B \succeq 0$, $(X/B) \geq 0$ *and* $y \in \mathcal{R}(B)$, *where* $\mathcal{R}(B)$ *is the range space of* B.

We adopt the following notation in this chapter. Given a matrix $A \in \mathbb{R}^{n\times n}$, we denote the (i,j)-th entry of A by either $A_{i,j}$ or $A(i,j)$. Let α and β be given index sets, i.e., subsets of $\{1,2,\cdots,n\}$. We denote the cardinality of α by $|\alpha|$ and its complement by $\alpha^c := \{1,2,\cdots,n\}\backslash\alpha$. Let $A_{\alpha,\beta}$ denote the submatrix of A with rows indexed by α and columns indexed by β, i.e.,

$$A_{\alpha,\beta} := \begin{pmatrix} A_{\alpha_1,\beta_1} & \cdots & A_{\alpha_1,\beta_{|\beta|}} \\ \vdots & & \vdots \\ A_{\alpha_{|\alpha|},\beta_1} & \cdots & A_{\alpha_{|\alpha|},\beta_{|\beta|}} \end{pmatrix}.$$

Without introducing any confusion, we write i for the index set $\{i\}$ and denote the complement of $\{i\}$ by $i^c := \{1,2,\cdots,n\}\backslash\{i\}$. Hence, A_{i^c,i^c} is the submatrix of A that remains after removing its i-th row and column, and $A_{i^c,i}$ is the ith column of the matrix A without the element $A_{i,i}$.

The rest of this chapter is organized as follows. In section 2.2, we present a prototype of

the RBR method for solving a general SDP. In section 2.3, the RBR method is specialized for solving SDPs with only diagonal element constraints. We interpret this RBR method in terms of the logarithmic barrier function in section 2.3.1 and prove convergence to a global minimizer. To handle general linear constraints, we apply the RBR method in section 2.4 to a sequence of unconstrained problems using an augmented Lagrangian function approach. Specialized versions for the maxcut SDP relaxation and the minimum nuclear norm matrix completion problem are presented in sections 2.4.2 and 2.4.3, respectively. Finally, numerical results for the maxcut and matrix completion problem, are presented in section 2.5 to demonstrate the robustness and efficiency of our algorithms.

2.2 A row-by-row method prototype

Consider the semidefinite programming (SDP) problem

$$\begin{aligned} \min_{X \in S^n} \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \mathcal{A}(X) = b, \quad X \succeq 0. \end{aligned} \tag{2.2.1}$$

Given a strictly feasible solution $X^k \succ 0$, we can construct a second-order cone programming (SOCP) restriction for the SDP problem (2.2.1) as follows. Fix the $n(n-1)/2$ variables in the $(n-1) \times (n-1)$ submatrix $B := X^k_{1^c,1^c}$ of X^k and let ξ and y denote the remaining unknown variables $X_{1,1}$ and $X_{1^c,1}$ (i.e., row 1/column 1), respectively, that is $X := \begin{pmatrix} \xi & y^\top \\ y & B \end{pmatrix} := \begin{pmatrix} \xi & y^\top \\ y & X^k_{1^c,1^c} \end{pmatrix}$. It then follows from Theorem 2.1.1 that $X \succeq 0$ is equiv-

alent to $\xi - y^\top B^{-1} y \geq 0$, and hence, the SDP problem (2.2.1) becomes

$$\begin{aligned} \min_{[\xi;y]\in\mathbb{R}^n} \quad & \widetilde{c}^\top [\xi; y] \\ \text{s.t.} \quad & \widetilde{A}\,[\xi; y] = \widetilde{b}, \\ & \xi - y^\top B^{-1} y \geq 0, \end{aligned} \tag{2.2.2}$$

where $\nu > 0$, $\widetilde{c}$, $\widetilde{A}$ and $\widetilde{b}$ are defined as follows using the subscript $i = 1$:

$$\widetilde{c} := \begin{pmatrix} C_{i,i} \\ 2C_{i^c,i} \end{pmatrix}, \quad \widetilde{A} := \begin{pmatrix} A_{i,i}^{(1)} & 2A_{i,i^c}^{(1)} \\ \dots & \dots \\ A_{i,i}^{(m)} & 2A_{i,i^c}^{(m)} \end{pmatrix} \text{ and } \widetilde{b} := \begin{pmatrix} b_1 - \left\langle A_{i^c,i^c}^{(1)}, B \right\rangle \\ \dots \\ b_m - \left\langle A_{i^c,i^c}^{(m)}, B \right\rangle \end{pmatrix}. \tag{2.2.3}$$

If we let $LL^\top = B$ be the Cholesky factorization of B and introduce a new variable $z = L^{-1} y$, the Schur complement constraint $\xi - y^\top B^{-1} y \geq 0$ is equivalent to the linear constraint $Lz = y$ and the rotated second-order cone constraint $\|z\|_2^2 \leq \xi$. Furthermore, positive definiteness of the solution X can be maintained if we replace the 0 on the right hand side of the Schur complement constraint in subproblem (2.2.2) by $\nu > 0$, i.e., if we replace subproblem (2.2.2) by

$$\begin{aligned} \min_{[\xi;y]\in\mathbb{R}^n} \quad & \widetilde{c}^\top [\xi; y] \\ \text{s.t.} \quad & \widetilde{A}\,[\xi; y] = \widetilde{b}, \\ & \xi - y^\top B^{-1} y \geq \nu. \end{aligned} \tag{2.2.4}$$

Clearly, similar problems can be constructed if for any i, $i = 1, \cdots, n$, all elements of X^k other than those in the i-th row/column are fixed and only the elements in the i-th row/column are treated as unknowns.

Remark 2.2.1. For any $i \in \{1, \cdots, n\}$, the Schur complement of X_{i^c,i^c} in X is $(X/X_{i^c,i^c}) := X_{i,i} - (X_{i^c,i})^\top X^\dagger_{i^c,i^c} X_{i^c,i}$. There exists a permutation matrix P of the rows and columns of X that put X_{i^c,i^c} into the lower right corner of X, leaving the rows and columns of X_{i,i^c} and $X_{i^c,i}$ in the same increasing order in X, that is $P^\top X P = \begin{pmatrix} X_{i,i} & X^\top_{i^c,i} \\ X_{i^c,i} & X_{i^c,i^c} \end{pmatrix}$. Therefore, the Schur complement of X_{i^c,i^c} in X is $((P^\top X P)/X_{i^c,i^c})$.

We now present our row-by-row (**RBR**) method for solving (2.2.1). Starting from a positive definite feasible solution X^1, we update one row/column of the solution X at each of n inner steps by solving subproblems of the form (2.2.4). This procedure from the first row to the n-th row is called a *cycle*. At the first step of the k-th cycle, we fix $B := X^k_{1^c,1^c}$, and solve subproblem (2.2.4), whose solution is denoted by $[\xi; y]$. Then the first row/column of X^k is replaced by $X^k_{1,1} := \xi$ and $X^k_{1^c,1} := y$. Similarly, we set $B := X^k_{i^c,i^c}$ in the i-th inner iteration and assign the parameters $\widetilde{c}$, $\widetilde{A}$ and $\widetilde{b}$ according to (2.2.3). Then the solution $[\xi; y]$ of (2.2.4) is used to set $X^k_{i,i} := \xi$ and $X^k_{i^c,i} := y$. The k-th cycle is finished after the n-th row/column is updated. Then we set $X^{k+1} := X^k$ and repeat this procedure until the relative decrease in the objective function on a cycle becomes smaller than some tolerance ε. Our RBR method prototype is outlined in Algorithm 1.

We also denote by $X^{k,j}$ the solution X^k at the end of the j-th inner iteration in the k-th cycle and use the convention that $X^{k,0} := X^k$ and $X^{k+1} := X^{k,n}$. Our RBR method is

Algorithm 1: A row-by-row (RBR) method prototype

Set $X^1 \succ 0$, $\nu \geq 0$, $k := 1$ and $\varepsilon \geq 0$. Compute $F^0 := \langle C, X^1 \rangle$ and set $F^1 := +\infty$.
while $\frac{F^{k-1}-F^k}{\max\{|F^{k-1}|,1\}} \geq \varepsilon$ **do**
  for $i = 1, \cdots, n$ **do**
    Set $B := X^k_{i^c,i^c}$ and the parameters $\widetilde{c}$, $\widetilde{A}$ and $\widetilde{b}$ according to (2.2.3).
    Solve the SOCP subproblem (2.2.4) whose solution is denoted by ξ and y.
    Update $X^k_{i,i} := \xi$, $X^k_{i^c,i} := y$ and $X^k_{i,i^c} := y^\top$.
  Compute $F^k := \langle C, X^k \rangle$. Set $X^{k+1} := X^k$ and $k := k+1$.

similar to a block *Gauss-Seidel* method for solving a system of linear equations and a block coordinate descent method (sometimes referred to as a nonlinear Gauss-Seidel method) for nonlinear programming, except that because of the symmetry of X, the blocks in our method overlap. Specifically, exactly one of the variables in any two inner iterations of the RBR method overlap. For example, $X_{1,2}$ is a variable in the three dimensional problem in both the first and second inner iterations:

$$X^{k,1} := \begin{pmatrix} X^{k,1}_{1,1} & X^{k,1}_{1,2} & X^{k,1}_{1,3} \\ X^{k,1}_{1,2} & X^{k,0}_{2,2} & X^{k,0}_{2,3} \\ X^{k,1}_{1,3} & X^{k,0}_{2,3} & X^{k,0}_{3,3} \end{pmatrix}, \quad X^{k,2} := \begin{pmatrix} X^{k,1}_{1,1} & \boxed{X^{k,2}_{1,2}} & X^{k,1}_{1,3} \\ \boxed{X^{k,2}_{1,2}} & X^{k,2}_{2,2} & X^{k,2}_{2,3} \\ X^{k,1}_{1,3} & X^{k,2}_{2,3} & X^{k,0}_{3,3} \end{pmatrix}.$$

2.3 The RBR method for SDP with only diagonal element constraints

In this section, we consider an SDP whose constraints $\mathcal{A}(X) = b$ are of the form $X_{i,i} = b_i$, where $b_i > 0$, $i = 1, \cdots, n$. Without loss of generality, we assume that $b_i = 1$, for $i = 1, \cdots, n$,

and study the problem

$$\begin{aligned} \min_{X \in S^n} \quad & \langle C, X \rangle \\ \text{s.t.} \quad & X_{ii} = 1, \quad i = 1, \cdots, n, \\ & X \succeq 0. \end{aligned} \tag{2.3.1}$$

Note that (2.3.1) is the well known SDP relaxation [3,10,25,33,39] for the maxcut problem, which seeks to partition the vertices of a graph into two sets so that the sum of the weighted edges connecting vertices in one set with vertices in the other set is maximized.

We now present the RBR subproblem for solving (2.3.1). Throughout the algorithm the diagonal elements of X are kept fixed at 1. At the ith step of the k-th cycle, we fix $B = X^k_{i^c,i^c}$, where X^k is the iterate at the $(i-1)$-th step of the k-th cycle. Here, we do not assume that B is positive definite and use the generalized Schur complement to construct the second-order cone constraint. Hence, the subproblem (2.2.4) for generic SDP is reduced to

$$\begin{aligned} \min_{y \in \mathbb{R}^{n-1}} \quad & \widehat{c}^{\top} y \\ \text{s.t.} \quad & 1 - y^{\top} B^{\dagger} y \geq \nu, \quad y \in \mathcal{R}(B), \end{aligned} \tag{2.3.2}$$

where $\widehat{c} := 2C_{i^c,i}$ and $\nu < 1$. Fortuitously, the optimal solution of (2.3.2) is determined by a single matrix-vector product as we can see from the following lemma.

Lemma 2.3.1. *If* $\gamma := \widehat{c}^{\top} B \widehat{c} > 0$, *the solution of problem* (2.3.2) *is given by*

$$y = -\sqrt{\frac{1-\nu}{\gamma}} B\widehat{c}. \tag{2.3.3}$$

Otherwise, the solution is $y = 0$.

Proof. Since the matrix $B \in S_+^n$ is real symmetric and the rank of B is $r > 0$, B has the spectral decomposition

$$B = Q\Lambda Q^\top = \begin{pmatrix} Q_r & Q_l \end{pmatrix} \begin{pmatrix} \Lambda_r & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Q_r^\top \\ Q_l^\top \end{pmatrix} = Q_r \Lambda_r Q_r^\top, \tag{2.3.4}$$

where Q is an orthogonal matrix, $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_r, 0, \cdots 0)$, and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0$.

Hence, the Moore-Penrose pseudo-inverse of B is

$$B^\dagger = \begin{pmatrix} Q_r & Q_l \end{pmatrix} \begin{pmatrix} \Lambda_r^{-1} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Q_r^\top \\ Q_l^\top \end{pmatrix} = Q_r \Lambda_r^{-1} Q_r^\top.$$

Let $z = Q^\top y =: [z_r; z_l]$. Since $y \in \mathcal{R}(B)$ and $\mathcal{R}(B) = \mathcal{R}(Q_r)$, $z_l = 0$; hence, problem (2.3.2) is equivalent to

$$\begin{aligned} \min_{z_r \in \mathbb{R}^r} \quad & (Q_r^\top \widehat{c})^\top z_r \\ \text{s.t.} \quad & 1 - z_r^\top \Lambda_r^{-1} z_r \geq \nu, \end{aligned} \tag{2.3.5}$$

whose Lagrangian function is $\ell(z_r, \lambda) = (Q_r^\top \widehat{c})^\top z_r - \frac{\lambda}{2}(1 - \nu - z_r^\top \Lambda^{-1} z_r)$, where $\lambda \geq 0$. At an optimal solution z_r^* to (2.3.5),

$$\nabla_{z_r} \ell(z_r^*, \lambda^*) = Q_r^\top \widehat{c} + \lambda^* \Lambda_r^{-1} z_r^* = 0,$$

which implies $z_r^* = -\Lambda_r Q_r^\top \widehat{c}/\lambda^*$. Since z_r^* is on the boundary of the constraint, i.e., $1 -$

$(z_r^*)^\top \Lambda^{-1} z_r^* = \nu$, we obtain

$$1 - \frac{\widehat{c}^\top Q_r \Lambda_r \Lambda_r^{-1} \Lambda_r Q_r^\top \widehat{c}}{(\lambda^*)^2} = 1 - \frac{\gamma}{(\lambda^*)^2} = \nu.$$

Hence, if $\gamma > 0$, we obtain $\lambda^* = \sqrt{\gamma/(1-\nu)}$ and

$$y^* = Q_r z_r^* = -\sqrt{\frac{1-\nu}{\gamma}} Q_r \Lambda_r Q_r^\top \widehat{c} = -\sqrt{\frac{1-\nu}{\gamma}} B\widehat{c}.$$

Otherwise, $\lambda^* = 0$ and $y^* = 0$. □

We present the RBR method for (2.3.1) in Algorithm 2.

Algorithm 2: A RBR method (PURE-RBR-M) for problem (2.3.1)

Set $X^1 \succ 0$, $\nu > 0$, $k := 1$ and $\varepsilon \geq 0$. Compute $F^0 := \langle C, X^1 \rangle$ and set $F^1 := +\infty$.
while $\frac{F^{k-1} - F^k}{\max\{|F^{k-1}|, 1\}} \geq \varepsilon$ **do**
  for $i = 1, \cdots, n$ **do**
    Compute $x := X^k_{i^c, i^c} C_{i^c, i}$ and $\gamma := x^\top C_{i^c, i}$. Set $X^k_{i,i} := 1$.
    if $\gamma > 0$ **then** compute $X^k_{i^c, i} := -\sqrt{\frac{1-\nu}{\gamma}} x$, **else** set $X^k_{i^c, i} := \mathbf{0}$.
  Compute $F^k := \langle C, X^k \rangle$. Set $X^{k+1} := X^k$ and $k := k + 1$.

Algorithm 2 is extremely simple since only a single matrix-vector product is involved at each inner step. Numerical experiments show Algorithm 2 works fine if the initial solution X is taken as the identity matrix even if we take $\nu = 0$. However, the following is a 3×3 example which shows that if started at a rank one point that is not optimal, the RBR method using $\nu = 0$ either does not move away from the initial solution or it moves to a non-optimal

Table 2.1: Feasible rank-one and optimal solutions of (2.3.6)

solution	(x,y,z)	$\frac{1}{2}\langle C,X\rangle = \frac{3}{4}x-y-z$
$X^{(1)}$	(1,1,1)	-5/4
$X^{(2)}$	(-1,1,-1)	-3/4
$X^{(3)}$	(-1,-1,1)	-3/4
$X^{(4)}$	(1,-1,-1)	11/4
$X^{(5)}$	(-1/9,2/3,2/3)	-17/12

rank-one solution and stays there. Let

$$X = \begin{pmatrix} 1 & x & y \\ x & 1 & z \\ y & z & 1 \end{pmatrix} \text{ and } C = \begin{pmatrix} 0 & 3/4 & -1 \\ 3/4 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix}. \tag{2.3.6}$$

The rank-one feasible solutions $X^{(1)}$, $X^{(2)}$, $X^{(3)}$ and $X^{(4)}$ and the rank-two optimal solution $X^{(5)}$ for this example are given in Table 2.1. Starting at $X^{(1)}$, each row-by-row minimization step leaves the matrix unchanged. Starting at $X^{(2)}$, $X^{(3)}$ or $X^{(4)}$, the row-by-row method moves to the point $X^{(1)}$ and then remains there. Interestingly, Algorithm 2 is able to find the optimal solution if the off-diagonal elements x,y and z of the rank-one solutions in Table 2.1 are scaled by 0.999 (or even by a number much closer to 1).

2.3.1 Interpretation of Algorithm 2 in terms of the logarithmic barrier function

In this subsection, we interpret Algorithm 2 as a variant of the row-by-row method applied to a logarithmic barrier function approximation to (2.3.1).

Lemma 2.3.2 below relates the optimal solution (2.3.3) of the RBR subproblem (2.3.2) to the solution of the following logarithmic barrier function minimization

$$\min_{y\in\mathbb{R}^{n-1}} \quad \widehat{c}^\top y - \sigma\log(1-y^\top B^\dagger y), \quad \text{s.t. } y\in\mathcal{R}(B). \tag{2.3.7}$$

Lemma 2.3.2. *If* $\gamma := \widehat{c}^\top B\widehat{c} > 0$, *the solution of problem* (2.3.7) *is*

$$y = -\frac{\sqrt{\sigma^2+\gamma}-\sigma}{\gamma}B\widehat{c}. \tag{2.3.8}$$

Hence, the subproblem (2.3.2) *has the same solution as* (2.3.7) *if* $\nu = 2\sigma\frac{\sqrt{\sigma^2+\gamma}-\sigma}{\gamma}$.

Proof. Similarly to Lemma 2.3.1, we have the spectral decomposition (2.3.4) of B. Let $z = Q^\top y =: [z_r; z_l]$. Since $y\in\mathcal{R}(B)$ and $\mathcal{R}(B) = \mathcal{R}(Q_r)$, we obtain $z_l = 0$ and hence $y = Q_r z_r$. Therefore, problem (2.3.7) is equivalent to

$$\min_{z_r} \quad (Q_r^\top\widehat{c})^\top z_r - \sigma\log(1-z_r^\top\Lambda_r^{-1}z_r), \tag{2.3.9}$$

whose first-order optimality conditions are

$$Q_r^\top\widehat{c} + \frac{2\sigma\Lambda_r^{-1}z_r^*}{1-(z_r^*)^\top\Lambda_r^{-1}z_r^*} = 0, \text{ and } 1-(z_r^*)^\top\Lambda_r^{-1}z_r^* > 0. \tag{2.3.10}$$

Let $\theta = 1-(z_r^*)^\top\Lambda_r^{-1}z_r^*$. Then equation (2.3.10) implies that $z_r^* = -\frac{\theta\Lambda_r Q_r^\top\widehat{c}}{2\sigma}$. Substituting this expression for z_r^* into the definition of θ, we obtain $\theta^2\frac{\gamma}{4\sigma^2}+\theta-1=0$, which has a

positive root $\theta = \frac{2\sigma\sqrt{\sigma^2+\gamma}-2\sigma^2}{\gamma}$. Hence, $y^* = -\frac{\sqrt{\sigma^2+\gamma}-\sigma}{\gamma}B\widehat{c}$. Since

$$1-(z_r^*)^\top \Lambda_r^{-1} z_r^* = 1-(y^*)^\top B^\dagger y^* = 1-\frac{\left(\sqrt{\sigma^2+\gamma}-\sigma\right)^2}{\gamma} = \frac{2\sigma\sqrt{\sigma^2+\gamma}-2\sigma^2}{\gamma} > 0,$$

and $\nabla^2\phi_\sigma(y) \succeq 0$, y^* is an optimal solution of (2.3.7). Furthermore, problems (2.3.2) and (2.3.7) are equivalent if

$$\frac{\sqrt{\sigma^2+\gamma}-\sigma}{\gamma} = \sqrt{\frac{1-\nu}{\gamma}},$$

that is $\nu = 2\sigma\frac{\sqrt{\sigma^2+\gamma}-\sigma}{\gamma}$. □

Remark 2.3.1. Note from (2.3.8) that $\lim_{\sigma\to 0} y = -\frac{B\widehat{c}}{\sqrt{\gamma}}$.

We now consider the logarithmic barrier problem for (2.3.1), i.e.,

$$\begin{array}{ll} \min\limits_{X\in S^n} & \phi_\sigma(X) := \langle C, X\rangle - \sigma\log\det X \\ \text{s.t.} & X_{ii} = 1, \forall i = 1,\cdots,n, \quad X \succeq 0, \end{array} \tag{2.3.11}$$

where we define $\log\det(X)$ to be negative infinity for X not positive definite. Given a row i and fixing the block $B = X_{i^c,i^c}$, we have from (2.1.2) that

$$\det(X) = \det(B)(1 - X_{i^c,i}^\top B^{-1} X_{i^c,i}),$$

which implies that

$$\phi_\sigma(X) := \widehat{c}^\top X_{i^c,i} - \sigma\log(1 - X_{i^c,i}^\top B^{-1} X_{i^c,i}) + w(B),$$

where $\widehat{c} = 2C_{i^c,i}$ and $w(B)$ is a function of B (i.e., a constant). Hence, problem (2.3.11) becomes the unconstrained minimization problem

$$\min_{y\in\mathbb{R}^{n-1}} \quad \widehat{c}^{\top}y - \sigma\log(1-y^{\top}B^{-1}y), \tag{2.3.12}$$

which is a special version of problem (2.3.7). Therefore, it follows from Lemma 2.3.2 that Algorithm 2 is essentially a row-by-row method for solving problem (2.3.11), if in that algorithm ν is replaced by $2\sigma\frac{\sqrt{\sigma^2+\gamma}-\sigma}{\gamma}$.

Our RBR method can be extended to solve

$$\begin{array}{ll} \min\limits_{X\in S^n} & \psi_\sigma(X) := f(X) - \sigma\log\det X \\ \text{s.t.} & X\in\mathcal{X} := \{X\in S^n \mid L\le X\le U, X\succeq 0\}. \end{array} \tag{2.3.13}$$

where $f(X)$ is a convex function of X, the constant matrices $L,U\in S^n$ satisfy $L\le U$ and $L\le X$ means that $L_{i,j}\le X_{i,j}$ for all $i,j=1,\cdots,n$. Note that problem (2.3.13) includes problem (2.3.11) as a special case if $L_{i,i}=U_{i,i}=1$ for $i=1,\cdots,n$ and $L_{i,j}=-\infty$ and $U_{i,j}=\infty$, otherwise. Starting from the point $X^k\succ 0$ at the k-th cycle, we fix the $n(n-1)/2$ variables in the $(n-1)\times(n-1)$ submatrix $B := X^k_{i^c,i^c}$ of X^k and let ξ and y denote the remaining unknown variables $X_{i,i}$ and $X_{i^c,i}$ (i.e., row i/column i), respectively; i.e., $X^k :\approx \begin{pmatrix} \xi & y^{\top} \\ y & B \end{pmatrix}$.

Hence, the RBR subproblem of (2.3.13) becomes

$$\begin{aligned} \min_{X \in S^n} \quad & \widetilde{f}(\xi, y) - \sigma \log(\xi - y^\top B^{-1} y) \\ \text{s.t.} \quad & \begin{pmatrix} L_{i,i} \\ L_{i^c,i} \end{pmatrix} \leq \begin{pmatrix} \xi \\ y \end{pmatrix} \leq \begin{pmatrix} U_{i,i} \\ U_{i^c,i} \end{pmatrix}, \end{aligned} \tag{2.3.14}$$

where $\widetilde{f}(\xi, y) := f(X^k)$. Inspired by Proposition 2.7.1 in [6], we now prove a basic convergence result for the RBR method applied to problem (2.3.13).

Theorem 2.3.3. *Let $\{X^k\}$ be a sequence generated by the row-by-row method for solving* (2.3.14)*. Then every limit point of $\{X^k\}$ is a global minimizer of* (2.3.14)*.*

Proof. Clearly, our RBR method produces a sequence of nondecreasing objective function values

$$\psi_\sigma(X^k) \geq \psi_\sigma(X^{k,1}) \geq \psi_\sigma(X^{k,2}) \geq \cdots \geq \psi_\sigma(X^{k,n-1}) \geq \psi_\sigma(X^{k+1}). \tag{2.3.15}$$

Let $\widetilde{X}$ be a limit point of the sequence $\{X^k\}$. It follows from equation (2.3.15) that the sequences $\{\psi_\sigma(X^k)\}$, $\{\psi_\sigma(X^{k,1})\}$, $\cdots$, $\{\psi_\sigma(X^{k,n-1})\}$ all converge to $\psi_\sigma(\widetilde{X})$. Hence, $\widetilde{X}$ must be postitive definite because $\widetilde{X}$ is the limit point of a sequence of matrices that all lie in the compact level set $\{X \in \mathcal{X} | \psi_\sigma(X) \leq \psi_\sigma(X^1)\}$, all of whose members are positive definite. We now show that $\widetilde{X}$ minimizes $\psi_\sigma(X)$.

Let $\{X^{k_j}\}$ be a subsequence of $\{X^k\}$ that converges to $\widetilde{X}$. We first show that $\{X^{k_j,1} - X^{k_j}\}$ converges to zero as $j \to \infty$. Assume on the contrary, that $\{X^{k_j,1} - X^{k_j}\}$ does not

converges to zero. Then there exists a subsequence $\{\bar{k}_j\}$ of $\{k_j\}$ and some $\bar{\gamma} > 0$ such that $\gamma^{\bar{k}_j} := \|X^{\bar{k}_j,1} - X^{\bar{k}_j}\|_F \geq \bar{\gamma}$ for all j. Let $S^{\bar{k}_j,1} := (X^{\bar{k}_j,1} - X^{\bar{k}_j})/\gamma^{\bar{k}_j}$. Thus $X^{\bar{k}_j,1} = X^{\bar{k}_j} + \gamma^{\bar{k}_j} S^{\bar{k}_j,1}$, $\|S^{\bar{k}_j,1}\|_F = 1$ and $S^{\bar{k}_j,1}$ differs from zero only along the first row/column. Since $S^{\bar{k}_j,1}$ belongs to a compact set, it has a limit point $\bar{S}^1$. Hence, there exists a subsequence of $\{\hat{k}_j\}$ of $\{\bar{k}_j\}$ such that $S^{\hat{k}_j,1}$ converges to $\bar{S}^1$. Consider an arbitrary $t \in [0,1]$. Since $0 \leq t\bar{\gamma} \leq \gamma^{\hat{k}_j}$, $X^{\hat{k}_j} + tS^{\hat{k}_j,1}$ lies on the segment joining $X^{\hat{k}_j}$ and $X^{\hat{k}_j} + \gamma^{\hat{k}_j} S^{\hat{k}_j,1} = X^{\hat{k}_j,1}$, and belongs to $\mathcal{X}$ since $\mathcal{X}$ is a convex set. Moreover, since $X^{\hat{k}_j,1}$ uniquely minimizes $\psi_\sigma(X)$ over all X that differ from $X^{\hat{k}_j}$ along the first row/column, it follows from the convexity of $\psi_\sigma(X)$ that

$$\psi_\sigma(X^{\hat{k}_j,1}) = \psi_\sigma(X^{\hat{k}_j} + \gamma^{\hat{k}_j} S^{\hat{k}_j,1}) \leq \psi_\sigma(X^{\hat{k}_j} + t\gamma^{\hat{k}_j} S^{\hat{k}_j,1}) \leq \psi_\sigma(X^{\hat{k}_j}). \tag{2.3.16}$$

Since $\psi_\sigma(X^{\hat{k}_j,1})$ converges to $\psi_\sigma(\widetilde{X})$, it follows (2.3.16) that $\psi_\sigma(\widetilde{X}) \leq \psi_\sigma(\widetilde{X} + t\bar{\gamma}\bar{S}^1) \leq \psi_\sigma(\widetilde{X})$, which implies that $\psi_\sigma(\widetilde{X}) = \psi_\sigma(\widetilde{X} + t\bar{\gamma}\bar{S}^1)$ for all $t \in [0,1]$. Since $\bar{\gamma}\bar{S}^1 \neq 0$, this contradicts the fact that $\psi_\sigma(X)$ is strictly convex; hence $X^{k_j,1} - X^{k_j}$ converges to zero and $X^{k_j,1}$ converges to $\widetilde{X}$.

From the definition (2.3.13), we have $\psi_\sigma(X^{k_j,1}) \leq \psi_\sigma(X), \forall X \in V^{k_j,1}$, where

$$V^{k_j,1} := \left\{ \begin{pmatrix} \xi & y^\top \\ y & X^{k_j}_{1^c,1^c} \end{pmatrix} \middle| \begin{pmatrix} \xi \\ y \end{pmatrix} \in \mathbb{R}^n, \begin{pmatrix} L_{1,1} \\ L_{1^c,1} \end{pmatrix} \leq \begin{pmatrix} \xi \\ y \end{pmatrix} \leq \begin{pmatrix} U_{1,1} \\ U_{1^c,1} \end{pmatrix} \right\}.$$

Taking the limit as j tends to infinity, we obtain that $\psi_\sigma(\widetilde{X}) \leq \psi_\sigma(X), \forall X \in V^1$, where

$$V^1 := \left\{ \begin{pmatrix} \xi & y^\top \\ y & \widetilde{X}_{1^c,1^c} \end{pmatrix} \Big| \begin{pmatrix} \xi \\ y \end{pmatrix} \in \mathbb{R}^n, \begin{pmatrix} L_{1,1} \\ L_{1^c,1} \end{pmatrix} \leq \begin{pmatrix} \xi \\ y \end{pmatrix} \leq \begin{pmatrix} U_{1,1} \\ U_{1^c,1} \end{pmatrix} \right\},$$

which implies that, for any $p \in \{1, \cdots, n\}$,

$$\psi_\sigma(\widetilde{X}) \leq \psi_\sigma(X), \quad \forall X \in V^1 \text{ and } X_{p^c,1} = \widetilde{X}_{p^c,1},$$

i.e., all components of the first row and column $[\xi; y]$ other than the p-th are fixed. Since $\widetilde{X}$ lies in the open convex set S^n_{++}, we obtain from the optimality conditions that, for any $p \in \{1, \cdots, n\}$,

$$\left\langle \nabla\psi_\sigma(\widetilde{X}), X - \widetilde{X} \right\rangle \geq 0, \quad \forall X \in V^1 \text{ and } X_{p^c,1} = \widetilde{X}_{p^c,1},$$

which further gives that, for any $p \in \{1, \cdots, n\}$,

$$\left(\nabla\psi_\sigma(\widetilde{X})\right)_{p,1} \left(X_{p,1} - \widetilde{X}_{p,1}\right) \geq 0, \quad \forall X_{p,1} \text{ such that } L_{p,1} \leq X_{p,1} \leq U_{p,1}. \tag{2.3.17}$$

Repeating the above argument shows that for $i = 2, \cdots, n$, the points $X^{k_j,i}$ also converges to $\widetilde{X}$ and

$$\left(\nabla\psi_\sigma(\widetilde{X})\right)_{p,i} \left(X_{p,i} - \widetilde{X}_{p,i}\right) \geq 0, \quad \forall L_{p,i} \leq X_{p,i} \leq U_{p,i}, \tag{2.3.18}$$

for any $p \in \{1, \cdots, n\}$. Therefore, for any $X \in \mathcal{X}$, it follows from (2.3.17) and (2.3.18) that

$$\left\langle \nabla \psi_\sigma(\widetilde{X}), X - \widetilde{X} \right\rangle = \sum_{i,j=1,\cdots,n} \left(\nabla \psi_\sigma(\widetilde{X}) \right)_{i,j} \left(X_{i,j} - \widetilde{X}_{i,j} \right) \geq 0,$$

which implies that $\widetilde{X}$ is a global minimizer. □

2.4 An RBR method for SDP with general linear constraints

We now consider SDP with general linear constraints. Unfortunately, in this case, the RBR method may not converge to an optimal solution of problem (2.2.1). This is similar to the fact that (block) coordinate descent method may not converge to an optimal solution for a linearly constrained convex problem [28]. It has long been known that the coordinate descent method for general nonlinear programming may not converge [47]. Here is a 2-dimensional example that shows that for general linear constraints the RBR method may not converge to a global minimizer. Consider the SDP

$$\begin{array}{ll} \min & X_{11} + X_{22} - \log\det(X) \\ \text{s.t.} & X_{11} + X_{22} \geq 4, \quad X \succeq 0. \end{array} \tag{2.4.1}$$

Starting from a point X, where $X_{11} = 1$, $X_{12} = 0$ and $X_{22} = 3$, the RBR subproblems are

$$\min \quad X_{11} - \log(3X_{11} - X_{12}^2), \text{ s.t. } X_{11} \geq 1,$$

and

$$\min \quad X_{22} - \log(X_{22} - X_{12}^2), \text{ s.t. } X_{22} \geq 3,$$

since $\log\det(X) = \log(X_{11}X_{22} - X_{12}^2)$. It is readily verified that optimal solutions to these subproblems are, respectively, $X_{11} = 1$, $X_{12} = 0$ and $X_{12} = 0$, $X_{22} = 3$; hence, the row-by-row method remains at the initial point, while the true optimal solution is $X = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$.

To overcome this type of failure, the coordinate descent method is usually applied to a sequence of unconstrained problems obtained by penalizing the constraints in the objective function. We adopt a similar approach here by embedding the pure RBR method in an augmented Lagrangian function framework. We then introduce specialized versions of this algorithm for the SDP relaxation of the maxcut problem (2.3.1) and the minimum nuclear norm matrix completion problem.

2.4.1 An RBR augmented Lagrangian method

In this subsection, we first introduce an augmented Lagrangian method and then combine it with the row-by-row method for solving the standard form SDP (2.2.1).

As is well known (see chapter 12.2 in [21]), the augmented Lagrangian method (1.2.5) is equivalent to minimizing a quadratic penalty function:

$$X^k := \arg\min_X \ \mathcal{F}(X, b^k, \mu^k) := \langle C, X\rangle + \frac{1}{2\mu^k}\|\mathcal{A}(X) - b^k\|_2^2, \text{ s.t. } X \succeq 0, \tag{2.4.2}$$

where $b^k = b + \mu^k\pi^k$ and the difference between $\mathcal{L}(X, \pi^k, \mu^k)$ (defined in (1.2.4)) and $\mathcal{F}(X, b^k, \mu^k)$

is the constant $-\frac{\mu^k}{2}\|\pi^k\|_2^2$. Hence, we consider an alternative version of the augmented Lagrangian method which solves (2.4.2) and updates b^k by

$$b^{k+1} := b + \frac{\mu^{k+1}}{\mu^k}\left(b^k - \mathcal{A}(X^k)\right), \tag{2.4.3}$$

where $b^1 := b$. We now apply the RBR method to minimize (2.4.2). Starting from the point $X^k \succ 0$ at the k-th iteration, the RBR subproblem corresponding to the quadratic SDP (2.4.2) that is obtained by fixing all elements of X^k other than those in the i-th row and column results in a minimization problem with two conic constraints. Specifically, we fix the $n(n-1)/2$ variables in the $(n-1)\times(n-1)$ submatrix $B := X^k_{i^c,i^c}$ of X^k and let ξ and y denote the remaining unknown variables $X_{i,i}$ and $X_{i^c,i}$ (i.e., row i/column i), respectively. Hence, the quadratic SDP problem (2.4.2) becomes, after, replacing the zero on the right hand side of the Schur complement constraints by $\nu > 0$ to ensure positive definiteness of X^k,

$$\begin{aligned} \min_{(\xi;y)\in\mathbb{R}^n} \quad & \widetilde{c}^\top \begin{pmatrix}\xi\\ y\end{pmatrix} + \frac{1}{2\mu^k}\left\|\widetilde{A}\begin{pmatrix}\xi\\ y\end{pmatrix} - \widetilde{b}\right\|_2^2 \\ \text{s.t.} \quad & \xi - y^\top B^{-1} y \geq \nu, \end{aligned} \tag{2.4.4}$$

where $\widetilde{c}$, $\widetilde{A}$ and $\widetilde{b}$ are given by (2.2.3) with b_i for $i = 1,\cdots,m$ replaced by b_i^k. If we let $LL^\top = B$ be the Cholesky factorization of B and introduce a new variable $z = L^{-1}y$, problem

(2.4.4) can be written as:

$$\begin{aligned} \min_{(\xi;z;\tau)} \quad & \widetilde{c}^{\top}\begin{pmatrix}\xi\\ Lz\end{pmatrix}+\frac{1}{2\mu}\tau \\ \text{s.t.} \quad & \left\|\widetilde{A}\begin{pmatrix}\xi\\ Lz\end{pmatrix}-\widetilde{b}\right\|_2^2 \le \tau \\ & \|z\|_2^2 \le \xi-\nu. \end{aligned} \tag{2.4.5}$$

Therefore, each step of our RBR augmented Lagrangian method involves solving an SOCP with two rotated second-order cone constraints. If B is only positive semidefinite, we can derive a similar SOCP by using the spectral decomposition of B. For references on solving SOCPs, see [1] for example. Our combined RBR augmented Lagrangian method for minimizing (2.2.1) is presented in Algorithm 3.

Algorithm 3: Row-by-row augmented Lagrangian method

Set $X^1 \succ 0$, $b^1 = b$, $\eta \in (0,1)$, $\nu > 0$, $\mu^1 > 0$, $\varepsilon, \varepsilon_r, \varepsilon_f \ge 0$ and $k := 1$.
Compute $F^0 := \langle C, X^1\rangle$ and set $F^1 := +\infty$.
while $\frac{F^{k-1}-F^k}{\max\{|F^{k-1}|,1\}} \ge \varepsilon$ *or* $\|\mathcal{A}(X^k)-b\|_2 \ge \varepsilon_r$ **do**
  Compute $f^0 := \langle C, X^k\rangle + \frac{1}{2\mu^k}\|\mathcal{A}(X^k)-b^k\|_2^2$ and set $f^1 := +\infty$.
  while $\frac{f^{k-1}-f^k}{\max\{|f^{k-1}|,1\}} \ge \varepsilon_f$ **do**
    for $i = 1, \cdots, n$ **do**
      S1 Set $B := X^k_{i^c,i^c}$ and compute $\widetilde{c}$, $\widetilde{A}$ and $\widetilde{b}$ from (2.2.3) with b replaced by b^k.
      S2 Solve the SOCP (2.4.4) and denote its solution by ξ and y.
      S3 Set $X^k_{i,i} := \xi$, $X^k_{i^c,i} := y$ and $X^k_{i,i^c} := y^{\top}$.
    Compute $F^k := \langle C, X^k\rangle$ and $f^k := F^k + \frac{1}{2\mu^k}\|\mathcal{A}(X^k)-b^k\|_2^2$.
  S4 Update $b^{k+1} := b + \frac{\mu^{k+1}}{\mu^k}\left(b^k - \mathcal{A}(X^k)\right)$.
  Choose $\mu^{k+1} \in [\eta\mu^k, \mu^k]$ and set $X^{k+1} := X^k$ and $k := k+1$.

The RBR method applied to problem (2.4.2) converges by Theorem 2.3.3 since solv-

ing the RBR subproblem (2.4.4) essentially corresponds to minimizing the unconstrained function obtained by subtracting $\sigma \log(\xi - y^\top B^{-1} y)$ from the objective function in (2.4.4) using an argument analogous to the one made in section 2.3.1.

2.4.2 Application to SDPs with only diagonal element constraints

Since the constraints in problem (2.3.1) are $X_{i,i} = 1$ for $i = 1, \cdots, n$, the quadratic term in the objective function of the RBR subproblem simplifies to

$$\left\| \widetilde{A} \begin{pmatrix} \xi \\ y \end{pmatrix} - \widetilde{b} \right\|^2 = (\xi - b_i^k)^2,$$

and problem (2.4.4) reduces to

$$\begin{aligned} \min_{(\xi; y) \in \mathbb{R}^n} \quad & c\xi + \widehat{c}^\top y + \frac{1}{2\mu^k}(\xi - b_i^k)^2 \\ \text{s.t.} \quad & \xi - y^\top B^{-1} y \geq \nu, \end{aligned} \tag{2.4.6}$$

where $c := C_{i,i}$, $\widehat{c} := 2C_{i^c,i}$ and $b_i^1 = 1$. The first-order optimality conditions for (2.4.6) are

$$\begin{aligned} \xi &= b_i^k + \mu^k(\lambda - c) \\ y &= -\frac{1}{2\lambda} B\widehat{c} \\ \xi &\geq y^\top B^{-1} y + \nu, \quad \lambda \geq 0 \text{ and } (\xi - y^\top B^{-1} y - \nu)\lambda = 0. \end{aligned}$$

If $\widehat{c} = 0$, then $y = 0$ and $\xi = \max\{\nu, b_i^k - \mu^k c\}$. Otherwise, λ is the unique real root of the cubic equation:

$$\varphi(\lambda) := 4\mu^k\lambda^3 + 4(b_i^k - \mu^k c - \nu)\lambda^2 - \gamma = 0, \tag{2.4.7}$$

which is positive. This follows from the continuity of $\varphi(\lambda)$ and the facts that $\varphi(0) = -\widehat{c}^\top B\widehat{c} < 0$, $\lim_{\lambda\to+\infty}\varphi(\lambda) = +\infty$ and

$$\varphi'(\lambda) = 12\mu^k\lambda^2 + 8(b_i^k - \mu^k c - \nu)\lambda \geq 4\mu^k\lambda^2$$

since $\xi = b_i^k - \mu^k c + \mu^k\lambda \geq \nu$, which implies that $\varphi'(0) = 0$ and $\varphi'(\lambda) > 0$ for $\lambda \neq 0$. We now present a specialized version of the RBR augmented Lagrangian method for problem (2.3.1) as Algorithm 4.

Algorithm 4: Row-by-row augmented Lagrangian method (ALAG-RBR-M) for problem (2.3.1)

This is a specialized version of Algorithm 3. In Algorithm 3,

1. replace $\mathcal{A}(X^k)$ by $\mathrm{diag}(X^k)$;

2. replace steps **S1-S3** by the following steps:

Set $c := C_{i,i}$, $\widehat{c} := 2C_{i^c,i}$ and $B := X^k_{i^c,i^c}$.
If $\widehat{c} = 0$, set $X^k_{ii} = \max\{\nu, b_i^k - \mu^k c\}$, $X^k_{i^c,i} = (X^k_{i,i^c})^\top = 0$.
Otherwise, compute the positive solution λ of (2.4.7), and set
$X^k_{i,i} = b_i^k + \mu^k(\lambda - c)$, $X^k_{i^c,i} = -\frac{1}{2\lambda}B\widehat{c}$ and $X^k_{i,i^c} = (X^k_{i^c,i})^\top$.

2.4.3 Matrix Completion

Given a matrix $M \in \mathbb{R}^{p\times q}$ and an index set

$$\Omega \subseteq \{(i,j) \mid i \in \{1,\cdots,p\}, j \in \{1,\cdots,q\}\},$$

the nuclear norm matrix completion problem is

$$\begin{array}{ll} \min_{W\in\mathbb{R}^{p\times q}} & \|W\|_* \\ s.t. & W_{ij} = M_{ij},\ \forall\ (i,j) \in \Omega. \end{array} \tag{2.4.8}$$

An equivalent SDP formulation of (2.4.8) is

$$\begin{array}{ll} \min_{X\in S^n} & Tr(X) \\ s.t. & X := \begin{bmatrix} X^{(1)} & W \\ W^\top & X^{(2)} \end{bmatrix} \succeq 0 \\ & W_{ij} = M_{ij},\ \forall\ (i,j) \in \Omega, \end{array} \tag{2.4.9}$$

where $n = p+q$ and the number of linear constraints is $m = |\Omega|$. Let M_Ω be the vector whose elements are the components of $\{M_{i,j} \mid (i,j) \in \Omega\}$ obtained by stacking the columns of M from column 1 to column q and then keeping only those elements that are in Ω. Hence, M_Ω corresponds to the right hand side b of the constraints in the general SDP (2.2.1).

We now present the RBR subproblem (2.4.4) corresponding to problem (2.4.9). First, the vector y can be partitioned into two subvectors whose elements are, respectively, in and

not in the set Ω:

$$y :\approx \begin{pmatrix} \widehat{y} \\ \widetilde{y} \end{pmatrix}, \quad \widehat{y} := X_{\alpha,i}, \text{ and } \widetilde{y} := X_{\beta,i},$$

where, the index sets α and β are

$$\beta := i^c \backslash \alpha, \; \alpha := \begin{cases} \{j+p, \mid j \in \bar{\alpha}\}, \text{ where } \bar{\alpha} := \{j \mid (i,j) \in \Omega, j = 1, \cdots, q\}, \text{ if } i \leq p, \\ \{j \mid (j,i) \in \Omega, j = 1, \cdots, p\}, \text{ if } p < i \leq n. \end{cases} \tag{2.4.10}$$

Hence, it follows from the special constraints $W_{ij} = M_{ij}, (i,j) \in \Omega$ in (2.4.8) that the norm of the residual of each RBR subproblem (2.4.4) is

$$\left\| \widetilde{A} \begin{pmatrix} \xi \\ y \end{pmatrix} - \widetilde{b} \right\| = \left\| X_{\alpha,i} - \widetilde{b} \right\| =: \|\widehat{y} - \widetilde{b}\|,$$

where

$$\widetilde{b} := \begin{cases} (M^k_{i,\bar{\alpha}})^\top, & \text{if } i \leq p, \\ M^k_{\alpha,i-p}, & \text{if } p < i \leq n, \end{cases} \tag{2.4.11}$$

and $M^1 = M$. Therefore, the SOCP (2.4.4) becomes

$$\begin{aligned} \min_{(\xi;y) \in \mathbb{R}^n} \quad & \xi + \frac{1}{2\mu^k} \left\| \widehat{y} - \widetilde{b} \right\|_2^2 \\ \text{s.t.} \quad & \xi - y^\top B^{-1} y \geq \nu, \end{aligned} \tag{2.4.12}$$

where the matrix $B = \begin{pmatrix} X^k_{\alpha,\alpha} & X^k_{\alpha,\beta} \\ X^k_{\beta,\alpha} & X^k_{\beta,\beta} \end{pmatrix}$.

Lemma 2.4.1. *The optimal solution of the RBR subproblem*(2.4.12) *is given by*

$$\begin{cases} \xi = \dfrac{1}{2\mu^k}\widehat{y}^\top(\widetilde{b}-\widehat{y}) + \nu, \\ \widehat{y} = \left(2\mu^k I + X^k_{\alpha,\alpha}\right)^{-1} X^k_{\alpha,\alpha}\widetilde{b}, \quad \widetilde{y} = \dfrac{1}{2\mu^k} X^k_{\beta,\alpha}(\widetilde{b}-\widehat{y}). \end{cases} \tag{2.4.13}$$

Proof. Note that the optimal solution $[\xi; y] = [\xi; \widehat{y}; \widetilde{y}]$ of (2.4.12) must satisfy $\xi = y^\top B^{-1} y + \nu$. Hence, (2.4.12) is equivalent to an unconstrained quadratic minimization problem

$$\min_y y^\top B^{-1} y + \frac{1}{2\mu^k}\left\|\widehat{y}-\widetilde{b}\right\|_2^2, \tag{2.4.14}$$

whose optimality conditions are

$$\begin{pmatrix} X^k_{\alpha,\alpha} & X^k_{\alpha,\beta} \\ X^k_{\beta,\alpha} & X^k_{\beta,\beta} \end{pmatrix}^{-1} \begin{pmatrix} \widehat{y} \\ \widetilde{y} \end{pmatrix} + \frac{1}{2\mu^k}\begin{pmatrix} \widehat{y}-\widetilde{b} \\ \mathbf{0} \end{pmatrix} = 0, \tag{2.4.15}$$

which implies that

$$\begin{pmatrix} \widehat{y} \\ \widetilde{y} \end{pmatrix} + \frac{1}{2\mu^k}\begin{pmatrix} X^k_{\alpha,\alpha} \\ X^k_{\beta,\alpha} \end{pmatrix}\widehat{y} = \frac{1}{2\mu^k}\begin{pmatrix} X^k_{\alpha,\alpha} \\ X^k_{\beta,\alpha} \end{pmatrix}\widetilde{b}.$$

Therefore, $\widetilde{y} = \frac{1}{2\mu^k} X^k_{\beta,\alpha}(\widetilde{b}-\widehat{y})$, where $\widehat{y}$ can be computed from the system of linear equations

$$\left(2\mu^k I + X^k_{\alpha,\alpha}\right)\widehat{y} \;=\; X^k_{\alpha,\alpha}\widetilde{b}.$$

Then, it follows from $\xi = y^\top B^{-1} y + \nu$ and (2.4.15) that

$$\begin{aligned} \xi &= \begin{pmatrix} \widehat{y} \\ \widetilde{y} \end{pmatrix}^\top \begin{pmatrix} X^k_{\alpha,\alpha} & X^k_{\alpha,\beta} \\ X^k_{\beta,\alpha} & X^k_{\beta,\beta} \end{pmatrix}^{-1} \begin{pmatrix} \widehat{y} \\ \widetilde{y} \end{pmatrix} + \nu \\ &= \frac{1}{2\mu^k} \begin{pmatrix} \widehat{y} \\ \widetilde{y} \end{pmatrix}^\top \begin{pmatrix} \widetilde{b} - \widehat{y} \\ \mathbf{0} \end{pmatrix} + \nu = \frac{1}{2\mu^k} \widehat{y}^\top (\widetilde{b} - \widehat{y}) + \nu. \end{aligned} \tag{2.4.16}$$

□

Note from (2.4.13) that we only need to solve a single system of linear equations, whose size is the number of sample elements in the row and hence expected to be small, to obtain the minimizer of the RBR subproblem (2.4.12). The specialized RBR method for minimizing (2.4.9) is presented in Algorithm 5.

2.5 Numerical Results

Although the numerical results that we present in this section are limited to two special classes of SDP problems, they illustrate the effectiveness of our RBR algorithmic framework. Specifically, they show that large scale SDPs can be solved in a moderate amount of time using only moderate amount of memory. Moreover, our tests show that the number cycles taken by our algorithm grows very slowly with the size of the problem.

Algorithm 5: The RBR method (RBR-MC) for problem (2.4.9)

Set $X^1 \succ 0$, $b^1 = b$, $\eta \in (0,1)$, $\nu > 0$, $\mu^1 > 0$, $\varepsilon, \varepsilon_r, \varepsilon_f \geq 0$ and $k := 1$.
Compute $F^0 := Tr(X^1)$ and set $F^1 := +\infty$.
while $\frac{F^{k-1}-F^k}{\max\{|F^{k-1}|,1\}} \geq \varepsilon$ *or* $\|X^k_\Omega - M_\Omega\|_2 \geq \varepsilon_r$ **do**
  Compute $f^0 := Tr(X^k) + \frac{1}{2\mu^k}\|X^k_\Omega - M^k_\Omega\|_2^2$ and set $f^1 := +\infty$.
  while $\frac{f^{k-1}-f^k}{\max\{|f^{k-1}|,1\}} \geq \varepsilon_f$ **do**
    for $i = 1, \cdots, n$ **do**
      Set α and β by (2.4.10), and $\widetilde{b}$ by (2.4.11).
      if $|\alpha| = 0$ **then** Set $X^k_{i,i} = 0$, $X^k_{i^c,i} = 0$ and $X^k_{i,i^c} = 0$.
      else
        Compute $X^k_{\alpha,i} := \left(2\mu^k I + X^k_{\alpha,\alpha}\right)^{-1} X_{\alpha,\alpha}\widetilde{b}$; $X^k_{\beta,i} = \frac{1}{2\mu} X_{\beta,\alpha}(\widetilde{b} - X^k_{\alpha,i})$;
        $X^k_{i,i} = \frac{1}{2\mu}(X^k_{\alpha,i})^\top(\widetilde{b} - X^k_{\alpha,i}) + \nu$ and set $X^k_{i,i^c} = (X^k_{i^c,i})^\top$.
    Compute $F^k := Tr(X^k)$ and $f^k := F^k + \frac{1}{2\mu^k}\|X^k_\Omega - M^k_\Omega\|_2^2$.
  Update $M^{k+1}_\Omega := M_\Omega + \frac{\mu^{k+1}}{\mu^k}\left(M^k_\Omega - X^k_\Omega\right)$.
  Choose $\mu^{k+1} \in [\eta\mu^k, \mu^k]$ and set $X^{k+1} := X^k$ and $k := k+1$.

2.5.1 The maxcut SDP relaxation

In this subsection, we demonstrate the effectiveness of the RBR methods Algorithms 2 (PURE-RBR-M) and 4 (ALAG-RBR-M) on a set of maxcut SDP relaxation problems and compare them with the code DSDP (version 5.8) [3]. The DSDP code implements a dual interior point method that is designed to take advantage of the structure of such problems. The main parts of our code were written in C language MEX-files for MATLAB (Release 7.3.0), and all experiments were performed on a Dell Precision 670 workstation with an Intel Xeon 3.4GHZ CPU and 6GB of RAM.

The test problems are based on graphs generated by "rudy", a machine independent graph generator written by G.Rinaldi. These graphs range in size from $n = 1000$ to $n = 4000$ and the arguments of "rudy" are similar to those used in the G Set of graphs tested

in [3, 33]. Specifically, for size $n = 1000$, we produced five different graphs "R1000-1", $\cdots$, "R1000-5" with a density of 1% (4,995 edges) and with random edge weights from $\{-1, 1\}$ by using the command

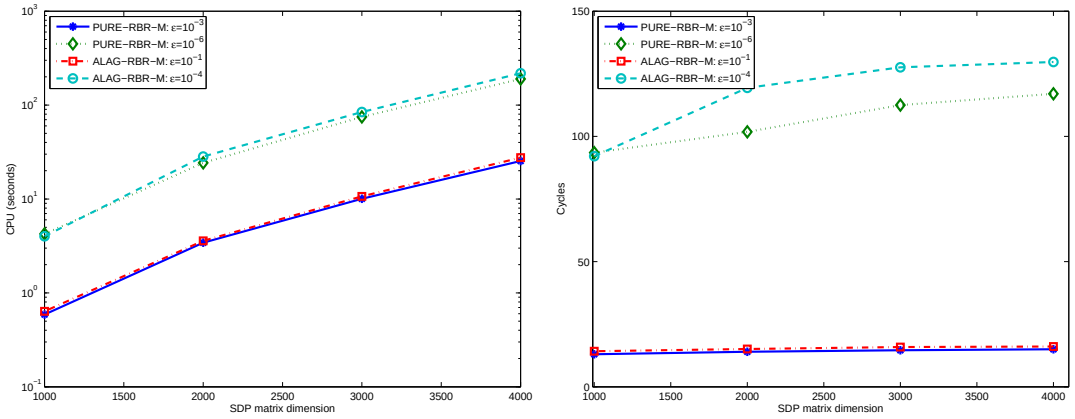
```
rudy -rnd_graph n 1 seed -random 0 1 seed  -times 2 -plus -1,
```

where seed $= 10n + i$, $i = 1, \cdots, 5$, respectively. The graphs from "R2000-1" to "R4000-5" were generated in a similar fashion. We also tested "almost" planar graphs having as their edge set the union of the edges of two (almost maximal) planar graphs with random edge weights from $\{-1, 1\}$; that is, the graphs "P1000-1" to "P4000-5" were generated by the command

```
rudy -planar n 99 seed1 -planar n  99 seed2  + -random 0 1 seed3 -times 2 -plus -1,
```

where seed1 $= 10n + i + 4$, seed2 $= 10n + i + 5$ and seed3 $= 10n + i$, for $n = 1000, \cdots, 4000$ and $i = 1, \cdots, 5$. In all cases the cost matrix C was the Laplace matrix of the graph divided by 4, i.e., $C = -\frac{1}{4}(\mathrm{diag}(Ae) - A)$, where A was the weighted adjacency matrix of the graph.

The parameters of DSDP were set to their default values. The parameter ν in the RBR methods was set to 10^{-6}. We ran PURE-RBR-M with two different tolerances, i.e., ε was set to 10^{-3} (moderately accurate) and 10^{-6} (highly accurate), respectively. Similarly, we ran ALAG-RBR-M with two different tolerance settings, that is, $\varepsilon, \varepsilon_r, \varepsilon_f$ were all set to 10^{-1} and 10^{-4}, respectively. For practical considerations, we terminated minimizing each augmented Lagrangian function if the number of cycles was greater than 5. The initial penalty parameter μ^1 in ALAG-RBR-M was set to 5 and was updated by $\mu^{k+1} = \max(0.5\mu^k, 10^{-1})$.



(a) CPU Time (b) Cycles

Figure 2.1: Relationship between the computational cost and SDP matrix dimension for the maxcut SDP relaxation

A summary of the computational results is presented in Table 2.2. In the table, "obj" denotes the objective function of the dual problem computed by DSDP, "rel-obj" denotes the relative error between the objective function value computed by the RBR methods and "obj", "CPU" denotes CPU time measured in seconds, and "cycle" denotes the total number RBR cycles. From the table, we can see that both RBR methods are able to solve the maxcut SDP relaxation very efficiently. The number of cycles required was almost the same for all of the problems, no matter what their size was.

To illustrate the relationship between the computational cost of the RBR methods and the dimension of the SDP matrices, we plot the average of the CPU time versus the dimension in Figure 2.1 (a) and the average of the number of cycles versus the dimension in Figure 2.1 (b). Somewhat surprisingly, our augmented Lagrangian RBR algorithm solved the maxcut SDP problems as efficiently as our pure RBR algorithm for a given relative error.

Table 2.2: Computational results for the maxcut SDP relaxation.

	DSDP		PURE-RBR-M						ALAG-RBR-M					
			$\epsilon = 10^{-3}$			$\epsilon = 10^{-6}$			$\epsilon = \epsilon_r = \epsilon_f = 10^{-1}$			$\epsilon = \epsilon_r = \epsilon_f = 10^{-4}$		
Name	obj	CPU	rel-obj	CPU	cycle	rel-obj	CPU	cycle	rel-obj	CPU	cycle	rel-obj	CPU	cycle
random graphs														
R1000-1	-1.4e+3	52.6	4.9e-3	0.6	13	3.0e-5	3.9	90	5.4e-3	0.6	13	3.2e-5	3.7	87
R1000-2	-1.4e+3	57.0	5.0e-3	0.6	13	3.6e-5	4.1	96	4.9e-3	0.6	14	4.2e-5	3.7	88
R1000-3	-1.5e+3	50.8	5.0e-3	0.6	13	3.8e-5	4.3	99	4.9e-3	0.6	14	4.0e-5	4.2	99
R1000-4	-1.4e+3	51.3	5.0e-3	0.6	13	3.2e-5	4.0	94	4.8e-3	0.6	14	3.3e-5	3.9	92
R1000-5	-1.5e+3	50.1	4.6e-3	0.6	13	3.5e-5	3.7	87	4.1e-3	0.6	14	3.6e-5	3.4	81
R2000-1	-4.1e+3	607.6	5.0e-3	3.9	14	3.7e-5	26.5	97	5.9e-3	3.8	14	1.9e-5	33.5	121
R2000-2	-4.1e+3	602.3	5.2e-3	3.9	14	3.6e-5	27.5	101	5.5e-3	4.2	15	1.9e-5	35.4	127
R2000-3	-4.2e+3	680.5	5.1e-3	4.3	14	3.4e-5	26.4	97	5.3e-3	4.2	15	1.6e-5	33.7	123
R2000-4	-4.2e+3	646.7	5.2e-3	3.9	14	3.2e-5	26.1	96	5.2e-3	4.2	15	1.4e-5	32.3	118
R2000-5	-4.1e+3	661.5	4.9e-3	3.9	14	3.9e-5	26.1	96	5.9e-3	3.8	14	2.0e-5	34.5	126
R3000-1	-7.7e+3	2576	5.0e-3	12.8	15	4.1e-5	90.0	103	5.1e-3	13.9	16	2.1e-5	110.8	127
R3000-2	-7.7e+3	2606	5.2e-3	13.2	15	3.7e-5	89.4	105	5.2e-3	14.1	16	2.1e-5	111.2	128
R3000-3	-7.9e+3	2530	5.0e-3	13.0	15	4.0e-5	91.4	107	5.1e-3	14.0	16	2.5e-5	109.8	127
R3000-4	-7.9e+3	2518	5.1e-3	13.6	15	4.0e-5	98.0	107	5.2e-3	13.9	16	2.3e-5	110.3	128
R3000-5	-7.7e+3	2514	5.3e-3	12.9	15	3.7e-5	91.8	107	5.4e-3	14.0	16	1.9e-5	109.8	128
R4000-1	-1.2e+4	6274	5.9e-3	36.5	15	4.0e-5	261.1	108	6.2e-3	39.0	16	2.4e-5	316.5	130
R4000-2	-1.2e+4	6310	5.7e-3	36.3	15	3.9e-5	265.7	108	6.0e-3	39.0	16	2.1e-5	313.5	130
R4000-3	-1.2e+4	6529	5.8e-3	36.0	15	4.1e-5	264.1	110	5.9e-3	39.5	16	2.5e-5	313.5	130
R4000-4	-1.2e+4	7018	5.8e-3	36.6	15	3.7e-5	261.3	108	6.1e-3	39.3	16	2.1e-5	315.3	130
R4000-5	-1.2e+4	5994	5.6e-3	36.7	15	3.8e-5	270.1	112	5.1e-3	41.6	17	2.5e-5	309.8	129
random planar graphs														
P1000-1	-1.4e+3	45.1	5.0e-3	0.6	13	4.0e-5	4.9	102	4.1e-3	0.7	15	3.8e-5	4.3	96
P1000-2	-1.4e+3	45.5	4.4e-3	0.6	13	2.9e-5	4.2	89	3.3e-3	0.6	14	2.3e-5	3.9	87
P1000-3	-1.5e+3	42.6	4.6e-3	0.6	13	3.5e-5	4.2	88	3.0e-3	0.7	15	2.7e-5	4.2	93
P1000-4	-1.4e+3	44.1	4.7e-3	0.6	13	3.8e-5	4.7	97	3.4e-3	0.7	14	3.8e-5	4.3	96
P1000-5	-1.4e+3	44.6	4.4e-3	0.6	13	3.2e-5	4.7	93	2.8e-3	0.7	15	2.4e-5	4.6	102
P2000-1	-2.9e+3	386.1	5.5e-3	3.0	14	3.7e-5	21.6	102	5.4e-3	3.0	15	2.5e-5	22.5	114
P2000-2	-2.8e+3	362.8	5.8e-3	2.9	14	3.9e-5	22.1	109	5.8e-3	3.2	16	2.9e-5	23.4	119
P2000-3	-2.9e+3	359.0	5.4e-3	2.9	14	3.7e-5	22.2	105	4.9e-3	3.2	16	2.5e-5	23.0	117
P2000-4	-2.9e+3	348.2	5.5e-3	2.9	14	4.0e-5	22.8	111	4.9e-3	3.0	15	2.9e-5	23.7	121
P2000-5	-2.9e+3	377.9	5.6e-3	2.9	14	3.9e-5	21.3	104	4.7e-3	3.2	16	3.2e-5	21.2	108
P3000-1	-4.3e+3	1400	6.0e-3	7.3	15	4.0e-5	56.3	117	6.2e-3	7.0	15	3.0e-5	58.3	127
P3000-2	-4.3e+3	1394	6.5e-3	7.0	14	4.7e-5	57.2	119	5.1e-3	7.5	16	3.3e-5	59.1	129
P3000-3	-4.4e+3	1351	6.3e-3	6.8	14	4.3e-5	57.0	118	5.2e-3	7.4	16	3.5e-5	58.1	128
P3000-4	-4.3e+3	1647	6.7e-3	7.0	14	4.8e-5	60.6	125	6.2e-3	7.4	16	4.0e-5	58.6	129
P3000-5	-4.3e+3	1757	6.7e-3	6.9	14	4.4e-5	57.0	117	5.7e-3	7.5	16	3.3e-5	57.2	125
P4000-1	-5.7e+3	3688	6.5e-3	14.3	15	4.3e-5	114.2	124	6.0e-3	15.5	16	3.0e-5	123.6	130
P4000-2	-5.9e+3	3253	6.5e-3	14.4	15	4.9e-5	116.7	126	6.2e-3	15.9	16	4.1e-5	125.2	130
P4000-3	-5.8e+3	3790	6.3e-3	14.2	15	4.8e-5	115.1	126	5.6e-3	15.3	16	3.8e-5	120.1	128
P4000-4	-5.8e+3	3474	6.8e-3	14.3	15	4.6e-5	118.8	128	6.5e-3	15.5	16	4.1e-5	123.8	131
P4000-5	-5.9e+3	3389	6.1e-3	14.3	15	4.4e-5	111.9	120	5.5e-3	15.4	16	3.6e-5	121.1	129

2.5.2 Matrix Completion

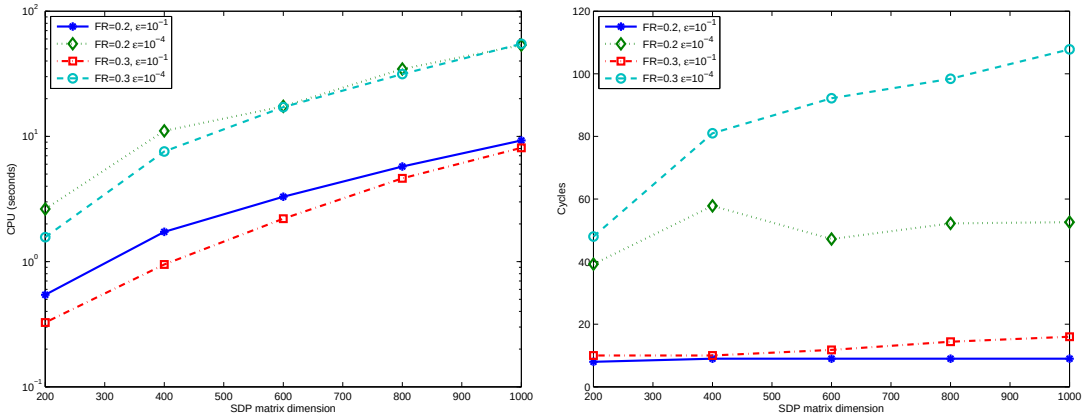
In this subsection, we evaluate the RBR method Algorithm 5 (RBR-MC) on the matrix completion problem (2.4.9). Note that the pure RBR method can be directly applied to this problem. However, preliminary numerical testing showed that this approach is much slower (i.e., converges much more slowly) than using RBR-MC. Random matrices $M \in \mathbb{R}^{p\times q}$ with rank r were created by the following procedure [41]: we first generated random matrices $M_L \in \mathbb{R}^{p\times r}$ and $M_R \in \mathbb{R}^{q\times r}$ with i.i.d. Gaussian entries and then set $M = M_L M_R^\top$; then we sampled a subset Ω of m entries uniformly at random. The ratio $m/(pq)$ between the number of measurements and the number of entries in the matrix is denoted by "SR" (sampling ratio). The ratio $r(p+q-r)/m$ between the dimension of a rank r matrix to the number of samples is denoted by "FR". In our tests, the rank r and the sampling entry m were taken consistently so that according to the theory in [15] the matrix M is the optimal solution of problem (2.4.9). Specifically, FR was set to 0.2 and 0.3 and r was set to 10. We tested five square matrices M with dimensions $p = q \in \{100, 200, \cdots, 500\}$ and set the number m to $r(p+q-r)/\mathrm{FR}$. All parameters p, q, r, m and the random seeds "seed" used by the random number generators "rand" and "randn" in MATLAB are reported in Table 2.3.

All parameters of RBR-MC were set to the same values as those used in ALAG-RBR-M. A summary of the computational results is presented in Table 2.3. In the table, "rel-X" denotes the relative error between the true and the recovered matrices, which is defined as rel-X $:= \frac{\|X-M\|_F}{\|M\|_F}$, and "rel-obj" denotes the relative error between the objective function

Table 2.3: Computational results for the matrix completion problem

	FR=0.2								FR=0.3							
	$\varepsilon = 10^{-1}$				$\varepsilon = 10^{-4}$				$\varepsilon = 10^{-1}$				$\varepsilon = 10^{-4}$			
seed	rel-X	rel-obj	CPU	cycle	rel-X	rel-obj	CPU	cycle	rel-X	rel-obj	CPU	cycle	rel-X	rel-obj	CPU	cycle
	p=q=100; r=10; m=9500; SR=0.95								p=q=100; r=10; m=6333; SR=0.63							
68521	1.7e-6	7.7e-3	0.5	8	3.3e-7	2.5e-4	2.8	42	5.1e-5	1.0e-3	0.3	10	4.3e-7	5.0e-5	1.7	51
56479	8.4e-7	3.7e-3	0.5	8	3.0e-7	2.9e-4	2.4	35	5.4e-5	4.9e-4	0.3	10	3.9e-7	5.0e-5	1.4	44
115727	1.3e-6	4.0e-3	0.5	8	3.4e-7	2.4e-4	2.4	37	3.8e-5	7.4e-4	0.3	10	4.5e-7	4.7e-5	1.4	43
27817	1.2e-6	4.6e-3	0.5	8	3.2e-7	2.4e-4	2.8	42	5.3e-5	1.0e-3	0.3	10	4.1e-7	5.4e-5	1.7	53
9601	1.1e-6	4.1e-3	0.6	8	3.1e-7	2.4e-4	2.7	40	3.1e-5	6.2e-4	0.3	10	4.1e-7	5.2e-5	1.6	49
	p=q=200; r=10; m=19500; SR=0.49								p=q=200; r=10; m=13000; SR=0.33							
68521	7.5e-5	1.1e-4	1.7	9	2.5e-7	6.4e-5	9.6	50	1.0e-3	4.9e-4	1.0	10	3.6e-7	1.9e-5	6.8	73
56479	6.0e-5	1.1e-4	1.8	9	2.3e-7	6.8e-5	8.1	42	1.5e-3	7.7e-4	0.9	10	3.3e-7	2.3e-5	8.1	87
115727	6.8e-5	2.1e-4	1.7	9	2.4e-7	6.8e-5	11.6	59	2.4e-3	6.0e-4	1.0	10	3.4e-7	2.0e-5	7.5	80
27817	5.3e-5	6.5e-4	1.7	9	2.3e-7	8.5e-5	14.0	74	2.5e-4	5.4e-4	0.9	10	3.2e-7	2.3e-5	7.8	84
9601	6.3e-5	3.3e-4	1.7	9	2.3e-7	8.0e-5	12.1	64	6.6e-4	5.0e-4	1.0	10	3.3e-7	2.1e-5	7.6	81
	p=q=300; r=10; m=29500; SR=0.33								p=q=300; r=10; m=19666; SR=0.22							
68521	1.0e-4	2.1e-4	3.3	9	2.0e-7	4.0e-5	21.7	59	1.0e-3	4.9e-4	2.0	11	3.0e-7	1.4e-5	17.7	96
56479	1.0e-4	2.8e-4	3.3	9	2.0e-7	3.8e-5	20.8	56	3.3e-4	4.3e-4	2.3	12	3.1e-7	1.3e-5	17.2	93
115727	9.4e-5	1.4e-4	3.3	9	2.0e-7	4.2e-5	24.7	67	1.2e-2	7.5e-4	2.4	13	3.2e-7	1.3e-5	15.5	83
27817	9.7e-5	7.1e-4	3.3	9	2.0e-7	3.7e-5	10.4	28	3.8e-3	5.0e-4	2.1	11	2.9e-7	1.3e-5	18.0	96
9601	1.0e-4	2.3e-3	3.3	9	1.9e-7	3.5e-5	9.5	26	1.8e-3	4.7e-4	2.2	12	2.9e-7	1.3e-5	17.2	93
	p=q=400; r=10; m=39500; SR=0.25								p=q=400; r=10; m=26333; SR=0.16							
68521	1.0e-4	1.2e-3	5.6	9	1.8e-7	2.6e-5	28.3	43	9.8e-3	6.2e-4	4.8	15	5.4e-6	9.2e-6	29.3	92
56479	9.9e-5	2.1e-4	5.8	9	1.8e-7	3.2e-5	48.1	71	3.0e-3	5.5e-4	4.5	14	2.7e-7	1.0e-5	31.8	101
115727	9.9e-5	1.0e-3	5.6	9	1.8e-7	2.7e-5	31.6	50	2.0e-3	5.0e-4	4.5	14	2.7e-7	1.0e-5	32.3	101
27817	2.2e-4	4.1e-4	5.8	9	1.8e-7	3.0e-5	37.9	57	8.3e-3	6.5e-4	4.5	14	2.8e-7	9.7e-6	32.6	100
9601	1.0e-4	1.3e-3	5.8	9	1.8e-7	2.5e-5	26.7	40	8.0e-3	5.5e-4	4.9	15	2.8e-7	9.6e-6	31.3	98
	p=q=500; r=10; m=49500; SR=0.20								p=q=500; r=10; m=33000; SR=0.13							
68521	2.4e-4	8.6e-4	9.1	9	1.7e-7	2.1e-5	56.0	56	5.3e-3	6.1e-4	8.2	16	2.6e-7	7.6e-6	54.4	107
56479	1.1e-4	8.5e-4	8.9	9	1.6e-7	2.3e-5	53.7	53	5.4e-3	5.9e-4	8.0	16	2.6e-7	7.2e-6	54.4	109
115727	1.1e-4	9.4e-4	9.4	9	1.6e-7	2.3e-5	49.8	48	8.2e-3	5.8e-4	8.1	16	2.6e-7	7.5e-6	54.4	108
27817	3.3e-4	1.5e-3	9.1	9	1.6e-7	2.1e-5	53.6	53	9.2e-3	6.7e-4	8.2	16	2.6e-7	7.6e-6	53.6	104
9601	1.1e-4	1.5e-3	9.9	9	1.6e-7	2.4e-5	54.9	53	2.1e-3	4.2e-4	8.0	16	2.5e-7	7.6e-6	57.3	111

value obtained and $\|M\|_*$, which is defined as rel-obj := $\frac{|\frac{1}{2}Tr(X)-\|M\|_*|}{\|M\|_*}$. DSDP is not included in this comparison because it takes too long to solve all problems. To illustrate the relationship between the computational cost of the RBR methods and the dimension of the matrices, we plot the average of the CPU time versus the dimension of the SDP matrix (i.e., $p+q$) in Figure 2.2 (a) and the average of the number of cycles versus this dimension in Figure 2.2 (b).



(a) CPU Time (b) Cycles

Figure 2.2: Relationship between the computational cost and SDP matrix dimension for nuclear norm matrix completion problems

2.6 Conclusion

In this chapter, we presented a new first-order algorithmic framework, the row-by-row (RBR) method, for solving semidefinite programming problems based on replacing the positive semidefinite constraint $X \succeq 0$ by requiring nonnegativity of the Schur complement of any $(n-1)$-dimensional principal submatrix of X, which is temporally fixed. By doing this, the positive semidefinite constraint is reduced to a simple second-order cone constraint. The pure RBR method is extremely effective in solving an SDP whose only constraints are that the diagonal elements of X are constant, since only a single matrix-vector product is involved at each inner step. To handle more general linear constraints in an SDP, we apply the pure RBR method to a sequence of unconstrained problems by using an augmented Lagrangian approach. Our method is especially suitable for solving the maxcut SDP relaxation and nuclear norm matrix completion problems since closed-form solutions for the RBR subproblems are available.

Chapter 3

Alternating Direction Augmented Lagrangian Methods for SDP

3.1 Introduction

In this chapter we present an alternating direction method based on an augmented Lagrangian framework for solving SDP problems. Alternating direction methods have been extensively studied to minimize the augmented Lagrangian function for optimization problems arising from partial differential equations (PDEs) [22, 24]. In these methods, the variables are partitioned into several blocks according to their roles, and then the augmented Lagrangian function is minimized with respect to each block by fixing all other blocks at each inner iteration. This simple idea has been applied to many other problem classes, such as, variational inequality problems [31, 32], linear programming [19], nonlinear convex optimization [7, 16, 36, 38, 54], maximal monotone operators [20] and nonsmooth ℓ_1

minimization arising from compressive sensing [57,60,63]. In [61], an alternating direction method for SDP is presented by reformulating the complementary condition as a projection equation.

Our algorithm applies the alternating direction method within a dual augmented Lagrangian framework. When our method is applied to an SDP in standard form, at each iteration it first minimizes the dual augmented Lagrangian function with respect to the Lagrange multipliers corresponding to the linear constraints, and then with respect to the dual slack variables while keeping the other variables fixed, after which it updates the primal variables. This algorithm is very closely related to the regularization method in [42]. While the theoretical algorithm in [42] updates the primal variable X only after a certain condition is satisfied, the actual algorithm implemented in [42] (i.e., Algorithm 5.1 in [42]) updates the primal variable X immediately after all other blocks are updated at each iteration, which is exactly our alternating direction method. The algorithms in [42] cannot be applied directly to SDPs with inequality constraints, and in particular, with positivity constraints, i.e., every component of X is nonnegative. In order to preserve the structure of these inequality constraints, such as sparsity and orthogonality, we do not transform them to equality constraints but add some extra steps to our alternating direction method to minimize the dual augmented Lagrangian function with respect to the Lagrange multipliers corresponding to the inequality constraints. This gives us a multiple-splitting alternating direction method. Numerical experiments on, for example, frequency assignment problems show that the performance of our method is significantly better than those in [42, 64].

Our contributions are as follows. Although the techniques for analyzing the alternating

direction methods for variational inequalities in [31] and a class of nonlinear optimization problems in [7] can be applied to analyze our algorithm for SDPs in standard form, we present a different and simple convergence proof by formulating our algorithm as a fixed point method. We note that the convergence properties of the actual implementation of the regularization method in [42] has not been studied. Moreover, we present a multiple-splitting alternating direction method to solve SDPs with inequality constraints directly without transforming them into SDPs in standard form by introducing a lot of auxiliary variables and constraints.

The rest of this chapter is organized as follows. We present an alternating direction augmented Lagrangian method for SDP in standard form in subsection 3.2.1 and analyze its convergence in subsection 3.2.2, and then extend this method to an expanded problem with inequality and positivity constraints in subsection 3.2.3. We discuss practical issues related to the eigenvalue decomposition performed at each iteration, strategies for adjusting the penalty parameter, the use of a step size for updating the primal variable X, termination rules and how to detect stagnation to enhance the performance of our methods in section 3.3. Finally, numerical results for frequency assignment, maximum stable set and binary integer quadratic programming problems are presented in section 3.4 to demonstrate the robustness and efficiency of our algorithm.

3.2 Alternating direction augmented Lagrangian methods

3.2.1 A two-splitting scheme for standard form SDPs

Consider the standard form SDP

$$\begin{array}{ll} \min\limits_{X\in S^n} & \langle C,X\rangle \\ \text{s.t.} & \mathcal{A}(X)=b, \quad X\succeq 0. \end{array} \tag{3.2.1}$$

We make the following assumption throughout our presentation.

Assumption 3.2.1. *The matrix A has full row rank and the Slater condition holds for* (3.2.1)*; that is, there exists a matrix* $\bar{X}\succ 0$ *satisfying* $\mathcal{A}(\bar{X})=b$.

The dual problem of (3.2.1) is

$$\begin{array}{ll} \min\limits_{y\in\mathbb{R}^m, S\in S^n} & -b^\top y \\ \text{s.t.} & \mathcal{A}^*(y)+S=C, \quad S\succeq 0. \end{array} \tag{3.2.2}$$

The augmented Lagrangian function for the dual SDP (3.2.2) corresponding to the linear constraints is defined as:

$$\mathcal{L}_\mu(X,y,S) \;:=\; -b^\top y+\langle X,\mathcal{A}^*(y)+S-C\rangle+\frac{1}{2\mu}\|\mathcal{A}^*(y)+S-C\|_F^2,$$

where $X\in S^n$ and $\mu>0$. Starting from $X^0=\mathbf{0}$, the augmented Lagrangian method solves

on the k-th iteration

$$\min_{y\in\mathbb{R}^m, S\in\mathcal{S}^n} \mathcal{L}_\mu(X^k, y, S), \quad \text{s.t.} \quad S \succeq 0, \tag{3.2.3}$$

for y^{k+1} and S^{k+1}, and then updates the primal variable X^{k+1} by

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu}. \tag{3.2.4}$$

Since solving problem (3.2.3) exactly is very expensive, we consider instead an alternating direction method. Starting from X^0 and S^0 at the first iteration, we update on the kth iteration the variables y, S and X by first minimizing $\mathcal{L}_\mu(X, y, S)$ with respect to y to obtain y^{k+1} with $X := X^k$ and $S := S^k$ fixed; then minimizing $\mathcal{L}_\mu(X, y, S)$ with respect to S subject to $S \succeq 0$ to obtain S^{k+1} with $X := X^k$ and $y := y^{k+1}$ fixed; and finally, updating X^k by (3.2.4); that is, we compute

$$y^{k+1} := \arg\min_{y\in\mathbb{R}^m} \mathcal{L}_\mu(X^k, y, S^k), \tag{3.2.5a}$$

$$S^{k+1} := \arg\min_{S\in\mathcal{S}^n} \mathcal{L}_\mu(X^k, y^{k+1}, S), \quad S \succeq 0, \tag{3.2.5b}$$

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu}. \tag{3.2.5c}$$

The first-order optimality conditions for (3.2.5a) are

$$\nabla_y \mathcal{L}_\mu(X^k, y^{k+1}, S^k) := \mathcal{A}(X^k) - b + \frac{1}{\mu}\mathcal{A}(\mathcal{A}^*(y^{k+1}) + S^k - C) = 0.$$

Since by Assumption 3.2.1 $\mathcal{A}\mathcal{A}^*$ is invertible , we obtain $y^{k+1} := y(S^k, X^k)$, where

$$y(S,X) := -(\mathcal{A}\mathcal{A}^*)^{-1}\left(\mu(\mathcal{A}(X)-b)+\mathcal{A}(S-C)\right). \tag{3.2.6}$$

By rearranging the terms of $\mathcal{L}_\mu(X^k, y^{k+1}, S)$, it is easily verified that problem (3.2.5b) is equivalent to

$$\min_{S\in S^n} \quad \left\|S - V^{k+1}\right\|_F^2, \quad S \succeq 0, \tag{3.2.7}$$

where $V^{k+1} := V(S^k, X^k)$ and the function $V(S,X)$ is defined as

$$V(S,X) := C - \mathcal{A}^*(y(S,X)) - \mu X. \tag{3.2.8}$$

Hence, we obtain the solution $S^{k+1} := V_\dagger^{k+1} := Q_\dagger \Sigma_+ Q_\dagger^\top$, where

$$Q\Sigma Q^\top = \begin{pmatrix} Q_\dagger & Q_\ddagger \end{pmatrix} \begin{pmatrix} \Sigma_+ & \mathbf{0} \\ \mathbf{0} & \Sigma_- \end{pmatrix} \begin{pmatrix} Q_\dagger^\top \\ Q_\ddagger^\top \end{pmatrix}$$

is the spectral decomposition of the matrix V^{k+1}, and Σ_+ and Σ_- are the nonnegative and negative eigenvalues of V^{k+1}. It follows from the updating equation (3.2.5c) that

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu} = \frac{1}{\mu}(S^{k+1} - V^{k+1}) = \frac{1}{\mu} V_\ddagger^{k+1}, \tag{3.2.9}$$

where $V_{\ddagger}^{k+1} := -Q_{\ddagger}\Sigma_{-}Q_{\ddagger}^{\top}$. Note that X^{k+1} is also the optimal solution of

$$\min_{X \in S^n} \quad \left\| \mu X + V^{k+1} \right\|_F^2, \quad X \succeq 0. \tag{3.2.10}$$

From the above observation, we arrive at the alternating direction augmented Lagrangian method in Algorithm 6, below.

Algorithm 6: Alternating direction augmented Lagrangian method for SDP

Set $X^0 \succeq 0$ and $S^0 \succeq 0$.
for $k = 0, 1, \cdots$ **do**
  Compute y^{k+1} according to (3.2.6).
  Compute V^{k+1} and its eigenvalue decomposition, and set $S^{k+1} := V_{\dagger}^{k+1}$.
  Compute $X^{k+1} = \frac{1}{\mu}(S^{k+1} - V^{k+1})$.

Remark 3.2.1. In the boundary point method [46] and regularization method [42], X^k is fixed until $\frac{1}{\mu}(S^{k+1} - V^{k+1})$ is nearly feasible. However, the actual regularization method implemented in the numerical experiments in [42] is exactly Algorithm 6.

Remark 3.2.2. If $\mathcal{A}\mathcal{A}^* = I$, as is the case for many SDP relaxations of combinatorial optimization problems, such as the maxcut SDP relaxation, the SDP relaxation of the maximum stable set problem and the bisection SDP relaxation, step (3.2.5a), i.e., (3.2.6), is very inexpensive. If $\mathcal{A}\mathcal{A}^* \neq I$, we can compute $\mathcal{A}\mathcal{A}^*$ and its inverse (or its Cholesky factorization) prior to executing Algorithm 6. If computing the Cholesky factorization of $\mathcal{A}\mathcal{A}^*$ is very expensive, the strategies in [16, 20] can be used to compute an approximate minimizer y^{k+1} in step (3.2.5a).

3.2.2 Convergence analysis of Algorithm 6

Although the techniques for analyzing the convergence properties of the alternating direction methods for optimization problems arising from PDEs in [24], variational inequalities in [31] and a class of nonlinear optimization problems in [7] can be applied to our algorithm, we present here a different and simple argument by formulating our algorithm as a fixed point method. For any matrix $V \in S^n$, let the matrix $\left(V_\dagger, V_\ddagger\right)$ be denoted by $\mathcal{P}(V)$. Hence, each iteration of Algorithm 6 can be expressed as

$$y^{k+1} := y(S^k, X^k) \text{ and } \left(S^{k+1}, \mu X^{k+1}\right) := \mathcal{P}(V^{k+1}) = \mathcal{P}(V(S^k, X^k)). \tag{3.2.11}$$

We first describe a result of the orthogonal decomposition.

Lemma 3.2.2. *(Theorem 3.2.5 in [34], J.-J. Moreau) Let K be a closed convex cone and $K^\diamond$ be the polar cone of K, that is, $K^\diamond := \{s \in \mathbb{R}^n : \langle s, x\rangle \le 0$ for all $x \in K\}$. For the three elements x, x_1 and x_2 in $\mathbb{R}^n$, the properties below are equivalent:*

(i) *$x = x_1 + x_2$ with $x_1 \in K$ and $x_2 \in K^\diamond$ and $\langle x_1, x_2\rangle = 0$;*

(ii) *$x_1 = P_K(x)$ and $x_2 = P_{K^\diamond}(x)$,*

where $P_K(x)$ is the projection of x on K.

The next lemma shows that any optimal solution of (3.2.1) is a fixed point of the equations (3.2.11).

Lemma 3.2.3. *Suppose Assumption 3.2.1 holds. Then, there exist primal and dual optimal solutions (X, y, S) for* (3.2.1) *and* (3.2.2) *and the following two statements are equivalent:*

(i) (X,y,S) *satisfies the KKT optimality conditions*

$$\mathcal{A}(X) = b, \quad \mathcal{A}^*(y) + S = C, \quad SX = 0, \quad X \succeq 0, \quad Z \succeq 0.$$

(ii) (X,y,S) *satisfies*

$$y = y(S,X) \text{ and } \left(S, \mu X\right) = \mathcal{P}(V(S,X)).$$

Proof. The proof here is similar to Proposition 2.6 in [42]. Since the Slater condition holds, there exists primal and dual optimal solution (X,y,S) for (3.2.1) and (3.2.2) so that statement (i) is true. Direct algebraic manipulation shows that $y = y(S,X)$ from statement (i). It follows from Lemma 3.2.2 that

$$S - \mu X = V(S,X), \quad X \succeq 0, \quad S \succeq 0, \quad SX = 0,$$

is equivalent to $S = V_\dagger(S,X)$ and $\mu X = V_\ddagger(S,X)$. Hence, statement (i) implies statement (ii).

Now, suppose statement (ii) is true; i.e., $S = V_\dagger(S,X)$ and $\mu X = V_\ddagger(S,X)$. Since $S - \mu X = V(S,X)$, it follows from $V(S,X) = C - \mathcal{A}^*(y) - \mu X$ that $S = C - \mathcal{A}^*(y)$. From $y = y(S,X)$, we obtain

$$(\mathcal{A}\mathcal{A}^*)y = \mu(b - \mathcal{A}(X)) + \mathcal{A}(C - S) = \mu(b - \mathcal{A}(X)) + (\mathcal{A}\mathcal{A}^*)y,$$

which implies that $\mathcal{A}(X) = b$. Hence, statement (ii) implies statement (i). □

We now show that the operator $\mathcal{P}(V)$ is nonexpansive.

Lemma 3.2.4. *For any $V, \widehat{V} \in S^n$,*

$$\left\| \mathcal{P}(V) - \mathcal{P}(\widehat{V}) \right\|_F \leq \|V - \widehat{V}\|_F, \tag{3.2.12}$$

with equality holding if and only if $V_\dagger^\top \widehat{V}_\ddagger = 0$ and $V_\ddagger^\top \widehat{V}_\dagger = 0$.

Proof. We denote $V_\dagger - \widehat{V}_\dagger$ by $W_\dagger$ and $V_\ddagger - \widehat{V}_\ddagger$ by $W_\ddagger$. Since $V_\dagger^\top V_\ddagger = 0$ and $\widehat{V}_\dagger^\top \widehat{V}_\ddagger = 0$, we obtain $-W_\dagger^\top W_\ddagger = V_\dagger^\top \widehat{V}_\ddagger + \widehat{V}_\dagger^\top V_\ddagger$. The positive semidefinitess of the matrices $V_\dagger, V_\ddagger, \widehat{V}_\dagger, \widehat{V}_\ddagger \succeq 0$ implies that $\mathbf{Tr}(V_\dagger^\top \widehat{V}_\ddagger) \geq 0$ and $\mathbf{Tr}(\widehat{V}_\dagger^\top V_\ddagger) \geq 0$. Expanding terms of $V - \widehat{V}$, we obtain

$$\begin{aligned}
\left\| V - \widehat{V} \right\|_F^2 &= \mathbf{Tr}\left((W_\dagger - W_\ddagger)^\top (W_\dagger - W_\ddagger) \right) = \mathbf{Tr}\left(W_\dagger^\top W_\dagger + W_\ddagger^\top W_\ddagger \right) - 2\,\mathbf{Tr}\left(W_\dagger^\top W_\ddagger \right) \\
&= \left\| \mathcal{P}(V) - \mathcal{P}(\widehat{V}) \right\|_F^2 + 2\,\mathbf{Tr}\left(V_\dagger^\top \widehat{V}_\ddagger + \widehat{V}_\dagger^\top V_\ddagger \right) \\
&\geq \left\| \mathcal{P}(V) - \mathcal{P}(\widehat{V}) \right\|_F^2,
\end{aligned}$$

which proves (3.2.12). □

The following lemma shows that the operator $V(S,X)$ is nonexpansive.

Lemma 3.2.5. *For any $S, X, \widehat{S}, \widehat{X} \in S_+^n$,*

$$\|V(S,X) - V(\widehat{S},\widehat{X})\|_F \leq \left\| \left(S - \widehat{S}, \mu(X - \widehat{X}) \right) \right\|_F \tag{3.2.13}$$

with equality holding if and only if $V - \widehat{V} = (S - \widehat{S}) - \mu(X - \widehat{X})$.

Proof. From the definition of $y(S,X)$ in (3.2.6), we have

$$y(\widehat{S},\widehat{X})-y(S,X) \;=\; (\mathcal{A}\mathcal{A}^*)^{-1}\left(\mu\mathcal{A}(X-\widehat{X})+\mathcal{A}(S-\widehat{S})\right),$$

which together with (3.2.8) gives

$$\begin{aligned} V(S,X)-V(\widehat{S},\widehat{X}) &= (C-\mathcal{A}^*(y(S,X))-\mu X)-(C-\mathcal{A}^*(y(\widehat{S},\widehat{X}))-\mu\widehat{X}) \\ &= \mathcal{A}^*(y(\widehat{S},\widehat{X})-y(S,X))+\mu(\widehat{X}-X) \\ &= \mathrm{mat}\left(-\mu(I-M)\,\mathrm{vec}(X-\widehat{X})+M\mathrm{vec}(S-\widehat{S})\right), \end{aligned} \tag{3.2.14}$$

where $M := A^\top(AA^\top)^{-1}A$. Since M is an orthogonal projection matrix whose spectral radius is 1, we obtain from (3.2.14) that

$$\begin{aligned} \left\|V(S,X)-V(\widehat{S},\widehat{X})\right\|_F^2 &= \left\|\mu(I-M)\,\mathrm{vec}(X-\widehat{X})-M\mathrm{vec}(S-\widehat{S})\right\|_2^2 \\ &\le \|\mu\mathrm{vec}(X-\widehat{X})\|_2^2+\|\mathrm{vec}(S-\widehat{S})\|_2^2 \\ &= \left\|\left(S-\widehat{S},\mu(X-\widehat{X})\right)\right\|_F^2, \end{aligned} \tag{3.2.15}$$

which proves (3.2.13).

If the equality in (3.2.13) holds, it also holds in (3.2.15); that is,

$$\|\mu(I-M)\mathrm{vec}(X-\widehat{X})\|_2^2+\|M\mathrm{vec}(S-\widehat{S})\|_2^2=\|\mu\mathrm{vec}(X-\widehat{X})\|_2^2+\|\mathrm{vec}(S-\widehat{S})\|_2^2.$$

This implies that $M\mathrm{vec}(X-\widehat{X})=0$ and $(I-M)\mathrm{vec}(S-\widehat{S})=0$. Using this relations in

(3.2.14), we obtain

$$V(S,X) - V(\widehat{S},\widehat{X}) = \mathrm{mat}\left(\mathrm{vec}(S-\widehat{S}) - \mu\mathrm{vec}(X-\widehat{X})\right),$$

which proves the second statement. □

Lemma 3.2.6. *Let* (X^*,y^*,S^*)*, where* $y^* = y(S^*,X^*)$*, be an optimal solution of* (3.2.1) *and* (3.2.2)*. Under Assumption 3.2.1, if*

$$\|\mathcal{P}(V(S,X)) - \mathcal{P}(V(S^*,X^*))\|_F = \left\|\left(S-S^*, \mu(X-X^*)\right)\right\|_F, \tag{3.2.16}$$

then, $(S,\mu X)$ *is a fixed point, that is,* $(S,\mu X) = \mathcal{P}(V(S,X))$*, and hence,* (X,y,S)*, where* $y = y(S,X)$*, is a primal and dual optimal solutions of* (3.2.1) *and* (3.2.2)*.*

Proof. From Lemma 3.2.3, we have $(S^*,\mu X^*) = \mathcal{P}(V(S^*,X^*))$. From Lemmas 3.2.4 and 3.2.5, we have

$$V(S,X) - V(S^*,X^*) = (S-S^*) - \mu(X-X^*),$$

which implies that $V(S,X) = S - \mu X$. Since S and X are all positive semidefinite, and $S^\top X = 0$, we obtain from Lemma 3.2.2 that $(S,\mu X) = \mathcal{P}(V(S,X))$. □

Given Lemmas 3.2.4, 3.2.5 and 3.2.6, we can prove convergence of Algorithm 6 by following the proof of Theorem 4.5 in [30].

Theorem 3.2.7. *The sequence* $\{(X^k,y^k,S^k)\}$ *generated by Algorithm 6 from any starting*

point (X^0, y^0, S^0) *converges to a solution* $(X^*, y^*, S^*) \in \mathcal{X}^*$*, where* $\mathcal{X}^*$ *is the set of primal and dual solutions of* (3.2.1) *and* (3.2.2)*.*

Proof. Since both $\mathcal{P}(\cdot)$ and $V(\cdot,\cdot)$ are non-expansive, $\mathcal{P}(V(\cdot,\cdot))$ is also nonexpansive. Therefore, $\{(S^k, \mu X^k)\}$ lies in a compact set and must have a limit point, say $\bar{S} = \lim_{j\to\infty} S^{k_j}$ and $\bar{X} = \lim_{j\to\infty} X^{k_j}$. Also, for any $(X^*, y^*, S^*) \in \mathcal{X}^*$,

$$\begin{aligned}
\left\|(S^{k+1}, \mu X^{k+1}) - (S^*, \mu X^*)\right\|_F &= \left\|\mathcal{P}(V(S^k, \mu X^k)) - \mathcal{P}(V(S^*, \mu X^*))\right\|_F \\
&\leq \left\|V(S^k, \mu X^k) - V(S^*, \mu X^*)\right\|_F \\
&\leq \left\|(S^k, \mu X^k) - (S^*, \mu X^*)\right\|_F,
\end{aligned}$$

which means that the sequence $\{\|(S^k, \mu X^k) - (S^*, \mu X^*)\|_F\}$ is monotonically non-increasing. Therefore,

$$\lim_{k\to\infty} \left\|(S^k, \mu X^k) - (S^*, \mu X^*)\right\|_F = \left\|(\bar{S}, \mu\bar{X}) - (S^*, \mu X^*)\right\|_F, \tag{3.2.17}$$

where $(\bar{S}, \mu\bar{X})$ can be any limit point of $\{(S^k, \mu X^k)\}$. By the continuity of $\mathcal{P}(V(\cdot,\cdot))$, the image of $(\bar{S}, \mu\bar{X})$,

$$\mathcal{P}(V(\bar{S}, \mu\bar{X})) = \lim_{j\to\infty} \mathcal{P}(V(S^{k_j}, \mu X^{k_j})) = \lim_{j\to\infty} (S^{k_j+1}, \mu X^{k_j+1}),$$

is also a limit of $\{(S^k, \mu X^k)\}$. Therefore, we have

$$\left\|\mathcal{P}(V(\bar{S}, \mu\bar{X})) - \mathcal{P}(V(S^*, \mu X^*))\right\|_F = \left\|(\bar{S}, \mu\bar{X}) - (S^*, \mu X^*)\right\|_F,$$

which allows us to apply Lemma 3.2.6 to get that $(\bar{S}, \bar{y}, \mu\bar{X})$, where $\bar{y} = y(\bar{S}, \bar{X})$, is an optimal solution to problems (3.2.1) and (3.2.2). Finally, by setting $(S^*, \mu X^*) = (\bar{S}, \mu\bar{X})$ in (3.2.17), we get that

$$\lim_{k\to\infty} \left\| (S^k, \mu\bar{X}^k) - (\bar{S}, \mu\bar{X}) \right\|_F = \lim_{k\to\infty} \left\| (S^{k_j}, \mu\bar{X}^{k_j}) - (\bar{S}, \mu\bar{X}) \right\|_F = 0,$$

i.e., $\{(S^k, \mu X^k)\}$ converges to its unique limit of $(\bar{S}, \mu\bar{X})$. □

We now describe the relationship between the primal infeasibility $\|\mathcal{A}(X^{k+1}) - b\|_2$ and dual infeasibility $\|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_F$ and the difference between the matrices $\{V^k\}$.

Corollary 3.2.8. *Let $\{(X^k, y^k, S^k)\}$ be a sequence generated by Algorithm 6. Then*

1. $\mathcal{A}(X^{k+1}) - b = \frac{1}{\mu}\mathcal{A}(S^{k+1} - S^k)$ *and* $\|\mathcal{A}(X^{k+1}) - b\|_2 \le \frac{\|A\|_2}{\mu}\|V^{k+1} - V^k\|_F$.

2. $C - \mathcal{A}^*(y^{k+1}) - S^{k+1} = \mu(X^k - X^{k+1})$ *and* $\|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_F \le \|V^{k+1} - V^k\|_F$.

3. $\|V^{k+1} - V^k\|_F \le \|V^k - V^{k-1}\|_F$.

Proof. 1). It follows from (3.2.6), (3.2.8) and Assumption 3.2.1 that

$$\begin{aligned}
\mathcal{A}(X^{k+1}) - b &= \frac{1}{\mu}\mathcal{A}(S^{k+1} - V^{k+1}) - b \\
&= \frac{1}{\mu}\mathcal{A}(S^{k+1} - C) + \mathcal{A}(X^k) - b + (b - \mathcal{A}(X^k)) + \frac{1}{\mu}\mathcal{A}(C - S^k) \\
&= \frac{1}{\mu}\mathcal{A}(S^{k+1} - S^k).
\end{aligned}$$

Since the projection $V_\dagger^{k+1}$ is nonexpansive, we obtain

$$\|\mathcal{A}(X^{k+1}) - b\|_2 \leq \frac{\|A\|_2}{\mu}\|\mathrm{vec}(S^{k+1} - S^k)\|_2 \leq \frac{\|A\|_2}{\mu}\|V^{k+1} - V^k\|_F.$$

2) Rearranging the terms of (3.2.5c), we obtain the first part of statement 2. The nonexpansiveness of the projection $V_\ddagger^{k+1}$ gives

$$\|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_2 = \|V_\ddagger^{k+1} - V_\ddagger^k\|_F \leq \|V^{k+1} - V^k\|_F.$$

3) From Lemmas 3.2.4 and 3.2.5, we obtain

$$\begin{aligned}
\|V^{k+1} - V^k\|_F &\leq \left\| \left(S^k - S^{k-1}, \mu(X^k - X^{k-1}) \right) \right\|_F \\
&= \left\| \left(V_\dagger^k - V_\dagger^{k-1}, V_\ddagger^k - V_\ddagger^{k-1} \right) \right\|_F \\
&\leq \|V^k - V^{k-1}\|_F.
\end{aligned}$$

□

3.2.3 A multiple-splitting scheme for an expanded problem

Although SDP problems with inequality constraints can be put into the standard form (3.2.1), it is often more convenient to treat the inequality constraints directly in order to preserve special structure of the constraints, such as sparsity and orthogonality. We now extend our alternating direction method (3.2.5a)-(3.2.5c) to solve an expanded SDP prob-

lem, that includes, in particular, positivity constraints on the elements of the matrix X; i.e.,

$$\min_{X\in S^n} \langle C,X\rangle, \text{ s.t. } \mathcal{A}(X)=b, \quad \mathcal{B}(X)\geq d, \quad X\succeq 0, \quad X\geq 0, \tag{3.2.18}$$

where $d\in\mathbb{R}^q$ and the linear map $\mathcal{B}(\cdot): S^n\to\mathbb{R}^q$ is defined as

$$\mathcal{B}(X) := \left(\left\langle B^{(1)},X\right\rangle,\cdots,\left\langle B^{(q)},X\right\rangle\right)^\top, \quad B^{(i)}\in S^n, i=1,\cdots,q. \tag{3.2.19}$$

As we did for the operator $\mathcal{A}$, we define the operators $\mathcal{B}^*$, $\mathcal{B}\mathcal{B}^*$ and $\mathcal{B}^*\mathcal{B}$ by introducing

$$B=\left(\text{vec}(B^{(1)}),\cdots,\text{vec}(B^{(q)})\right)^\top\in\mathbb{R}^{q\times n^2}.$$

We also make the following assumption.

Assumption 3.2.9. *The matrices A and B have full row rank and a refined Slater condition holds for* (3.2.18)*; that is, there exists a positive definite matrix* $\bar{X}$ *satisfying* $\mathcal{A}(\bar{X})=b$, $\mathcal{B}(\bar{X})\geq d$ *and* $\bar{X}\geq 0$.

The dual of problem (3.2.18) is

$$\begin{aligned}\min_{y\in\mathbb{R}^m,v\in\mathbb{R}^q,S\in S^n,Z\in S^n} \quad & -b^\top y-d^\top v,\\ \text{s.t.} \quad & \mathcal{A}^*(y)+\mathcal{B}^*(v)+S+Z=C,\\ & v\geq 0, \quad S\succeq 0, \quad Z\geq 0.\end{aligned} \tag{3.2.20}$$

The augmented Lagrangian function for the dual SDP (3.2.20) corresponding to the linear

constraints is defined as:

$$\begin{aligned}\mathcal{L}_\mu(X,y,v,Z,S) &:= -b^\top y - d^\top v + \langle X, \mathcal{A}^*(y) + \mathcal{B}^*(v) + S + Z - C\rangle \\ &\quad + \frac{1}{2\mu}\|\mathcal{A}^*(y) + \mathcal{B}^*(v) + S + Z - C\|_F^2,\end{aligned} \tag{3.2.21}$$

where $X \in S^n$ and $\mu > 0$. Starting from $X^0 \succeq 0$, $v^0 \geq 0$, $Z^0 \geq 0$ and $S^0 \succeq 0$, our alternating direction method computes new iterates similar to the procedure (3.2.5a)-(3.2.5c) as follows

$$y^{k+1} := \arg\min_{y\in\mathbb{R}^m} \mathcal{L}_\mu(X^k, y, v^k, Z^k, S^k), \tag{3.2.22a}$$

$$v^{k+1} := \arg\min_{v\in\mathbb{R}^q} \mathcal{L}_\mu(X^k, y^{k+1}, v, Z^k, S^k), \quad v \geq 0, \tag{3.2.22b}$$

$$Z^{k+1} := \arg\min_{Z\in S^n} \mathcal{L}_\mu(X^k, y^{k+1}, v^{k+1}, Z, S^k), \quad Z \geq 0, \tag{3.2.22c}$$

$$S^{k+1} := \arg\min_{S\in S^n} \mathcal{L}_\mu(X^k, y^{k+1}, v^{k+1}, Z^{k+1}, S), \quad S \succeq 0, \tag{3.2.22d}$$

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + \mathcal{B}^*(v^{k+1}) + S^{k+1} + Z^{k+1} - C}{\mu}. \tag{3.2.22e}$$

Note that y^{k+1}, S^{k+1} and X^{k+1} can be computed in the same fashion as in (3.2.6), (3.2.7) and (3.2.9), respectively. By rearranging the terms of $\mathcal{L}_\mu(X^k, y^{k+1}, v, Z^k, S^k)$, it is easily verified that problem (3.2.22b) is equivalent to the strictly convex quadratic program

$$\min_{v\in\mathbb{R}^n} \quad \left(\mathcal{B}\left(X^k + \frac{1}{\mu}Y^{k+1}\right) - d\right)^\top v + \frac{1}{2\mu}v^\top(\mathcal{B}\mathcal{B}^*)v, \quad v \geq 0, \tag{3.2.23}$$

where $Y^{k+1} := \mathcal{A}^*(y^{k+1}) + S^k + Z^k - C$. By rearranging the terms of $\mathcal{L}_\mu(X^k, y^{k+1}, v^{k+1}, Z, S^k)$,

it is easily verified that problem (3.2.22c) is equivalent to

$$\min_{Z \in S^n} \quad \left\| Z - U^{k+1} \right\|_F^2, \quad Z \geq 0,$$

where $U^{k+1} := C - \mathcal{A}^*(y^{k+1}) - \mathcal{B}^*(v^{k+1}) - S^k - \mu X^k$. Hence, the solution of problem (3.2.22c) is $Z^{k+1} = U_+^{k+1}$, the positive part of U^{k+1}, that is, $(U_+^{k+1})_{i,j} := \max(U_{ij}^{k+1}, 0)$.

3.3 Practical Issues

In this section, we discuss practical issues related to the eigenvalue decomposition performed at each iteration, strategies for adjusting the penalty parameter, the use of a step size for updating the primal variable X, termination rules and how to detect stagnation to enhance the performance of our methods. Our focus is on procedures (3.2.5a)-(3.2.5c) for problem (3.2.1), but our discussion applies equally to the expanded problem (3.2.18).

3.3.1 Eigenvalue decomposition

One of the bottlenecks of our alternating direction method is the computation of the eigenvalue decomposition. Fortunately, for many problems in practice, either the primal solution X or the dual solution S is a low rank matrix. For example, the primal variable X in the max-cut SDP relaxation often has low rank while the dual variable S in frequency assignment problem often has low rank. Moreover, since the optimal solution pair (X, y, S) satisfies the complementary condition $XS = 0$, the matrices X and S share the same set of eigenvectors

and the positive eigenvalues of X (S) correspond to zero eigenvalues of S (X). Therefore, we only need to compute either $V_\dagger^k$, the part corresponding to the positive eigenvalues of V^k, or $V_\ddagger^k$, the part corresponding to the negative eigenvalues of V^k, at iteration k. Specifically, the following adaptive scheme can be used at the end of iteration $k-1$ to decide whether $V_\dagger^k$ or $V_\ddagger^k$ should be computed. Suppose that $V_\dagger^{k-1}$ has been computed and let $\kappa_+(V^{k-1})$ be the total number of the positive eigenvalues of V^{k-1}. If $\kappa_+(V^{k-1}) \leq \frac{n}{2}$, this suggests that S^k might have low rank and we compute $V_\dagger^k$. Otherwise, it is possible that X^k has low rank and we compute $V_\ddagger^k$. If the total number of the negative eigenvalues $\kappa_-(V^{k-1})$ of V^{k-1} is known, a similar strategy can be used.

There are two types of methods, direct and iterative, for computing selected eigenvalues and eigenvectors of a real symmetric matrix V. Direct methods, which reduce V to tridiagonal form and then compute the required eigenvalues and eigenvectors from the tridiagonal matrix, are suitable for small and medium sized matrices. Since n is less than 5000 in our numerical experiments, the code "DSYEVX" in LAPACK works fairly well. Iterative methods, such as the Lanczos algorithm (for example, "ARPACK"), require only matrix-vector products and hence they are suitable for sparse matrices or matrices for which the required matrix-vector products are cheap. If the matrices C and $\mathcal{A}^*(y)$ are sparse or have low rank, then advantage can be taken of these structures since $V := C - \mathcal{A}^*(y) - \mu X$.

Note from Corollary 3.2.8 that the primal infeasibility $\|\mathcal{A}(X^{k+1}) - b\|_2$ and dual infeasibility $\|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_F$ are bounded by the difference $\|V^k - V^{k+1}\|_F$ which is non-increasing. Hence, when the alternative direction method converges slowly, $\|V^k - V^{k+1}\|_F$ is often quite small. Hence, the spectral decomposition of V^{k+1} is close to that of V^k. How-

ever, neither the direct nor the iterative methods mentioned above can take advantage of a good initial guess. Assume that $V_\dagger^k := Q_\dagger^k \Sigma_+^k (Q_\dagger^k)^\top$ has low rank. Since $V_\dagger^k$ is the optimal solution of $\min_{S \succeq 0} \|S - V^k\|_F^2$, we can use nonlinear programming approaches, such as the limited-memory BFGS method, to obtain $R^{k+1} := \arg\min_{R \in \mathbb{R}^{n \times \kappa}} \|RR^\top - V^{k+1}\|_F^2$ starting from $R := Q_\dagger^k (\Sigma_+^k)^{\frac{1}{2}}$, and set $S^{k+1} := R^{k+1}(R^{k+1})^\top$, where $\kappa := \kappa_+(V^k)$. Similarly, since $V_\ddagger^k$ is the optimal solution of $\min_{Y \succeq 0} \|Y + V^k\|_F^2$, we can compute $V_\ddagger^{k+1}$ from the optimal solution of $\min_{R \in \mathbb{R}^{n \times \kappa}} \|RR^\top + V^{k+1}\|_F^2$, where $\kappa := \kappa_-(V^k)$.

3.3.2 Updating the penalty parameter

Although our analysis shows that our alternating direction method Algorithm 6 converges for any fixed penalty parameter $\mu > 0$, numerical performance can be improved by adjusting the value of μ. We next present a strategy for doing this dynamically. Since the complementary condition $X^k S^k = 0$ is always satisfied, the pair (X^k, y^k, S^k) is close to optimal if the primal and dual infeasibilities $\|\mathcal{A}(X^{k+1}) - b\|_2$ and $\|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_F$ are small. Hence, we can tune μ so that the primal and dual infeasibilities are balanced, that is, $\|\mathcal{A}(X^{k+1}) - b\|_2 \approx \|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_F$. Specifically, we have from Corollary 3.2.8 that

$$\mathcal{A}(X^{k+1}) - b = \frac{1}{\mu}\mathcal{A}(S^{k+1} - S^k) \text{ and } C - \mathcal{A}^*(y^{k+1}) - S^{k+1} = \mu(X^k - X^{k+1}),$$

which suggests that the primal and dual infeasibilities are proportional to $\frac{1}{\mu}$ and μ, respectively. Therefore, we decrease (increase) μ by a factor γ ($\frac{1}{\gamma}$), $0 < \gamma < 1$, if the primal

infeasibility is less than (greater than) a multiple of the dual infeasibility for a number of consecutive iterations. In addition, μ is required to remain within an interval $[\mu_{\min}, \mu_{\max}]$, where $0 < \mu_{\min} < \mu_{\max} < \infty$.

3.3.3 Step size for updating the primal variable X

For many alternating direction methods [16, 22, 24, 31, 36, 38, 54], numerical performance is often improved if a step size is added to the update of the Lagrange multiplier. Here, we replace step (3.2.5c) by

$$\begin{aligned} X^{k+1} &:= X^k + \rho \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu} \\ &= (1-\rho)X^k + \frac{\rho}{\mu}(S^{k+1} - V^{k+1}) \\ &:= (1-\rho)X^k + \rho \bar{X}^{k+1}, \end{aligned} \tag{3.3.1}$$

where $\rho \in (0, \frac{1+\sqrt{5}}{2})$ and $\bar{X}^{k+1} := \frac{1}{\mu}(S^{k+1} - V^{k+1})$. Convergence of this variant of the algorithm can be proved in the same fashion as in [31]. Specifically, the following property holds.

Theorem 3.3.1. *Let* (X^*, y^*, S^*) *be an optimal solution of* (3.2.1) *and* (3.2.2), $\rho \in (0, \frac{1+\sqrt{5}}{2})$

and $T = 2 - \frac{1}{2}(1+\rho-\rho^2)$. *Then, we have*

$$\begin{aligned} &\|X^{k+1}-X^*\|_F^2 + \tfrac{\rho}{\mu^2}\|S^{k+1}-S^*\|_F^2 + \tfrac{\rho(T-\rho)}{\mu^2}\|\mathcal{A}^*(y^{k+1})+S^{k+1}-C\|_F^2 \\ &\le \|X^k-X^*\|_F^2 + \tfrac{\rho}{\mu^2}\|S^k-S^*\|_F^2 + \tfrac{\rho(T-\rho)}{\mu^2}\|\mathcal{A}^*(y^k)+S^k-C\|_F^2 \\ &-\tfrac{(1+\rho-\rho^2)\rho}{3\mu^2}\left(\|\mathcal{A}^*(y^{k+1})+S^{k+1}-C\|_F^2 + \|S^k-S^{k+1}\|_F^2\right). \end{aligned} \tag{3.3.2}$$

Hence, we obtain

$$\lim_{k\to\infty}\left(\|\mathcal{A}^*(y^{k+1})+S^{k+1}-C\|_F^2 + \|S^k-S^{k+1}\|_F^2\right) = 0. \tag{3.3.3}$$

Based on Theorem 3.3.1, we can show that both the primal and dual infeasibilities and the violation of the complementary condition converge to zero. From statement 1 of Corollary (3.2.8), we have

$$\begin{aligned} \|\mathcal{A}(X^{k+1})-b\|_2 &\le \|\mathcal{A}(\bar{X}^{k+1})-b\|_2 + \|\mathcal{A}(X^{k+1})-\mathcal{A}(\bar{X}^{k+1})\|_2 \\ &\le \frac{1}{\mu}\|A\|_2\,\|S^{k+1}-S^k\|_F + \|A\|_2\,\|X^{k+1}-\bar{X}^{k+1}\|_F. \end{aligned} \tag{3.3.4}$$

We obtain from (3.3.1) that

$$\begin{aligned} \|X^{k+1}S^{k+1}\|_F &= (1-\rho)\|X^kS^{k+1}\|_F \qquad (3.3.5)\\ &\leq (1-\rho)\left(\|(X^k-\bar{X}^{k+1})S^{k+1}\|_F+\|\bar{X}^{k+1}S^{k+1}\|_F\right)\\ &= (1-\rho)\|(X^k-\bar{X}^{k+1})S^{k+1}\|_F\\ &\leq (1-\rho)\|(X^k-\bar{X}^{k+1})\|_F\|S^{k+1}\|_F. \end{aligned}$$

It follows from (3.3.1) and (3.3.3) that

$$\lim_{k\infty}\|X^{k+1}-X^k\|_F=0,\ \lim_{k\infty}\|\bar{X}^{k+1}-X^k\|_F=0,\ \lim_{k\infty}\|X^{k+1}-\bar{X}^{k+1}\|_F=0. \qquad (3.3.6)$$

Combining (3.3.3)-(3.3.6), we obtain

$$\begin{aligned} &\lim_{k\to\infty} \quad \|\mathcal{A}^*(y^{k+1})+S^{k+1}-C\|_F^2=0,\\ &\lim_{k\to\infty} \quad \|\mathcal{A}(X^{k+1})-b\|_F^2=0,\\ &\lim_{k\to\infty} \quad \|X^{k+1}S^{k+1}\|_F=0. \end{aligned}$$

3.3.4 Termination rules and detection of stagnation

Since the rate of convergence of first-order methods can slow down as the iterates approach an optimal solution, it is critical to detect this stagnation and stop properly. However, it is difficult to predict whether an algorithm can get out of a region in which it is temporarily trapped and then resume a fast rate of convergence. Hence, it is usually beneficial to allow

some flexibility in the termination rules. Similarly to the rules used in the Seventh DIMACS Implementation Challenge, we measure the infeasibilities and closeness to optimality for the primal and dual problems as follows

$$\text{pinf} = \frac{\|\mathcal{A}(X) - b\|_2}{1 + \|b\|_2}, \quad \text{dinf} = \frac{\|C + S + Z - \mathcal{A}^*(y)\|_F}{1 + \|C\|_1}, \quad \text{gap} = \frac{|b^\top y - \langle C, X\rangle|}{1 + |b^\top y| + \langle C, X\rangle}. \tag{3.3.7}$$

We stop our algorithm when

$$\delta := \max\{\text{pinf, dinf, gap}\} \leq \varepsilon,$$

for $\varepsilon > 0$. We use "it_stag" to count the number of consecutive iterations that δ does not decrease below the best value obtained thus far and stop if the following criteria are satisfied:

$$\begin{aligned}(\text{it\_stag} > h_1 \text{ and } \delta <= 10\varepsilon) \text{ or } (\text{it\_stag} > h_2 \text{ and } \delta <= 10^2\varepsilon) \\ \text{or } (\text{it\_stag} > h_3 \text{ and } \delta <= 10^3\varepsilon),\end{aligned} \tag{3.3.8}$$

where $1 < h_1 < h_2 < h_3$ are integers representing different levels of difficulty.

The complete pseudo-code for our algorithm SDPAD (SDP Alternating Direction) is presented in Algorithm 7. SDPAD uses eleven parameters, the values of only a few of which are critical to its convergence and performance. The default value of the termination tolerance ε is 10^{-6}, and h_1, h_2 and h_3 are used to detect stagnation. The following parameters are for adjusting the penalty parameter μ: the initial value of μ is 5, and its minimal and

maximal values $\mu_{\min}$ and $\mu_{\max}$ are set to 10^{-4} and 10^{4}, respectively, the factor for reducing (increasing) μ is $\gamma = 0.5$, and this is done if there are h_4 consecutive iterations in which the ratio of the primal to the dual infeasibility is less than or equal to η_1 (greater than η_2). The step size ρ for updating X is set to 1.6. We choose the initial iterate $X^0 = I$ and $S^0 = \mathbf{0}$. Note that SDPAD can be extended naturally to the expanded problem (3.2.18); we shall also refer to the resulting algorithm as SDPAD .

Algorithm 7: SDPAD

Set $0 < \mu_{\min} \le \mu \le \mu_{\max} < +\infty$, $\varepsilon > 0$, $\gamma \in (0,1)$, $0 < \eta_1 \le \eta_2 < \infty$, $\rho \in (0, \frac{1+\sqrt{5}}{2})$, $1 < h_1 < h_2 < h_3$ and $h_4 > 1$. Set $X^0 \succeq 0$ and $S^0 \succeq 0$. Set $eigS = true$, $ref = +\infty$, $it\_stag = 0$, $it\_pinf = 0$ and $it\_dinf = 0$.

for $k = 0, 1, \cdots$ **do**

S1 *Update y^{k+1}, S^{k+1} and X^{k+1}:*

Compute y^{k+1} according to (3.2.6) and V^{k+1} according to (3.2.8).

if *eigS == true* **then**

Compute $V_\dagger^{k+1}$, set $S^{k+1} := V_\dagger^{k+1}$ and $X^{k+1} = \frac{1}{\mu}(S^{k+1} - V^{k+1})$.

if $\kappa(V_\dagger^{k+1}) \ge \frac{n}{2}$ **then** set $eigS = false$.

else

Compute $V_\ddagger^{k+1}$, set $S^{k+1} := V^{k+1} + V_\ddagger^{k+1}$ and $X^{k+1} = \frac{1}{\mu} V_\ddagger^{k+1}$.

if $\kappa(V_\ddagger^{k+1}) \ge \frac{n}{2}$ **then** set $eigS = true$.

Set $X^{k+1} := (1-\rho)X^k + \rho X^{k+1}$.

S2 *Check optimality and detect stagnation:*

Compute $\delta := \max\{\text{pinf, dinf, gap}\}$.

if $\delta \le \varepsilon$ **then** return the solution.

if $\delta \le ref$ **then** set $ref := \delta$ and $it\_stag = 0$ **else** set $it\_stag = it\_stag + 1$.

if *condition* (3.3.8) *is satisfied* **then** return the solution.

S3 *Update penalty parameter μ:*

if $pinf/dinf \le \eta_1$ **then**

Set $it\_pinf = it\_pinf + 1$ and $it\_dinf = 0$.

if $it\_pinf \ge h_4$ **then** set $\mu = \max(\gamma\mu, \mu_{\min})$ and $it\_pinf = 0$.

else if $pinf/dinf > \eta_2$ **then**

Set $it\_dinf = it\_dinf + 1$ and $it\_pinf = 0$.

if $it\_dinf \ge h_4$ **then** set $\mu = \min(\frac{1}{\gamma}\mu, \mu_{\max})$ and $it\_dinf = 0$.

3.4 Numerical Results

Although the numerical results that we present in this section are limited to three special classes of SDP problems, they illustrate the effectiveness of our alternating direction methods. The main parts of our code were written in C Language MEX-files in MATLAB (Release 7.3.0), and all experiments were performed on a Dell Precision 670 workstation with an Intel Xeon 3.4GHZ CPU and 6GB of RAM.

3.4.1 Frequency assignment relaxation

In this subsection, we demonstrate the effectiveness of SDPAD on the SDP relaxations of frequency assignment problems and compare the results with results obtained using the code SDPNAL [64]. These problems (see equations (4) and (5) on page 363 in [13], or equation (2) on page 5 of [45]) arise from wireless communication networks and contain both equality and inequality constraints. Let $G = (V,E)$ be an undirected graph with vertex set $V = \{1,\cdots,r\}$ and edge set $E \subseteq V \times V$, and let $W \in S^r$ be a weight matrix for G such that $w_{i,j} = w_{j,i}$ is the weight associated with edge $(i,j) \in E$. For those edges $(i,j) \notin E$, we assume $w_{i,j} = w_{j,i} = 0$. Let $T \subseteq E$ be a given edge subset. These problems can be

formulated as:

$$\begin{aligned} \min \quad & \left\langle \tfrac{1}{2k}\mathrm{diag}(We) + \tfrac{k-1}{2k}W, X \right\rangle \\ s.t. \quad & X_{i,j} \geq \tfrac{-1}{k-1}, \quad \forall (i,j) \in E \backslash T, \\ & X_{i,j} = \tfrac{-1}{k-1}, \quad \forall (i,j) \in T, \\ & \mathrm{diag}(X) = e, \quad X \succeq 0. \end{aligned} \tag{3.4.1}$$

Using the matrices corresponding to the edges in T and the constraint $\mathrm{diag}(X) = e$ to construct the operator $\mathcal{A}$, and the matrices corresponding to the edges in $E \backslash T$ to construct the operator $\mathcal{B}$, we can formulate (3.4.1) as the expanded problem (3.2.18) without the positivity constraints $X \geq 0$. We replaced the constraints $X_{ij} = \frac{-1}{k-1}$ by $X_{ij}/\sqrt{2} = \frac{-1}{\sqrt{2}(k-1)}$ and $X_{ij} \geq \frac{-1}{k-1}$ by $X_{ij}/\sqrt{2} \geq \frac{-1}{\sqrt{2}(k-1)}$, hence, $\mathcal{A}\mathcal{A}^* = I$, $\mathcal{B}\mathcal{B}^* = I$ and $\mathcal{A}$ and $\mathcal{B}$ are orthogonal to each other. Therefore, the optimal solution of the subproblems (3.2.22a) and (3.2.22b) are explicitly available:

$$y^{k+1} := -\left(\mu(\mathcal{A}(X^k) - b) + \mathcal{A}(S^k - C)\right)$$

and

$$v^{k+1} := \max\left(-\left(\mu(\mathcal{B}(X^k) - d) + \mathcal{B}(S^k - C)\right), \mathbf{0}\right).$$

To accommodate the inequality constraints, the primal infeasibility was measured by

$$\mathrm{pinf} = \frac{\|\mathcal{A}(X) - b\|_2 + \|\min(\mathcal{B}(X) - d, \mathbf{0})\|_2}{1 + \|b\|_2}.$$

Since the matrices corresponding to the operators $\mathcal{A}$ and $\mathcal{B}$ do not have to be stored, the memory required by our implementation is quite small.

The parameters of SDPNAL were set to their default values. The iteration counter set points h_1, h_2 and h_3 were set to 20, 150 and 300, respectively, the iteration counter h_4 for changing μ was set to 50 and the ratios η_1 and η_2 were set to 1. We stopped SDPAD when the total number of iterations reached 2000. All other parameters were set to their default values. A summary of the computational results is presented in Table 3.1. In that table, $\hat{m}$ denotes the total number $m+q$ of equality and inequality constraints, "itr" denotes the total number of iterations performed and "cpu" denotes the CPU time reported in the format of "hours:minutes:seconds". Since running SDPNAL on "fap25" and "fap36" is very time consuming (for example, in the results reported in [64], SDPNAL took more than 65 hours to solve problem "fap36"), we did not run SDPNAL on our own computer on these two problems and the results presented here were taken from Table 3 in [64]. Note that the numerical results of SDPNAL on problems from "fap01" to "fap12" in Table 3.1 are slightly different from those reported in Table 3 in [64]. Since the results in [64] were obtained from a PC with Intel Xeon 3.2GHZ and 4GB of RAM which has a very similar performance profile to the computer that we used, the numbers reported in our table and Table 3 in [64] are very similar and the comparison between them is meaningful. From these two tables, we can see that SDPAD is faster than SDPNAL for achieving a duality gap of almost the same order. The results of the boundary point method "mprw.m" reported in Table 4 in [64] are much worse than those of SDPAD. Since the implementation in "mprw.m" is essentially an alternating direction method applied to SDPs in standard form, we can con-

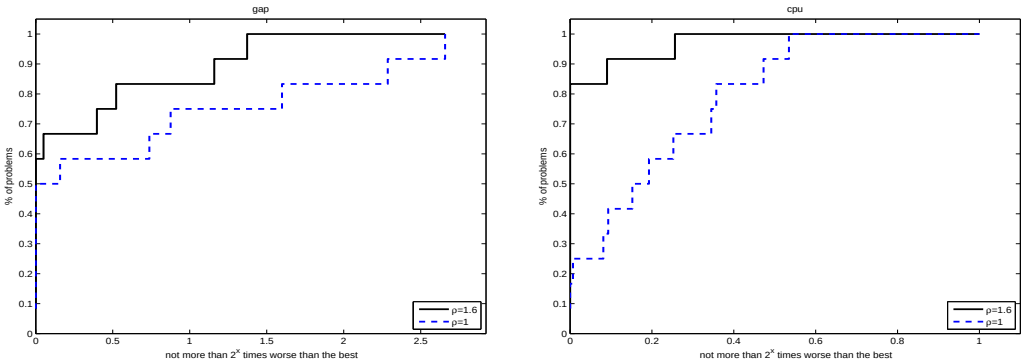
Table 3.1: Computational results on computing frequency assignment problems

			SDPAD							SDPNAL	
name	n	$\hat{m}$	pobj	dobj	pinf	dinf	itr	gap	cpu	gap	cpu
fap01	52	1378	3.28730e-2	3.28826e-2	2.78e-7	3.94e-7	607	9.06e-6	0.35	1.39e-7	6.31
fap02	61	1866	1.07696e-3	1.02391e-3	2.18e-6	1.94e-5	666	5.29e-5	0.49	1.15e-5	4.12
fap03	65	2145	4.94123e-2	4.94107e-2	5.89e-6	4.30e-6	840	1.48e-6	0.75	2.27e-6	7.44
fap04	81	3321	1.74875e-1	1.74845e-1	3.09e-6	2.21e-6	718	2.18e-5	1.21	1.53e-5	19.69
fap05	84	3570	3.08288e-1	3.08302e-1	3.49e-6	3.76e-6	768	8.84e-6	1.32	1.09e-5	31.59
fap06	93	4371	4.59362e-1	4.59379e-1	5.69e-6	5.72e-6	506	8.91e-6	1.07	1.73e-5	29.84
fap07	98	4851	2.11763e+	2.117686e+	4.54e-6	4.95e-6	543	9.89e-6	1.26	5.75e-6	29.88
fap08	120	7260	2.43583e+	2.43633e+	1.76e-5	8.74e-6	424	8.39e-5	1.57	5.93e-6	25.25
fap09	174	15225	1.07978e+1	1.07978e+1	7.99e-7	9.98e-7	505	2.06e-7	4.77	2.86e-6	59.67
fap10	183	14479	9.25963e-3	9.76489e-3	3.22e-6	5.46e-6	1250	4.96e-4	14.93	7.93e-5	1:50
fap11	252	24292	2.93137e-2	2.98421e-2	3.26e-6	3.21e-6	1654	4.99e-4	49.57	1.89e-4	5:15
fap12	369	26462	2.72774e-1	2.73768e-1	2.59e-6	3.19e-6	2000	6.43e-4	2:34	1.60e-4	13:14
fap25*	2118	322924	1.28632e+1	1.28802e+1	1.40e-5	1.63e-5	2000	6.37e-4	7:17:08	1.1e-4	10:53:22
fap36*	4110	1154467	6.98284e+1	6.98594e+1	2.03e-5	1.48e-5	2000	2.20e-4	53:14:12	2.5e-5	65:25:07

clude that treating inequality constraints directly can greatly improve the performance of such methods.

We now compare the numerical results of our alternating direction methods obtained using $\rho = 1$ and $\rho = 1.6$ by using performance profiles as proposed in [18]. Specifically, performance plots for the duality gap and the CPU time are presented in Figures 3.1(a) and (b), respectively. These figures show that the variant using $\rho = 1.6$ is both faster than the variant using $\rho = 1$ and achieves a smaller duality gap.

Figure 3.1: Performance profiles of two variants of SDPAD for frequency assignment problems



(a) gap (b) cpu

3.4.2 The SDP relaxation of the maximum stable set problem

Given a graph G with edge set E, two SDPs relaxations of the maximum stable set problem are

$$\begin{aligned} \theta(G) := \max \quad & \left\langle C^\top, X \right\rangle \\ \text{s.t.} \quad & X_{ij} = 0, \quad (i,j) \in E, \quad \langle I, X \rangle = 1, \\ & X \succeq 0, \end{aligned} \tag{3.4.2}$$

and

$$\begin{aligned} \theta_+(G) := \max \quad & \left\langle C^\top, X \right\rangle \\ \text{s.t.} \quad & X_{ij} = 0, \quad (i,j) \in E, \quad \langle I, X \rangle = 1, \\ & X \succeq 0, \quad X \geq 0, \end{aligned} \tag{3.4.3}$$

where $C = ee^\top$. We scaled the constraints so that $\mathcal{A}\mathcal{A}^* = I$, i.e., we replaced the constraints $X_{ij} = 0$ by $X_{ij}/\sqrt{2} = 0$ and $\langle I, X\rangle = 1$ by $\frac{1}{\sqrt{n}}\langle I, X\rangle = \frac{1}{\sqrt{n}}$. The matrix C was also scaled by n. The tests problems were taken from [35, 49, 53]. The numbers h_1, h_2 and h_3 were set to 20, 50 and 150, respectively, the iteration counter h_4 for changing μ was set to 100 and the ratios η_1 and η_2 were set to 1. We stopped SDPAD when the total number of iterations reached 1000. All other parameters were set to their default values. Summaries of the computational results for $\theta(G)$ and $\theta_+(G)$ are presented in Tables 3.2 and 3.3, respectively. In these tables, the duality "gap" was measured in the original scale, but "pinf" and "dinf" were computed for the scaled problems. Since running SDPNAL on all the test problems is very time consuming (for example, in the results reported in [64], SDPNAL took almost 81 hours to compute $\theta_+(G)$ on problem "1et.2048"), we did not run SDPNAL on our own computer on any of the $\theta(G)$ and $\theta_+(G)$ or BIQ (see next subsection) problems. Hence, the SDPNAL results in Tables 3.2 and 3.3 are taken from Tables 5 and 6 in [64]. Because the computer used to obtain the results in [64] has very similar performance characteristics to the computer that we used, the comparison presented in Tables 3.2 and 3.3 is meaningful. Specifically, when we run SDPNAL on smaller and easier problems, such as "fap01"-"fap12", on our computer, the cpu time differed from those reported in [64] by an insignificant amount. From Table 3.2, we can see that SDPAD achieves approximately the same level of duality gap as SDPNAL on problems such as "theta102" to "theta123", "c-fat200-1" and "brock400-1". Although SDPNAL is faster than SDPAD on problems such as "hamming-10-2" and "G43" to "G47", SDPAD is faster than SDPNAL on "2dc.512". From Table 3.3, we can see that SDPAD is faster than SDPNAL on most problems except

Table 3.2: Computational results on computing $\theta(G)$

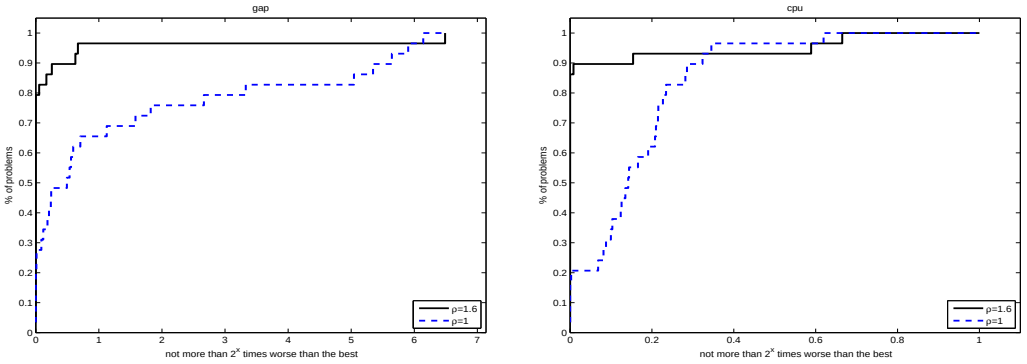
			SDPAD							SDPNAL	
name	n	$\hat{m}$	pobj	dobj	pinf	dinf	itr	gap	cpu	gap	cpu
theta102	500	37467	-3.8390626e+1	-3.8390551e+1	8.76e-7	6.27e-7	256	9.67e-7	47	1.6-8	50
theta103	500	62516	-2.2528588e+1	-2.2528572e+1	2.85e-7	9.42e-7	257	3.33e-7	57	4.6-8	1:00
theta104	500	87245	-1.3336159e+1	-1.3336141e+1	3.40e-7	9.72e-7	260	6.53e-7	48	7.6-8	58
theta123	600	90020	-2.4668678e+1	-2.4668655e+1	3.42e-7	9.71e-7	263	4.57e-7	1:43	4.1-8	1:34
MANN-a27	378	703	-1.3276623e+2	-1.3276388e+2	8.45e-7	3.50e-6	503	8.81e-6	27	8.3-8	07
sanr200-0.7	200	6033	-2.3836177e+1	-2.3836162e+1	7.83e-7	9.98e-7	219	3.04e-7	04	1.4-7	04
c-fat200-1	200	18367	-1.2000003e+1	-1.1999980e+1	1.00e-6	1.76e-7	302	9.15e-7	04	8.5-8	09
ham-10-2	1024	23041	-1.0239734e+2	-1.0239930e+2	4.72e-7	3.16e-6	597	9.51e-6	22:8	9.0-8	02
ham-8-3-4	256	16129	-2.5599950e+1	-2.5599909e+1	9.92e-8	9.94e-7	199	8.04e-7	05	1.3-8	10
ham-9-5-6	512	53761	-8.5332165e+1	-8.5334776e+1	5.70e-7	4.51e-6	1000	1.52e-5	2:48	1.4-6	1:33
brock400-1	400	20078	-3.9701971e+1	-3.9701904e+1	9.75e-7	8.06e-7	254	8.30e-7	25	1.7-8	26
keller4	171	5101	-1.4012231e+1	-1.4012258e+1	5.06e-7	9.88e-7	249	9.32e-7	03	1.3-8	05
p-hat300-1	300	33918	-1.0067984e+1	-1.0067963e+1	7.58e-7	6.81e-7	764	9.96e-7	37	5.3-7	1:45
G43	1000	9991	-2.8063120e+2	-2.8062688e+2	2.84e-6	3.91e-6	935	7.68e-6	21:17	4.2-8	1:33
G44	1000	9991	-2.8058951e+2	-2.8058568e+2	3.65e-6	4.21e-6	933	6.82e-6	21:02	3.3-7	2:59
G45	1000	9991	-2.8017918e+2	-2.8018294e+2	4.02e-6	3.88e-6	957	6.70e-6	21:27	5.6-8	2:51
G46	1000	9991	-2.7984557e+2	-2.7984014e+2	3.18e-6	5.46e-6	927	9.69e-6	21:02	2.3-7	2:53
G47	1000	9991	-2.8190252e+2	-2.8189748e+2	4.86e-6	6.01e-6	880	8.91e-6	19:41	1.3-7	2:54
2dc.512	512	54896	-1.1777815e+1	-1.1770773e+1	2.24e-5	2.57e-5	1000	2.87e-4	5:51	1.7-4	32:16
1dc.1024	1024	24064	-9.6053190e+1	-9.5999280e+1	7.82e-5	5.10e-5	747	2.79e-4	24:26	2.9-6	41:26
1et.1024	1024	9601	-1.8460646e+2	-1.8434339e+2	1.93e-4	1.32e-4	603	7.11e-4	20:03	1.8-6	1:01:14
1tc.1024	1024	7937	-2.0705051e+2	-2.0663863e+2	4.16e-4	5.21e-4	611	9.93e-4	21:47	2.2-6	1:48:04
1zc.1024	1024	16641	-1.2866647e+2	-1.2866658e+2	9.72e-7	3.78e-7	608	4.16e-7	23:15	3.3-8	4:15
2dc.1024	1024	169163	-1.8654205e+1	-1.8641209e+1	1.85e-5	2.71e-5	1000	3.39e-4	49:05	9.9-5	2:57:56
1dc.2048	2048	58368	-1.7527338e+2	-1.7492237e+2	2.82e-4	2.16e-4	473	9.99e-4	2:50:44	1.5-6	6:11:11
1et.2048	2048	22529	-3.4297575e+2	-3.4229432e+2	1.78e-4	4.05e-4	904	9.93e-4	4:54:57	8.8-7	7:13:55
1tc.2048	2048	18945	-3.7567281e+2	-3.7486271e+2	1.79e-4	4.48e-4	1000	1.08e-3	5:15:14	7.9-6	9:52:09
1zc.2048	2048	39425	-2.3739409e+2	-2.3739845e+2	1.42e-6	3.10e-6	941	9.17e-6	6:39:07	1.2-6	45:16
2dc.2048	2048	504452	-3.0698999e+1	-3.0679246e+1	1.88e-5	8.37e-6	1000	3.17e-4	7:12:50	4.4-5	15:13:19

"hamming-9-5-6", "hamming-10-2", "1zc.1024" and "1zc.2048" while achieving almost the same level of duality gap. Finally, performance plots for numerical results obtained using $\rho = 1$ and $\rho = 1.6$ for computing $\theta(G)$ and $\theta_+(G)$ are presented in Figures 3.2(a) and (b), and Figures 3.3(a) and (b), respectively. When both the final duality gap and CPU time are considered, these plots again show that using a fixed step size of $\rho = 1.6$ is preferable to a step size of $\rho = 1$.

Table 3.3: Computational results on computing $\theta_+(G)$

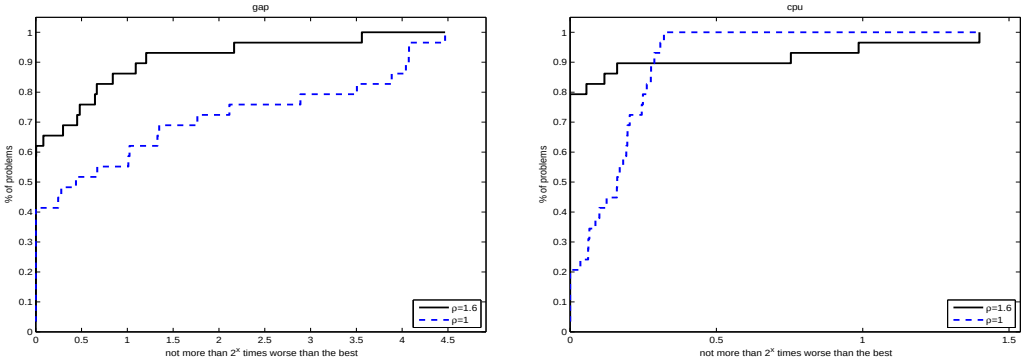
			SDPAD							SDPNAL	
name	n	$\hat{m}$	pobj	dobj	pinf	dinf	itr	gap	cpu	gap	cpu
theta102	500	37467	-3.8066274e+1	-3.8066252e+1	2.94e-7	9.58e-7	281	2.84e-7	1:01	8.4-8	3:31
theta103	500	62516	-2.2377445e+1	-2.2377422e+1	3.27e-7	9.52e-7	262	4.85e-7	1:01	2.3-8	3:28
theta104	500	87245	-1.3282631e+1	-1.3282610e+1	3.57e-7	9.55e-7	266	7.35e-7	52	1.6-7	2:35
theta123	600	90020	-2.4495182e+1	-2.4495152e+1	3.77e-7	9.43e-7	267	6.02e-7	1:39	1.2-7	6:44
MANN-a27	378	703	-1.3275956e+2	-1.3276174e+2	8.98e-7	4.08e-6	530	8.17e-6	32	1.6-7	35
sanr200-0.7	200	6033	-2.3633314e+1	-2.3633293e+1	8.61e-7	9.57e-7	228	4.47e-7	05	2.9-7	11
c-fat200-1	200	18367	-1.2000012e+1	-1.1999991e+1	4.71e-7	2.04e-7	306	8.40e-7	04	2.1-7	36
ham-8-3-4	256	16129	-2.5599951e+1	-2.5599909e+1	9.91e-8	9.94e-7	199	8.05e-7	05	2.7-10	05
ham-9-5-6	512	53761	-5.8666560e+1	-5.8666522e+1	6.49e-8	7.58e-7	472	3.25e-7	2:19	2.6-7	42
ham-10-2	1024	23041	-8.5333069e+1	-8.5333237e+1	4.99e-8	5.49e-7	653	9.78e-7	27:58	4.2-7	4:35
brock400-1	400	20078	-3.9331005e+1	-3.9330926e+1	9.98e-7	7.54e-7	258	9.83e-7	29	3.5-9	1:45
keller4	171	5101	-1.3466089e+1	-1.3466006e+1	3.61e-6	9.01e-6	331	2.98e-6	05	3.7-7	43
p-hat300-1	300	33918	-1.0020244e+1	-1.0020212e+1	1.34e-6	5.93e-7	567	1.50e-6	29	7.9-7	6:50
G43	1000	9991	-2.7972840e+2	-2.7973289e+2	4.40e-6	5.45e-6	864	8.02e-6	20:22	2.1-7	52:00
G44	1000	9991	-2.7975221e+2	-2.7974864e+2	4.95e-6	4.36e-6	893	6.36e-6	21:14	5.7-8	49:32
G45	1000	9991	-2.7931027e+2	-2.7931528e+2	5.11e-6	4.08e-6	916	8.96e-6	22:54	2.4-8	50:25
G46	1000	9991	-2.7904079e+2	-2.7903549e+2	4.34e-6	5.19e-6	886	9.47e-6	22:48	3.3-8	44:38
G47	1000	9991	-2.8089994e+2	-2.8089501e+2	6.05e-6	5.66e-6	838	8.76e-6	21:11	5.1-9	40:27
2dc.512	512	54896	-1.1385823e+1	-1.1383769e+1	9.43e-6	9.09e-6	1000	8.64e-5	5:10	3.8-4	2:25:15
1dc.1024	1024	24064	-9.5563960e+1	-9.5552471e+1	2.13e-5	8.50e-6	1000	5.98e-5	57:20	1.4-5	5:03:49
1et.1024	1024	9601	-1.8229378e+2	-1.8210159e+2	1.18e-4	2.16e-4	651	5.26e-4	36:04	1.1-5	6:45:50
1tc.1024	1024	7937	-2.0440122e+2	-2.0425679e+2	1.65e-4	2.67e-4	799	3.53e-4	49:13	8.7-4	10:37:57
1zc.1024	1024	16641	-1.2813904e+2	-1.2800286e+2	5.71e-5	1.80e-5	506	5.30e-4	28:22	1.6-7	40:13
2dc.1024	1024	169163	-1.7711337e+1	-1.7709936e+1	3.81e-6	2.37e-6	1000	3.85e-5	50:34	7.3-4	11:57:25
1dc.2048	2048	58368	-1.7477173e+2	-1.7442175e+2	2.01e-4	9.45e-5	517	9.99e-4	4:25:17	9.7-5	35:52:44
1et.2048	2048	22529	-3.3883524e+2	-3.3837436e+2	1.76e-4	2.66e-4	659	6.80e-4	5:15:09	4.0-5	80:48:17
1tc.2048	2048	18945	-3.7115486e+2	-3.7070213e+2	1.86e-4	3.97e-4	862	6.09e-4	6:35:33	1.4-3	73:56:01
1zc.2048	2048	39425	-2.3739370e+2	-2.3739764e+2	3.58e-7	4.60e-6	953	8.27e-6	7:14:21	2.3-7	2:13:04
2dc.2048	2048	504452	-2.8789604e+1	-2.8786706e+1	4.84e-6	2.79e-6	1000	4.95e-5	5:45:25	2.7-3	45:21:42

Figure 3.2: Performance profiles of two variants of SDPAD for computing $\theta(G)$



(a) gap (b) cpu

Figure 3.3: Performance profiles of two variants of SDPAD for computing $\theta_+(G)$



(a) gap (b) cpu

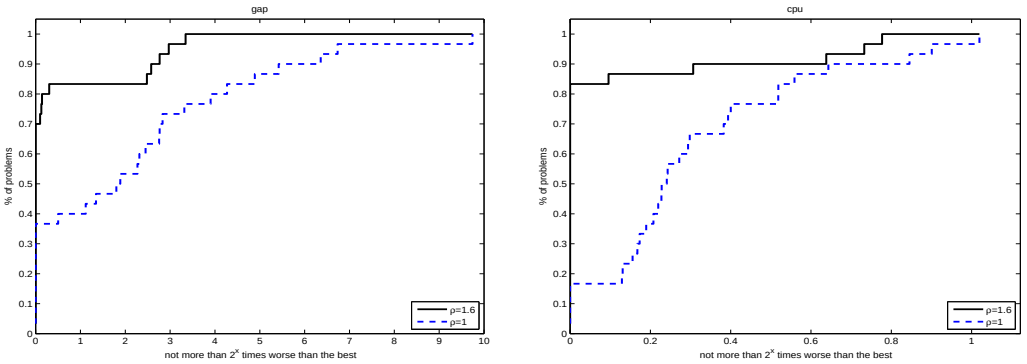
3.4.3 Binary Integer Quadratic Programming Problem

In this subsection, we report on how SDPAD performs on SDP relaxations of binary integer quadratic programming problems and compare these results to those obtained using SDPNAL. These problems have the form:

$$\begin{aligned} \min \quad & \left\langle \begin{pmatrix} Q & 0 \\ 0 & 0 \end{pmatrix}, X \right\rangle \\ s.t. \quad & X_{ii} - X_{n,i} = 0, i = 1, \cdots, n-1, \\ & X_{nn} = 1, \quad X \succeq 0, \quad X \geq 0, \end{aligned} \tag{3.4.4}$$

where $Q \in \mathbb{R}^{(n-1)\times(n-1)}$. We replaced the constraints $X_{ii} - X_{n,i} = 0$ by $\sqrt{\frac{2}{3}}\left(X_{ij} - X_{n,i}\right) = 0$ and the matrix Q was scaled by its Frobenious norm. The computational results obtained on the BIQ instances described in [58] are presented in Table 3.4, where "best upper bound" is the best known upper bound reported in [58], and "%pgap" and "%dgap" were computed

Figure 3.4: Performance profiles of two variants of SDPAD for the BIQ problems



(a) gap (b) cpu

as

$$\begin{aligned} \%\text{pgap} &:= \left| \frac{\text{best upper bound} - \text{pobj}}{\text{best upper bound}} \right| \times 100\%, \\ \%\text{dgap} &:= \left| \frac{\text{best upper bound} - \text{dobj}}{\text{best upper bound}} \right| \times 100\%. \end{aligned}$$

The numbers h_1, h_2 and h_3 were set to 50, 400 and 500, respectively, the iteration counter h_4 for changing μ was set to 0 and the ratios η_1 and η_2 were set to 1 and 100, respectively. We stopped SDPAD when the total number of iterations reached 10000. The minimum penalty parameter $\mu_{\min}$ is set to 0.1. All other parameters were set to their default values. Again, we did not run SDPNAL on our own computer but presented the results reported in Table 8 in [64] in Table 3.4. From this table, we can see that SDPAD is faster than SDPNAL for achieving comparable lower bounds. Finally, performance plots for numerical results obtained using $\rho = 1$ and $\rho = 1.6$ are presented in Figures 3.4(a) and (b). When both the final duality gap and CPU time are considered, these plots again show that using a fixed step size of $\rho = 1.6$ is preferable to using a step size of $\rho = 1$.

Table 3.4: Computational results on the BIQ problems

		SDPAD								SDPNAL	
name	n	pinf	dinf	gap	itr	best upper bound	%pgap	%dgap	cpu	%dgap	cpu
be200.3.1	201	7.09e-7	5.17e-7	9.99e-7	2484	-2.5453000e+4	8.891	8.891	32	8.891	10:29
be200.3.3	201	8.73e-6	1.42e-6	6.43e-6	2296	-2.8023000e+4	5.194	5.195	31	5.192	12:09
be200.3.5	201	2.66e-6	3.45e-8	3.55e-8	4782	-2.6355000e+4	6.519	6.519	1:04	6.519	10:38
be200.3.7	201	8.88e-6	1.78e-6	9.97e-6	2447	-3.0483000e+4	3.730	3.732	32	3.730	9:43
be200.3.9	201	1.00e-6	1.75e-8	1.50e-8	6940	-2.4683000e+4	7.106	7.106	1:31	7.106	8:28
be200.8.1	201	1.93e-6	3.06e-8	2.77e-8	4267	-4.8534000e+4	4.812	4.812	57	4.811	9:41
be200.8.3	201	8.79e-6	1.03e-6	1.00e-5	2107	-4.3207000e+4	7.051	7.053	28	7.052	10:53
be200.8.5	201	8.60e-6	1.48e-6	9.99e-6	1885	-4.1482000e+4	6.725	6.723	25	6.723	9:53
be200.8.7	201	8.59e-6	4.73e-7	9.97e-6	2186	-4.6828000e+4	5.394	5.391	28	5.392	4:30
be200.8.9	201	5.57e-6	7.95e-7	4.60e-6	2146	-4.3241000e+4	5.213	5.214	29	5.213	12:16
be250.1	251	9.99e-7	3.10e-8	8.74e-9	5923	-2.4076000e+4	4.334	4.334	2:14	4.332	16:41
be250.3	251	6.12e-6	4.58e-6	7.82e-6	2747	-2.2923000e+4	4.698	4.697	1:02	4.698	17:17
be250.5	251	9.98e-7	2.21e-8	1.03e-8	6326	-2.1057000e+4	6.258	6.258	2:24	6.254	14:30
be250.7	251	2.30e-6	5.18e-8	7.88e-9	5256	-2.4095000e+4	4.250	4.250	1:57	4.250	14:00
be250.9	251	1.00e-6	5.61e-8	8.25e-9	6532	-2.0051000e+4	6.713	6.713	2:30	6.713	17:13
bqp250-1	251	6.41e-6	2.57e-6	9.97e-6	3005	-4.5607000e+4	4.509	4.507	1:08	4.508	17:42
bqp250-3	251	3.18e-6	2.06e-6	7.68e-7	3006	-4.9037000e+4	4.159	4.159	1:05	4.160	10:36
bqp250-5	251	7.80e-6	1.10e-6	9.08e-6	3129	-4.7961000e+4	4.260	4.261	1:10	4.260	19:03
bqp250-7	251	5.63e-7	3.21e-7	9.98e-7	4801	-4.6757000e+4	4.630	4.630	1:46	4.630	16:36
bqp250-9	251	6.64e-6	3.47e-6	9.24e-6	3006	-4.8916000e+4	5.277	5.279	1:06	5.276	16:12
bqp500-1	501	2.36e-7	3.92e-7	1.00e-6	8960	-1.1658600e+5	8.044	8.044	19:44	8.045	1:00:59
bqp500-3	501	2.42e-7	3.61e-7	1.00e-6	8824	-1.3081200e+5	5.842	5.841	19:04	5.842	1:01:47
bqp500-5	501	4.44e-7	4.03e-7	9.99e-7	8288	-1.2548700e+5	6.857	6.857	18:41	6.857	1:36:43
bqp500-7	501	2.75e-7	4.83e-7	9.99e-7	9153	-1.2220100e+5	7.603	7.602	20:16	7.603	1:25:26
bqp500-9	501	2.82e-7	4.88e-7	1.00e-6	8439	-1.2079800e+5	7.856	7.856	18:54	7.857	1:24:40
gka2e	201	9.99e-7	1.65e-8	2.07e-8	4375	-2.3395000e+4	6.508	6.508	57	6.506	7:23
gka4e	201	8.35e-6	2.75e-7	3.42e-6	2735	-3.5594000e+4	4.583	4.582	36	4.582	11:25
gka1f	501	4.43e-7	5.83e-7	9.99e-7	8106	-6.1194000e+4	7.133	7.133	17:57	7.133	1:28:54
gka3f	501	3.43e-7	5.27e-7	9.99e-7	7785	-1.3803500e+5	8.778	8.777	17:04	8.778	1:31:34
gka5f	501	3.78e-7	7.40e-7	1.00e-6	8849	-1.9050700e+5	8.612	8.612	18:58	8.613	1:25:48

3.5 Conclusion

In this chapter, we presented alternating direction augmented Lagrangian methods for solving semidefinite programming (SDP) problems. At each inner iteration, the algorithm minimizes the dual augmented Lagrangian function with respect to each block of dual variables separately while other blocks are fixed and then updates the primal variables. For the version of our algorithm that uses a unit step size $\rho = 1$, complementary is enforced by computing partial eigenvalue decompositions at each iteration. The special structure of the constraints, such as sparsity and orthogonality, can often be used to simplify the computation when solving the subproblems. Our method can handle SDPs with inequality and positivity constraints directly without transforming them to equality constraints. Since in our numerical experience the low rank structure of the optimal solution is often exposed after relatively few iterations, we plan to explore how to take advantage of this to improve the efficiency of our algorithm.

Bibliography

[1] F. Alizadeh and D. Goldfarb, *Second-order cone programming*, Math. Programming **95** (2003), no. 1, Ser. B, 3–51, ISMP 2000, Part 3 (Atlanta, GA). MR MR1971381 (2004j:90060)

[2] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont, *Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data*, J. Mach. Learn. Res. **9** (2008), 485–516. MR MR2417243

[3] Steven J. Benson, Yinyu Ye, and Xiong Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization*, SIAM J. Optim. **10** (2000), no. 2, 443–461. MR MR1740955 (2000k:90074)

[4] Dimitri P. Bertsekas, *Necessary and sufficient condition for a penalty method to be exact*, Math. Programming **9** (1975), no. 1, 87–99. MR MR0384144 (52 #5021)

[5] ———, *Constrained optimization and Lagrange multiplier methods*, Computer Science and Applied Mathematics, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1982. MR MR690767 (84k:90068)

[6] ———, *Nonlinear programming*, Athena Scientific, September 1999.

[7] Dimitri P. Bertsekas and John N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[8] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge University Press, Cambridge, 2004. MR MR2061575 (2005d:90002)

[9] Samuel Burer, *Optimizing a polyhedral-semidefinite relaxation of completely positive programs*, Manuscript, Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1994, USA, December 2008, Submitted to *Mathematical Programming Computation*.

[10] Samuel Burer and Renato D. C. Monteiro, *A projected gradient algorithm for solving the maxcut SDP relaxation*, Optim. Methods Softw. **15** (2001), no. 3-4, 175–200. MR MR1892584 (2003b:90138)

[11] ———, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Program. **95** (2003), no. 2, Ser. B, 329–357. MR MR1976484 (2004b:90080)

[12] ———, *Local minima and convergence in low-rank semidefinite programming*, Math. Program. **103** (2005), no. 3, Ser. A, 427–444. MR MR2166543 (2006j:90058)

[13] Samuel Burer, Renato D. C. Monteiro, and Yin Zhang, *A computational study of a gradient-based log-barrier algorithm for a class of large-scale SDPs*, Math. Program. **95** (2003), no. 2, Ser. B, 359–379, Computational semidefinite and second order cone programming: the state of the art. MR MR1976485 (2004d:90056)

[14] Samuel Burer and Dieter Vandenbussche, *Solving lift-and-project relaxations of binary integer programs*, SIAM J. Optim. **16** (2006), no. 3, 726–750. MR MR2197554 (2006i:90033)

[15] Emmanuel J. Candes and Benjamin Recht, *Exact matrix completion via convex optimization*, Tech. report, California Institute of Technology, May 2008.

[16] Gong Chen and Marc Teboulle, *A proximal-based decomposition method for convex minimization problems*, Math. Programming **64** (1994), no. 1, Ser. A, 81–101. MR MR1274173 (95e:90069)

[17] Etienne de Klerk, *Aspects of semidefinite programming*, Applied Optimization, vol. 65, Kluwer Academic Publishers, Dordrecht, 2002, Interior point algorithms and selected applications. MR MR2064921 (2005a:90001)

[18] Elizabeth D. Dolan and Jorge J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program. **91** (2002), no. 2, Ser. A, 201–213. MR MR1875515 (2002j:90001)

[19] Jonathan Eckstein and Dimitri P. Bertsekas, *An alternating direction method for linear programming.*, LIDS-P, 1967. Cambridge, MA, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology.

[20] Jonathan Eckstein and Dimitri P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Programming **55** (1992), no. 3, Ser. A, 293–318. MR MR1168183 (93c:90057)

[21] R. Fletcher, *Practical methods of optimization*, second ed., A Wiley-Interscience Publication, John Wiley & Sons Ltd., Chichester, 1987. MR MR955799 (89j:65050)

[22] Michel Fortin and Roland Glowinski, *Augmented Lagrangian methods*, Studies in Mathematics and its Applications, vol. 15, North-Holland Publishing Co., Amsterdam, 1983, Applications to the numerical solution of boundary value problems, Translated from the French by B. Hunt and D. C. Spicer. MR MR724072 (85a:49004)

[23] Katsuki Fujisawa, Masakazu Kojima, and Kazuhide Nakata, *Exploiting sparsity in primal–dual interior-point methods for semidefinite programming*, Math. Programming **79** (1997), no. 1-3, Ser. B, 235–253, Lectures on mathematical programming (ISMP97) (Lausanne, 1997). MR MR1464769 (98h:90052)

[24] Roland Glowinski and Patrick Le Tallec, *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*, SIAM Studies in Applied Mathematics, vol. 9, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1989. MR MR1060954 (91f:73038)

[25] Michel X. Goemans and David P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. Assoc. Comput. Mach. **42** (1995), no. 6, 1115–1145. MR MR1412228 (97g:90108)

[26] D. Goldfarb and G. Iyengar, *Robust convex quadratically constrained programs*, Math. Program. **97** (2003), no. 3, Ser. B, 495–515, New trends in optimization and computational algorithms (NTOC 2001) (Kyoto). MR MR2008120 (2004g:90072)

[27] ———, *Robust portfolio selection problems*, Math. Oper. Res. **28** (2003), no. 1, 1–38. MR MR1961265 (2004f:90098)

[28] L. Grippo and M. Sciandrone, *On the convergence of the block nonlinear Gauss-Seidel method under convex constraints*, Oper. Res. Lett. **26** (2000), no. 3, 127–136. MR MR1746833 (2001a:90061)

[29] Luigi Grippo and Marco Sciandrone, *Globally convergent block-coordinate techniques for unconstrained optimization*, Optim. Methods Softw. **10** (1999), no. 4, 587–637. MR MR1694581 (2000a:90064)

[30] Elaine T. Hale, Wotao Yin, and Yin Zhang, *Fixed-point continuation for l_1-minimization: methodology and convergence*, SIAM J. Optim. **19** (2008), no. 3, 1107–1130. MR MR2460734

[31] B. S. He, H. Yang, and S. L. Wang, *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities*, J. Optim. Theory Appl. **106** (2000), no. 2, 337–356. MR MR1788928 (2001h:49016)

[32] Bingsheng He, Li-Zhi Liao, Deren Han, and Hai Yang, *A new inexact alternating directions method for monotone variational inequalities*, Math. Program. **92** (2002), no. 1, Ser. A, 103–118. MR MR1892298 (2003b:90111)

[33] C. Helmberg and F. Rendl, *A spectral bundle method for semidefinite programming*, SIAM J. Optim. **10** (2000), no. 3, 673–696. MR MR1741192 (2002b:90095)

[34] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal, *Convex analysis and minimization algorithms. I*, Grundlehren der Mathematischen Wissenschaften [Fundamental

Principles of Mathematical Sciences], vol. 305, Springer-Verlag, Berlin, 1993, Fundamentals. MR MR1261420 (95m:90001)

[35] David S. Johnson and Michael A. Trick (eds.), *Cliques, coloring, and satisfiability*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 26, American Mathematical Society, Providence, RI, 1996, Papers from the workshop held as part of the 2nd DIMACS Implementation Challenge in New Brunswick, NJ, October 11–13, 1993. MR MR1423138 (97g:68003)

[36] Krzysztof C. Kiwiel, Charles H. Rosa, and Andrzej Ruszczyński, *Proximal decomposition via alternating linearization*, SIAM J. Optim. **9** (1999), no. 3, 668–689. MR MR1681047 (2000d:90071)

[37] Michal Kočvara and Michael Stingl, *Pennon: a code for convex nonlinear and semidefinite programming*, Optim. Methods Softw. **18** (2003), no. 3, 317–333. MR MR1989707 (2004e:90076)

[38] Spyridon Kontogiorgis and Robert R. Meyer, *A variable-penalty alternating directions method for convex optimization*, Math. Programming **83** (1998), no. 1, Ser. A, 29–53. MR MR1643963 (99k:90116)

[39] László Lovász, *On the Shannon capacity of a graph*, IEEE Trans. Inform. Theory **25** (1979), no. 1, 1–7. MR MR514926 (81g:05095)

[40] Z. Q. Luo and P. Tseng, *On the convergence of the coordinate descent method for convex differentiable minimization*, J. Optim. Theory Appl. **72** (1992), no. 1, 7–35. MR MR1141764 (92k:90092)

[41] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and Bregman iterative methods for matrix rank minimization*, Tech. report, Department of IEOR, Columbia University, 2008.

[42] Jerome Malick, Janez Povh, Franz Rendl, and Angelika Wiegele, *Regularization methods for semidefinite programming*, SIAM Journal on Optimization **20** (2009), no. 1, 336–356.

[43] Jorge Nocedal and Stephen J. Wright, *Numerical optimization*, second ed., Springer Series in Operations Research and Financial Engineering, Springer, New York, 2006. MR MR2244940 (2007a:90001)

[44] Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin, *An iterative regularization method for total variation-based image restoration*, Multiscale Model. Simul. **4** (2005), no. 2, 460–489. MR MR2162864 (2006c:49051)

[45] Gàbor Pataki and Stefan Schmieta, *The dimacs library of semidefinite-quadratic-linear programs*, Tech. report, Center, Columbia University, 1999.

[46] J. Povh, F. Rendl, and A. Wiegele, *A boundary point method to solve semidefinite programs*, Computing **78** (2006), no. 3, 277–286. MR MR2266705 (2007i:90056)

[47] M. J. D. Powell, *On search directions for minimization algorithms*, Math. Programming **4** (1973), 193–201. MR MR0321541 (47 #10074)

[48] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res. **1** (1976), no. 2, 97–116. MR MR0418919 (54 #6954)

[49] N. J. A. Sloane, *Challenge problems: Independent sets in graphs*, http://research.att.com/ njas/doc/graphs.html.

[50] Wenyu Sun and Ya-Xiang Yuan, *Optimization theory and methods*, Springer Optimization and Its Applications, vol. 1, Springer, New York, 2006, Nonlinear programming. MR MR2232297 (2007c:90002)

[51] Xue-Cheng Tai and Jinchao Xu, *Global and uniform convergence of subspace correction methods for some convex optimization problems*, Math. Comp. **71** (2002), no. 237, 105–124. MR MR1862990 (2002h:65097)

[52] M. J. Todd, *Semidefinite optimization*, Acta Numer. **10** (2001), 515–560. MR MR2009698 (2004g:90004)

[53] Kim-Chuan Toh, *Solving large scale semidefinite programs via an iterative solver on the augmented systems*, SIAM J. Optim. **14** (2003), no. 3, 670–698. MR MR2085936 (2005c:90047)

[54] Paul Tseng, *Alternating projection-proximal methods for convex programming and variational inequalities*, SIAM J. Optim. **7** (1997), no. 4, 951–965. MR MR1479608 (99a:90156)

[55] Paul Tseng and Sangwoon Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Math. Program. **117** (2009), no. 1-2, Ser. B, 387–423. MR MR2421312

[56] Lieven Vandenberghe and Stephen Boyd, *Semidefinite programming*, SIAM Rev. **38** (1996), no. 1, 49–95. MR MR1379041 (96m:90005)

[57] Yilun Wang, Junfeng Yang, Wotao Yin, and Yin Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM J. Imaging Sci. **1** (2008), no. 3, 248–272. MR MR2486032

[58] Angelika Wiegele, *Biq mac library - a collection of max-cut and quadratic 0-1 programming instances of medium size*, Tech. report, 2007.

[59] Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe (eds.), *Handbook of semidefinite programming*, International Series in Operations Research & Management Science, 27, Kluwer Academic Publishers, Boston, MA, 2000, Theory, algorithms, and applications. MR MR1778223 (2001k:90001)

[60] Junfeng Yang, Yin Zhang, and Wotao Yin, *An efficient tvl1 algorithm for deblurring multichannel images corrupted by impulsive noise*, Tech. report, Rice University, 2008.

[61] Zhensheng Yu, *Solving semidefinite programming problems via alternating direction methods*, J. Comput. Appl. Math. **193** (2006), no. 2, 437–445. MR MR2229553 (2008a:90034)

[62] Fuzhen Zhang (ed.), *The Schur complement and its applications*, Numerical Methods and Algorithms, vol. 4, Springer-Verlag, New York, 2005. MR MR2160825 (2006e:15001)

[63] Yin Zhang, *User's guide for yall1: Your algorithms for l1 optimization*, Tech. report, Rice University, 2009.

[64] Xinyuan Zhao, Defeng Sun, and Kimchuan Toh, *A newton-cg augmented lagrangian method for semidefinite programming*, Tech. report, Department of Mathematics, National University of Singapore, 2008.

Appendix A

A1. Analysis of the augmented Lagrangian approach

We now present a convergence proof of our augmented Lagrangian approach (1.2.5)-(1.2.6). Let $\{X^k\}$ be the sequence generated by the iterative procedure (1.2.5)-(1.2.6). The first-order optimality conditions of (1.2.5) are

$$\mathcal{B}^{p^k}(X,X^k) := \left\langle \nabla \mathcal{L}(X^k,\pi^k,\mu^k), X - X^k \right\rangle = \left\langle C - p^k, X - X^k \right\rangle \geq 0, \qquad \text{(A.1)}$$

where $p^k = \frac{1}{\mu^k}\mathcal{A}^*(\mu^k\pi^k + b - \mathcal{A}(X^k))$, for all $X \in S^n_+$. Similar to Proposition 3.2 and Theorem 3.3 in [44], we have the following results on the convergence of the sequence $\{H(X^k)\}$, where $H(X) := \|\mathcal{A}(X) - b\|_2^2$.

Theorem A.0.1. *The iterates* $\{X^k\}$ *generated by the iterative procedure* (1.2.5)-(1.2.6) *satisfies the following:*

1) *Monotonic decreasing in H:* $H(X^{k+1}) \leq H(X^{k+1}) + 2\mu^{k+1}\mathcal{B}^{p^k}(X^{k+1}, X^k) \leq H(X^k)$ *for* $k \geq 1$.

2) $\mathcal{B}^{p^k}(X, X^k) + \mathcal{B}^{p^{k-1}}(X^k, X^{k-1}) + \frac{1}{\mu^k}H(X^k) \leq \frac{1}{\mu^k}H(X) + \mathcal{B}^{p^{k-1}}(X, X^{k-1})$ *for* $k \geq 2$.

3) *If* $\widetilde{X}$ *minimize* $H(X)$ *and* $\left\langle C, \widetilde{X} \right\rangle < \infty$, *then* $H(X^k) \leq \frac{\mu^1}{\mu^k}H(\widetilde{X}) + \frac{\mu^1}{k-1}\mathcal{B}^{p^1}(\widetilde{X}, X^1)$ *for* $k \geq 2$.

Proof. 1) The first part of statement one is obvious since $\mathcal{B}^{p^k}(X^{k+1}, X^k) \geq 0$. We now prove the second part of statement one. Using the updating formula (1.2.6), we obtain

$$\begin{aligned}
& \frac{1}{2\mu^{k+1}}H(X^{k+1}) + \mathcal{B}^{p^k}(X^{k+1}, X^k) \\
= \; & \frac{1}{2\mu^{k+1}}\|\mathcal{A}(X^{k+1}) - b\|_2^2 + \left\langle C, X^{k+1} - X^k \right\rangle \\
& + \left\langle \frac{1}{\mu^k}\mathcal{A}^*(\mathcal{A}(X^k) - b - \mu^k\pi^k), X^{k+1} - X^k \right\rangle \\
\leq \; & \frac{1}{2\mu^{k+1}}\|\mathcal{A}(X^k) - b\|_2^2 + \left\langle C, X^k \right\rangle - (\pi^{k+1})^\top(\mathcal{A}(X^k) - b) \\
& - \left\langle C, X^k \right\rangle + (\pi^{k+1})^\top(\mathcal{A}(X^{k+1}) - b) + \left\langle \frac{1}{\mu^k}\mathcal{A}^*(\mathcal{A}(X^k) - b - \mu^k\pi^k), X^{k+1} - X^k \right\rangle \\
= \; & \frac{1}{2\mu^{k+1}}\|\mathcal{A}(X^k) - b\|_2^2 = \frac{1}{2\mu^{k+1}}H(X^k),
\end{aligned}$$

where the inequality uses the fact that X^{k+1} is the optimal solution at iteration $k+1$.

2) From the definition of $\mathcal{B}^{p_k}(\cdot,\cdot)$, we have:

$$\mathcal{B}^{p^k}(X, X^k) + \mathcal{B}^{p^{k-1}}(X^k, X^{k-1}) - \mathcal{B}^{p^{k-1}}(X, X^{k-1}) = \left\langle p^{k-1} - p^k, X - X^k \right\rangle. \qquad \text{(A.2)}$$

We obtain from the definition of p^k that

$$\begin{aligned} p^{k-1}-p^k &= \frac{1}{\mu^{k-1}}\mathcal{A}^*\left(\mu^{k-1}\pi^{k-1}+b-\mathcal{A}(X^{k-1})\right)-\frac{1}{\mu^k}\mathcal{A}^*\left(\mu^k\pi^k+b-\mathcal{A}(X^k)\right) \\ &= \frac{1}{\mu^k}\left(\mathcal{A}^*\left(\mathcal{A}(X^k)-b\right)\right). \end{aligned} \tag{A.3}$$

It follows from the convexity of $H(X)$, (A.2) and (A.3) that

$$\begin{aligned} \frac{1}{\mu^k}H(X) &\geq \frac{1}{\mu^k}H(X^k)+\left\langle\frac{1}{\mu^k}\mathcal{A}^*\left(\mathcal{A}(X^k)-b\right),X-X^k\right\rangle \\ &= \frac{1}{\mu^k}H(X^k)+\left\langle p^k-p^{k+1},X-X^k\right\rangle, \end{aligned}$$

which gives the statement 2.

3) We take $X:=\widetilde{X}$ and summarize the inequality in statement 2 from $k\geq 2$ arriving at

$$\mathcal{B}^{p^k}(\widetilde{X},X^k)+\sum_{j=2}^{k}\left(\mathcal{B}^{p^{j-1}}(X^j,X^{j-1})+\frac{1}{\mu^j}H(X^j)-\frac{1}{\mu^j}H(\widetilde{X})\right)\leq\mathcal{B}^{p^1}(\widetilde{X},X^1). \tag{A.4}$$

Since $\mathcal{B}^{p^k}(\widetilde{X},X^k)\geq 0$, $\mathcal{B}^{p^{j-1}}(X^j,X^{j-1})\geq 0$ for $j=2,\cdots,k$, the monotonicity of $H(X^j)$ and $\mu^j\geq\mu^{j+1}$, we further conclude that $(k-1)\left(\frac{1}{\mu^1}H(X^k)-\frac{1}{\mu^k}H(\widetilde{X})\right)\leq\mathcal{B}^{p^1}(\widetilde{X},X^1)$, which completes the proof. □

We now present the proof of Theorem 1.2.6.

Proof. The optimality condition (A.1) at X^k gives

$$\begin{aligned}\left\langle C, X^k \right\rangle &\leq \langle C, X\rangle + \left\langle \frac{1}{\mu^k}\mathcal{A}^*(\mathcal{A}(X^k) - b - \mu^k\pi^k), X - X^k \right\rangle \\ &= \langle C, X\rangle + \frac{1}{\mu^k}\left\langle \mathcal{A}(X^k) - b - \mu^k\pi^k, \mathcal{A}(X) - \mathcal{A}(X^k)\right\rangle. \end{aligned} \tag{A.5}$$

Since the Slater condition holds, $H(\widetilde{X}) = 0$ is attainable. Hence, statement 3 of Theorem A.0.1 implies that $\lim_{k\to\infty} A(X^k) = b$. Let $\widehat{X}$ be a limit point of the sequence $\{X^k\}$. Taking limits on both sides of (A.5), we obtain

$$\left\langle C, \widehat{X}\right\rangle \leq \langle C, X\rangle + \left\langle \pi^k, \mathcal{A}(X) - b\right\rangle,$$

which gives $\left\langle C, \widehat{X}\right\rangle \leq \langle C, X\rangle$ for any X satisfying $\mathcal{A}(X) = b$; Therefore, $\widehat{X}$ is an optimal solution of (1.2.1). □