INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Redundancy in Semidefinite Programming

by

Shafiu Jibrin

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mathematics and Statistics

Carleton University
Ottawa, Ontario, Canada
December 1997

© Copyright 1997
Shafiu Jibrin

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-32337-4

Canada

The undersigned hereby recommend to
The Faculty of Graduate Studies and Research
acceptance of the thesis,
Redundancy in Semidefinite Programming
submitted by
Shafu Jibrin
in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

Dr K. S. Williams, Chairman
Department of Mathematics and Statistics

Dr I. Pressman, Thesis Supervisor

Dr. Lieven Vandenberghe, External Examiner

Carleton University

©S. Jibrin 1997

Abstract

This work is motivated by the increasing interest in semidefinite programming (SDP) and the associated demand for ways of eliminating the redundant constraints before the problem is solved. A *redundant constraint* is one that can be removed without changing the feasible region of the problem. The remaining constraints are called *nonredundant*.

We call the linear matrix inequality (LMI)

$$A_0 + \sum_{i=1}^n x_i A_i \succeq 0,$$

a *semidefinite constraint*. We address the problem of redundancy in a system of semidefinite constraints. We call this the *semidefinite redundancy problem*. The general problem is shown to be NP-hard. In this thesis, we develop some methods for dealing with the semidefinite redundancy problem.

The first, called the semidefinite-stand-hit method (SSH), applies to general semidefinite constraints and extends the stand-and-hit (SH) method of linear programming. SSH is a probabilistic algorithm that detects nonredundant constraints with the view of deleting the ones that are not detected. We also extend the Hypersphere Direction (HD) and the Coordinate Direction methods (CD) for linear

constraints to deal with semidefinite constraints. These extensions are called SHD and SCD respectfully. Since SH has a fixed hitting point, it allows an implementation of the *Undetected-First Rule*(UFR) which gives a further reduction in the amount of computation required. The SSH method implemented with UFR is denoted by SSH(UFR) or simply SSH. Our experiments indicate that SSH works better than both SHD and SCD.

The other methods are deterministic and are based on solving SDP problems. The Diagonal Semidefinite Constraint (DSC) method is one that applies to diagonalizable semidefinite constraints. We show how to improve the DSC method using SDP duality theory and a new technique for detecting duplicate linear constraints. The other method is called the *S1-redundant ellipsoid* (S1-RE) method and is applied to the problem of detecting redundant ellipsoids in a system of ellipsoids.

Finally, we study the application of SSH to the Löwner ellipsoid problem which is of interest in its own right and also arises naturally, for example, in the D-optimal experimental design problem. We also show that SSH can be used to improve the time-complexity of the S1-RE method.

Acknowledgements

First, I want to thank my supervisor, Professor Irwin Pressman for his guidance and encouragement in completing this thesis. This work would not have been completed without his help and support. Also, I wish to extend my sincere gratitude to Professor R. J. Caron, who first introduced me to the subject of redundancy in mathematical programming, when I was his Msc student at the University of Windsor. It is his inspiring talk at Carleton University that had the effect of changing the topic of my thesis to redundancy.

Secondly, it is my pleasure to thank Professor Lieven Vandenberghe and Henry Wolkowicz for their talks at Carleton University on the subject of Semidefinite Programming. I am particularly grateful to Professor Vandenberghe for his responses to my questions in the course of my research. Also, I want to thank the Director of the Institute of Graduate Studies and Research, Professor Bruce Richter for his constant support.

Lastly, I would like to express my appreciation to my family, especially my parents, Rukayyah and Abdurashid for their patience during my study.

Contents

1	Introduction	1
1.1	Notation and Terminology	1
1.2	Background and Survey of Literature	3
1.2.1	Positive Semidefinite Programming	3
1.2.2	Redundancy in Linear Programming	6
1.2.3	Redundancy in Positive Semidefinite Programming	10
1.2.4	On Complexity	12
1.3	Description of Test Problems	13
1.4	Summary of Main Results	16
2	A Probabilistic Algorithm for Detecting Nonredundant Semidefinite Constraints	18
2.1	The Semidefinite Redundancy Problem	18
2.2	The Semidefinite Redundancy Problem is NP-hard	22
2.3	Exact Penalty Functions for Semidefinite Programming	26
2.4	Semidefinite Stand-and-Hit (SSH) Algorithm	29
2.5	Stopping Rules for SSH	35

2.6	Complexity Estimates for SSH	38
2.7	SSH vs. SHD and SSH vs. SCD	42
3	SSH Complexity Reductions	46
3.1	Weighted Analytic Center	46
3.2	Is the analytic center a <i>good</i> hitting point for SSH ?	54
3.3	SSH with Undetected-First Rule	64
4	Redundant Diagonalizable Semidefinite Constraints and $S1$-redundant Ellipsoids	71
4.1	Diagonalizable Semidefinite Constraints	71
4.2	Computing SDP Dual Variables	76
4.3	Removing Duplicate Linear Constraints	86
4.4	$S1$ -redundant Convex Quadratic Constraints	88
5	Applications to the Löwner Ellipsoid Problems and Other Problems	98
5.1	Löwner Ellipsoid for $\{a^j\}_{j=1}^q$	98
5.2	Applying SSH to improve the $S1 - RE$ method	105
6	Conclusion	110
A	A Penalty function for low rank matrices	112
	Bibliography	118

List of Tables

1.1	Test Problems	14
3.1	Iterations/time to find all nonredundant constraints	61
3.2	Weight vs. Weighted analytic center in SDP1	63
3.3	Iterations/time to find all nonredundant constraints	69
A.1	Dimension m vs. Average $\bar{w}$ over $\mathcal{D}$	117

List of Figures

2.1	The angle of sight subtended at x_o by the j th constraint	36
2.2	Dimension n versus the smallest detection probability p_j	42
3.1	The feasible region of SDP1. This figure shows how $x_{ac} = (1.3292, 0.4530)$ moves to $x_{ac}(\omega) = (1.7764, -0.1568)$ under the influence of the weight vector $\omega = (50000, 1, 48000, 1, 2000)$	55
3.2	Contours of the graph of E_x over SDP1.	56
3.3	The graph of E_x over SDP1	56
3.4	Contours of of the graph of $H(x)$ over SDP1.	58
3.5	The graph of $H(x)$ over SDP1	58
3.6	Contours of of the graph of $-\tilde{H}(x)$ over SDP1	59
3.7	The graph of $-\tilde{H}(x)$ over SDP1	60
3.8	SSH(UFR) iteration from the hitting point x_0 in direction s	66
4.1	The feasible region of SDP12	82
4.2	The feasible region of SDP13. The inner ellipsoid is $\mathcal{E}_0$, and the outer dotted ellipsoid is the killer ellipsoid.	95
5.1	A set of 1000 points in $\mathbb{R}^2$ and their Löwner ellipsoid	102

5.2 The detected nonredundant points of Figure 5.1. and their Löwner ellipsoid.	102
--	-----

Chapter 1

Introduction

1.1 Notation and Terminology

$\mathbb{R}$ denotes the field of real numbers; $\mathbb{R}^n$ denotes the n -dimensional Euclidean space. The space of all $m \times m$ symmetric matrices is denoted by $\mathcal{H}_m$. The space $\mathcal{H}_m$ has dimension $m(m + 1)/2$.

Let $A = [a_{ij}]$ and $B = [b_{ij}]$ be $m \times m$ symmetric matrices, i.e., $A, B \in \mathcal{H}_m$. A is called *positive definite* if all its eigenvalues are strictly positive. A is called *positive semidefinite* if all its eigenvalues are nonnegative. If A is positive semidefinite (respectively positive definite), we write $A \succeq 0$ (respectively $A \succ 0$). The symbol $\succeq$ is the *Löwner partial order* for real matrices, i.e., $A \succeq B$ if and only if $A - B$ is positive semidefinite.

The smallest eigenvalue of A is denoted by $\lambda_{\min}(A)$ and its largest eigenvalue is denoted by $\lambda_{\max}(A)$. Therefore, $A \succeq 0$ if and only if $\lambda_{\min}(A) \geq 0$, and $A \succ 0$ if

and only if $\lambda_{\min}(A) > 0$. The eigenvalues of A are denoted in descending order by $\lambda_1(A) \geq \lambda_2(A) \geq \dots \geq \lambda_m(A)$. Therefore, $\lambda_1(A) = \lambda_{\max}(A)$ and $\lambda_m(A) = \lambda_{\min}(A)$.

Let Tr denote the *trace* of a matrix A . Tr is the sum of the diagonal elements of A and it equals the sum of the eigenvalues of A . The *spectral norm* of A is denoted by $\|A\|_2$, where

$$\|A\|_2 = [\lambda_{\max}(A^T A)]^{\frac{1}{2}}.$$

This generalizes the *Euclidean norm* of vectors.

Let $A = Q \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_m) Q^T$ be the *spectral decomposition* of A , where Q is an orthogonal matrix. For any $A \succeq 0$, $A^{\frac{1}{2}}$ denotes the *symmetric square root* of A , given by

$$A^{\frac{1}{2}} = Q \text{diag}(\lambda_1^{\frac{1}{2}}, \lambda_2^{\frac{1}{2}}, \dots, \lambda_m^{\frac{1}{2}}) Q^T$$

Clearly, $A^{\frac{1}{2}} A^{\frac{1}{2}} = A$. Since A is positive semidefinite, so is $A^{\frac{1}{2}}$, i.e., $A^{\frac{1}{2}} \succeq 0$.

For square matrices $\{A_i\}_{i=1}^k$, denote the block-diagonal matrix having $A_1, A_2, \dots, A_k$ as its block-diagonal elements, in the given order, by $\text{diag}(A_1, A_2, \dots, A_k)$. Define the inner product $A \bullet B$ of matrices A and B by

$$A \bullet B = \sum_i \sum_j a_{ij} b_{ij} = \text{Tr}(A^T B)$$

The *Frobenius norm* of A is denoted by $\|A\|_F$, where

$$\|A\|_F = [A \bullet A]^{\frac{1}{2}}.$$

1.2 Background and Survey of Literature

1.2.1 Positive Semidefinite Programming

A *semidefinite program* (SDP) is the programming problem

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0, \quad j = 1, 2, \dots, q \end{aligned} \quad (1.2.1)$$

where $A_i^{(j)}$, $i = 0, 1, \dots, n$ are $m_j \times m_j$ symmetric matrices and $x, c \in \mathbb{R}^n$. $\mathbb{R}^n$ is called the *ambient* space of the SDP problem.

SDP (1.2.1) is a convex program. The problem arises directly in many important applications such as Control Theory, Statistics and Combinatorial optimization (Alizadeh [1], Vandenberghe and Boyd [42]). It generalizes linear programming (LP) and quadratically constrained convex programming (QCQP) [42]. A linear constraint $\sum_{i=1}^n x_i a_i + b \geq 0$ can be equivalently written as

$$\sum_{i=1}^n x_i a_i + b \succeq 0$$

SDP has a duality theory that generalizes LP duality (Wolkowicz [45], Alizadeh [1], Ramana [34], Vandenberghe and Boyd [42]). There are different approaches to the duality. We use the *Lagrange-Slater* type approach of [42], because of its suitability for our problems. This is the *usual* Lagrangian dual obtained by treating SDP as a cone program.

The dual of SDP (1.2.1) is:

$$\begin{aligned}
 \max \quad & - \sum_{j=1}^q A_0^{(j)} \bullet Z_j \\
 \text{s.t.} \quad & \sum_{j=1}^q A_i^{(j)} \bullet Z_j = c_i, \quad i = 1, 2, \dots, n \\
 & Z_j \succeq 0, \quad j = 1, 2, \dots, q
 \end{aligned} \tag{1.2.2}$$

where the $m_j \times m_j$ symmetric matrices Z_j are the variables.

The primal problem (1.2.1) is called *strictly feasible* if there exists a vector x satisfying $A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succ 0$. Dual strict feasibility occurs when there is a feasible solution $Z_j \succ 0$, $j = 1, 2, \dots, q$. Optimality conditions for the SDP (1.2.1) are provided by the following theorem.

Theorem 1.2.1 *Suppose that both the primal (1.2.1) and its dual (1.2.2) are strictly feasible. Then, x is optimal if and only if there are $Z_j \succeq 0$ ($1 \leq j \leq q$) such that*

$$A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0, \quad j = 1, 2, \dots, q \tag{1.2.3}$$

$$\sum_{j=1}^q A_i^{(j)} \bullet Z_j = c_i, \quad i = 1, 2, \dots, n \tag{1.2.4}$$

$$Z_j(A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)}) = 0, \quad j = 1, 2, \dots, q \tag{1.2.5}$$

Furthermore, $\{Z_j\}_{j=1}^q$ is an optimal solution for the dual problem (1.2.2) and $c^T x = - \sum_{j=1}^q A_0^{(j)} \bullet Z_j$.

Condition (1.2.5) is called the *complementary slackness* condition. If the primal or dual is not strictly feasible, then optimal objective values of the primal and the dual need not be equal even if both exist. The assumption of strict feasibility does

not seem restrictive in practical applications of SDP [34]. The value of the primal objective function for any primal feasible point x is bigger than or equal to the dual objective value for any dual feasible solution $\{Z_j\}_{j=1}^q$ [42]. The difference between the primal and the dual objective values is called the *duality gap* associated with x and $\{Z_j\}_{j=1}^q$.

A more general SDP is the *maximum determinant* program (MAX-DET)

$$\begin{aligned} \min \quad & c^T x + \log \det(F(x)^{-1}) \\ \text{s.t.} \quad & A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0, \quad j = 1, 2, \dots, q \\ & F(x) = F_0 + \sum_{i=1}^n x_i F_i \succ 0 \end{aligned} \quad (1.2.6)$$

where $A_i^{(j)}$ are $m_j \times m_j$ symmetric matrices. All F_i are $l \times l$ symmetric matrices, and $x, c \in \mathbb{R}^n$. $\log \det(M)$ is the natural logarithm of the determinant of M .

The dual of MAX-DET (1.2.6) is

$$\begin{aligned} \max \quad & \log \det(W) - \sum_{j=1}^q A_0^{(j)} \bullet Z_j - F_0 \bullet W + l \\ \text{s.t.} \quad & F_i \bullet W + \sum_{j=1}^q A_i^{(j)} \bullet Z_j = c_i, \quad 1 \leq i \leq n \\ & W \succeq 0, \quad Z_j \succeq 0, \quad j = 1, 2, \dots, q \end{aligned} \quad (1.2.7)$$

where Z_j and W are respectively $m_j \times m_j$ and $l \times l$ symmetric matrices.

The MAX-DET problem is thoroughly described by Vandenberghe et al [43]. It reduces to an SDP, when $F(x) = 1$. The MAX-DET problem also arises in computational geometry, experimental design, information and communication theory.

The duality theory mentioned above for SDP, applies to the MAX-DET problem [43]. Nesterov and Nemirovskii in 1988 [30] showed how interior point methods for linear programming can be generalized to all convex programming problems. The generalization applies to convex programs with readily computable *self-concordant* barrier function. Over the years, several polynomial time interior point methods have been developed for solving SDP. They include Alizadeh [1], Nesterov and Nemirovskii [30] and Vandenberghe et al ([41],[42]).

1.2.2 Redundancy in Linear Programming

Even though fast computers and improved linear programming are now available, it is advantageous to detect and eliminate redundant constraints. A redundant constraint is one that can be removed without changing the feasible region for the problem. The remaining constraints are called *nonredundant*. Removing redundant constraints from a problem gives rise to a more efficient model, and can significantly reduce the solution time of the problem (Andersen and Andersen [3], Karwan et al. [25], Caron et al [15]). A proper formulation of an LP should contain as few constraints and variables as possible. Several methods for identifying redundant linear constraints are discussed in ([25], [3]). Many of them are LP based deterministic methods, which require solving an LP to determine whether or not a constraint is redundant.

Another approach uses probabilistic *Hit-and-Run* methods to identify the nonredundant linear constraints bounding the feasible region. Each iteration of a Hit-and-

Run method begins with the generation of a random line through an interior point of the feasible region. This hits the boundary of the feasible region at two points on two nonredundant constraints. After a finite number of iterations, the detected constraints are declared nonredundant. There is a possibility that some constraints have been missed. Let $\mathcal{R}$ be the feasible region defined by a system of q linear constraints

$$a_j^T x + b_j \leq 0, \quad j = 1, 2, \dots, q$$

where $a_j \in \mathbb{R}^n$ and b_j is a scalar. Assume that $\mathcal{R}$ is bounded and it has a non-empty interior. We describe the three most common Hit-and-Run methods.

Hypersphere Direction (HD):

The hypersphere direction method (HD) was described by Boneh and Golan [9] and independently by Smith [37]. An interior point x_0 of $\mathcal{R}$ is chosen and a direction vector s is generated uniformly from the unit hypersphere. The point x_0 and the random direction s together define a line $\{x_0 + \sigma s \mid \sigma \in \mathbb{R}\}$. The intersection points of the line with the boundary of $\mathcal{R}$ are called the *hit points* of $\mathcal{R}$, and the point x_0 is called the *hitting point* of $\mathcal{R}$. The two hit points identify two constraints. In the next iteration, the hitting point x_0 is chosen uniformly along the feasible line segment of the previous iteration.

The direction vector s is chosen by

$$s = \frac{\tilde{s}}{\|\tilde{s}\|_2}, \tag{1.2.8}$$

where $\tilde{s} \in \mathbb{R}^n$ with entries chosen randomly from the $N(0,1)$ normal distribution. It takes $2n$ multiplications/divisions and $n - 1$ additions to compute s . Each iteration

of HD involves computing the distances

$$\frac{b_j - a_j^T x_0}{a_j^T s}. \quad (1.2.9)$$

from the hitting point x_0 to each of the q constraints $a_j^T x + b_j \leq 0$. From (1.2.8) and (1.2.9), we see that each iteration of HD consists of $2n + (2n + 1)q$ multiplications/divisions and $n - 1 + (2n - 1)q$ additions/subtractions, i.e., $4nq + 3n - 1$ flops. HD is *uniformly recurrent* and the interior points it generates are asymptotically uniform [6].

Coordinate Direction (CD):

Telgen [39] introduced the coordinate direction method (CD) in order to reduce the time complexity of HD. CD differs from HD only in that CD picks direction vectors s chosen uniformly from the $2n$ directions parallel to the coordinate axes. Let e_i denote the canonical unit vector with 1 in the i th entry and 0 elsewhere. In (1.2.9) the term $a_j^T s = a_j^T e_i = (a_j)_i$. Hence, in each iteration CD computes one inner product $a_j^T x_0$ for all the q constraints. The cost per iteration is $(n + 1)q$ multiplications/divisions and nq additions, i.e., $(2n + 1)q$ flops. CD is not *uniformly recurrent*, but the interior points generated are asymptotically uniform [6].

Stand-and-Hit (SH):

The Stand-and-Hit method (SH) is a variation of HD, where the hitting point is fixed. As in HD, SH generates direction vector s uniformly from the unit hypersphere. Since x_0 is fixed, there is no need for computing $b_j - a_j^T x_0$ in (1.2.9). From (1.2.8) and (1.2.9), the main computational effort in an iteration requires

$2n + (n + 1)q$ multiplications/divisions and $n - 1 + (n - 1)q$ additions, i.e., a total of $2nq + 3n - 1$ flops.

We see that SH and CD each has a smaller iteration cost than that of HD. Since SH iteration takes $2nq + 3n - 1$ flops, and CD $(2n + 1)q$ flops, SH has a lower iteration cost than that of CD when $3n - 1 < q$. Also, since SH has a fixed hitting point, it allows an implementation of the *Undetected-First Rule*(UFR) ([8],[24],[15]) which gives a further reduction in the amount of computation required. Furthermore, SH implemented from a *good point* can significantly reduce the number of iterations for complete detection of the nonredundant constraints [24].

The first experiments of SH were done by Jibrin [24]. There is an extensive discussion of SH by Caron et al [15]. By fixing the iteration point of HD to get SH, we see that the cost per iteration reduces from $4nq + 3n - 1$ flops to $2nq + 3n - 1$ flops. Since CD iteration requires only $(2n + 1)q$ flops, it would be advantageous to have an SH version of CD. There is none, because CD uses only the $2n$ directions. Clearly, hitting the boundary of $\mathcal{R}$ in random coordinate directions from a fixed point can detect at most $2n$ constraints.

Hit-and-Run methods can be applied to convex quadratic constraints [32]. Semidefinite constraints extend convex quadratic constraints. Here, we generalize CD, HD and SH to semidefinite constraints. LP based methods for redundancy in linear constraints can not, in general, be extended to semidefinite constraints. This fact is elaborated in section 2.1.

Given an infeasible set of constraints, sometimes, it is important to remove some of the constraints to achieve feasibility. An important contribution in this direction, is that of Chinneck ([16], [17]).

1.2.3 Redundancy in Positive Semidefinite Programming

All interior point methods for solving SDP are based on similar principles [42], and the number of constraints has an impact on the solution time of the methods. Also, for each method the memory requirement per iteration increases as the number of constraints increases. To illustrate these points, we briefly discuss primal-dual potential reduction methods as given in the survey [42] for solving SDP (1.2.1). Each iteration of a primal-dual potential reduction method reduces the duality gap. If both the primal and its dual are strictly feasible, then the method converges after roughly $O(\sqrt{\sum_{j=1}^q m_j})$ iterations. In practice, the number of iterations is often independent of problem size. Primal-dual potential reduction methods are based on the potential function

$$\begin{aligned} \phi(x, Z_1, \dots, Z_q) &= \left(\sum_{j=1}^q m_j + k \sqrt{\sum_{j=1}^q m_j} \right) \log \left(\sum_{j=1}^q \text{Tr}(A^{(j)}(x) Z_j) \right) \\ &\quad - \sum_{j=1}^q \log \det(A^{(j)}(x)) - \sum_{j=1}^q \log \det(Z_j) - \sum_{j=1}^q m_j \log \left(\sum_{j=1}^q m_j \right), \end{aligned}$$

where k is a constant satisfying $k \geq 1$. The main steps are:

Initialization: Choose strictly feasible $x, \{Z_j\}_{j=1}^q$.

Repeat

Step 1: Calculate descent directions s and $\{S_j\}_{j=1}^q$.

Step 2: Find σ and $\{\alpha_j\}_{j=1}^q$ that minimizes

$$\phi(x + \sigma s, Z_1 + \alpha_1 S_1, \dots, Z_q + \alpha_q S_q).$$

Step 3: Set $x = x + \sigma s$ and $Z_j = Z_j + \alpha_j S_j$.

Until duality gap $\leq \epsilon$.

The cost of step 2 is small compared to that of step 1. The descent directions s and $\{S_j\}_{j=1}^q$ are given by the solution of the linear system

$$P_j S_j P_j + \sum_{i=1}^n s_i A_i^{(j)} = -D_j, \quad j = 1, 2, \dots, q \quad (1.2.10)$$

$$\sum_{j=1}^q \text{Tr}(A_i^{(j)} S_j) = 0, \quad i = 1, 2, \dots, n \quad (1.2.11)$$

where $\{P_j \succ 0\}_{j=1}^q$ and $\{D_j\}_{j=1}^q$ are symmetric matrices that change in each iteration. Primal-dual algorithms differ in the choice of $\{P_j\}_{j=1}^q$ and $\{D_j\}_{j=1}^q$. Equations (1.2.10) and (1.2.11) form a linear system having $n + \sum_{j=1}^q m_j(m_j + 1)/2$ equations and variables. This linear system is a least squares problem, whose solution requires $O(n^2(\sum_{j=1}^q m_j)^2)$ flops.

Since each j th constraint corresponds to the term m_j , the cost per iteration increases with an increasing number of constraints. As a result, SDP software such as *SDPSOL*© software (Wu and Boyd [46]) and the MATLAB LMI toolbox are restricted to problems of moderate size. This indicates a need of methods for detecting and eliminating redundant semidefinite constraints. Our main interest in problem (1.2.1) and (1.2.6) lies with the feasible region. We are primarily concerned

here with the problem of detecting and classifying the constraints that surround the feasible region. There has been little research in this area to date.

1.2.4 On Complexity

In order to discuss complexity of problems it is convenient and customary to analyze *decision problems* only. A *decision problem* X is a problem having two possible answers, either *TRUE* or *FALSE*.

More formally, a decision problem is a pair $X = (D, F)$ with $F \subseteq D$, where the elements of D are finite binary words. D is called the set of *instances* of X ; F is called the subset of *feasible instances* of X . Given an instance $y \in D$, we wish to determine whether or not $y \in F$. The only acceptable answers are TRUE or FALSE. A decision version of an optimization (minimization) problem is: Given an instance of the problem and an integer K , is the optimal objective value less than or equal to K ?

Consider an instance y of a decision problem. If M is the largest absolute value of its input data, then $\lceil \log M + 1 \rceil$ binary digits are required to record y . The *size* $|y|$ of the instance y is defined to be the number $\lceil \log M + 1 \rceil$.

The information used to check the validity of a TRUE instance of a decision problem is called a *certificate*. Let Z_+ denote the set of positive integers. Given a class of problems and an algorithm, the *time complexity* function $f : Z_+ \rightarrow Z_+$ of this algorithm is defined as follows: $f(n)$ is the maximum time needed to solve any

problem instance y of size $|y| = n$. The algorithm is called *polynomial time* if $f(n)$ is *polynomially bounded* in the size n of y , i.e., if there is a polynomial $g(x)$ and a constant $c > 0$ such that $f(n) \leq cg(n)$. The class of problems that can be solved by a polynomial-time algorithm is denoted by $\mathcal{P}$.

A decision problem belongs to the class $\mathcal{NP}$ if given any TRUE instance y of the problem, there exists a certificate bounded by a polynomial in the size of y whose validity can be checked by a polynomial-time algorithm. The class $\mathcal{NP}$ can be defined more formally using *non-deterministic Turing machines* [29].

Suppose $X_i = (D_i, F_i)$, $i = 1, 2$ are two decision problems. X_1 is said to be *polynomially transformable* to X_2 if there is a function $g : D_1 \rightarrow D_2$ that is polynomially bounded in the size of $y \in D_1$ such that for every $y \in D_1$, $y \in F_1 \iff g(y) \in F_2$.

A problem that belongs to class $\mathcal{NP}$ is said to be $\mathcal{NP}$ – **complete** if every $\mathcal{NP}$ problem can be polynomially transformed to it. A problem is called $\mathcal{NP}$ – **hard** if every $\mathcal{NP}$ – *complete* problem can be polynomially transformed to it. Thus, to solve an $\mathcal{NP}$ – *hard* problem is at least as difficult as it is to solve any $\mathcal{NP}$ – *complete* problem.

1.3 Description of Test Problems

Table 1.1 is a description of some of the semidefinite constrained regions used as the test problems.

Problem	Dimension n	Size of Matrices m_j	Nonredundant	Redundant
SDP1	2	2	4	1
SDP2	3	4	14	7
SDP3	5	6	16	5
SDP4	5	6	30	11
SDP5	7	8	20	1
SDP6	10	11	14	2
SDP7	20	2	437	563
SDP8	30	5	352	648
SDP9	7	3	72	929
SDP10	11	4	131	370
SDP11	51	10	204	297

Table 1.1: Test Problems

The constraints of SDP1 are given on pages 54 and 55, and a diagram of its feasible region is given in Figure 3.1. The test problems SDP2, SDP3, SDP4, SDP5 and SDP6 are ellipsoids of the form

$$\begin{bmatrix} I & Xa^j + y \\ (Xa^j + y)^T & 1 \end{bmatrix} \succeq 0, \quad j = 1, 2, \dots, q. \quad (1.3.1)$$

$$X \succ 0 \quad (1.3.2)$$

where q is a given positive integer, and each a^j is a given vector in $\Re^n$ whose entries are generated *randomly* from a uniform distribution on the interval $(0,1)$. The variables are X and y , where X is an $n \times n$ symmetric matrix and y is an n vector. In

SDP2, $q = 20$, and there are a total of 21 constraints including (1.3.2). By construction, with the exception of SDP1, the test problems are essentially random. The constraints (1.3.1) and (1.3.2) arise in the dual formulation of the *D-experimental design* problem [43].

Problems SDP7 and SDP8 are of the form

$$A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0, \quad j = 1, 2, \dots, q$$

where we are given the matrices $A_i^{(j)} = (A_i^{(j)})^T \in \mathbb{R}^m \times \mathbb{R}^m$, $A_0^{(j)}$ is the identity matrix, and x is the variable. Each $A_i^{(j)}$ is generated randomly by generating its entries from the $N(0,1)$ normal distribution. The remaining test problems SDP9, SDP10 and SDP11 are Lyapunov inequalities of the form

$$tI - A_j^T P + P A_j \succeq 0, \quad j = 1, 2, \dots, q \quad (1.3.3)$$

$$-t + \gamma \geq 0 \quad (1.3.4)$$

where we are given the matrices $A_j \in \mathbb{R}^m \times \mathbb{R}^m$. The entries of A_j are chosen from the $N(0,1)$ normal distribution, and γ is a given positive number. Lyapunov inequalities arise in stability analysis of dynamical systems [41]. We note that all of the test problems can be expressed in the form (1.2.1). The structure of the Lyapunov inequalities was considered in our experiments. The structure of the other test problems was ignored.

1.4 Summary of Main Results

We give an introduction to the theory of redundancy in Linear programming in Chapter 1. We also introduce the theory of semidefinite programming.

In Chapter 2, we present the semidefinite redundancy problem and show that in general it is NP-hard. We develop the semidefinite-Stand-and-Hit (SSH) method for detecting nonredundant semidefinite constraints, and give an efficient way of implementing the hitting step. We discuss the time complexity of SSH, and show the cost per iteration to be polynomial in the number of variables and the dimensions of the matrices. We give two Hit-and-Run variations of SSH, namely the co-ordinate direction method (SCD), and the hypersphere direction method (SHD).

In Chapter 3, we introduce the weighted analytic center and the analytic center of a semidefinite constrained region. We show that not every point of a semidefinite constrained region is a weighted analytic center. Our experiments indicate that the analytic center is a *good point* for implementing the SSH method.

We describe how to implement the Undetected-First Rule(UFR) in SSH as a method that can significantly reduce the cost per iteration. We call the new method SSH(UFR) or simply SSH. Experiments indicate there is up to a 1/3 time savings for the complete detection of the nonredundant constraints. Our experiments indicate that SSH implemented from the analytic center is a better method than both SHD and SCD. The SSH method requires less computer memory to implement than

that required by interior point methods for solving SDP. This is because the main computational step of SSH deals with one constraint at a time while methods for solving SDP deal with all the constraints together at one time. This means SSH might be as as a preprocessor for a problem, where the SDP solver failed due to computer memory limitations.

In Chapter 4 we give a method for removing redundant diagonalizable semidefinite constraints from a system consisting of a number of semidefinite constraints. We show how SDP duality improves this method. We also give an efficient technique for detecting duplicate linear constraints in the system. We introduce $S1$ -redundant ellipsoids, and give a polynomial-time method for detecting $S1$ -redundant ellipsoids, based on solving an SDP program.

Chapter 5 treats applications of SSH method to the Löwner ellipsoid problem and other problems. We show how the SSH method detects nonredundant quadratic constraints, and we describe a more efficient way of implementing SSH in this case. We describe how SSH can be used to reduce the number of the variables in the D-optimal Experimental Design Problem, and how to use SSH to improve the method of chapter 4 for removing $S1$ -redundant ellipsoids.

In chapter 6, we discuss some open problems and possible ways to improve the SSH method.

Chapter 2

A Probabilistic Algorithm for Detecting Nonredundant Semidefinite Constraints

We develop the SSH method for detecting nonredundant constraints and discuss its time and space complexity. We also present two variations of the SH method, namely SCD and SHD, and compare the costs per iteration between the three methods.

2.1 The Semidefinite Redundancy Problem

We will often refer to the linear matrix inequality (LMI)

$$A_0 + \sum_{i=1}^n x_i A_i \succeq 0,$$

as a *semidefinite constraint*. We present a probabilistic algorithm for detecting the *nonredundant* semidefinite constraints in a system of semidefinite constraints. The method will be called the *Semidefinite Stand-and-Hit method* (SSH). It is an extension of the Stand-and-Hit method (SH) for linearly constrained regions [24].

Consider the system of semidefinite constraints or LMI's:

$$A^{(j)}(x) := A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0, \quad j = 1, 2, \dots, q \quad (2.1.1)$$

where $A_i^{(j)}$, $i = 0, 1, \dots, n$ are all $m_j \times m_j$ symmetric matrices and $x \in \Re^n$. We say that constraint $A^{(j)}(x) \succeq 0$ is *strictly feasible*, if for some $x \in \Re^n$, $A^{(j)}(x) \succ 0$. We assume that all $A^{(j)}(x) \succeq 0$ are strictly feasible. The region

$$\mathcal{R} = \{x \mid A^{(j)}(x) \succeq 0, \quad j = 1, 2, \dots, q\}, \quad (2.1.2)$$

is called the *semidefinite feasible region* defined by the semidefinite constraints (2.1.1). We say $\mathcal{R}$ is *full-dimensional* if it has non-empty interior. We shall assume that $\mathcal{R}$ is bounded and full dimensional.

For $k = 1, 2, \dots, q$, define the regions

$$\mathcal{R}_k = \{x \mid A^{(j)}(x) \succeq 0, \quad j = 1, 2, \dots, k-1, k+1, \dots, q\}. \quad (2.1.3)$$

Clearly $\mathcal{R} \subseteq \mathcal{R}_k$.

Definition 2.1.1 *The semidefinite constraint $A^{(k)}(x) \succeq 0$ is called redundant if $\mathcal{R} = \mathcal{R}_k$. The constraint is called nonredundant if $\mathcal{R} \neq \mathcal{R}_k$.*

Therefore, a constraint is redundant in $\mathcal{R}$ if and only if its removal does not change the feasible region $\mathcal{R}$. The *semidefinite redundancy problem* is:

Given the set of semidefinite constraints $\{A^{(j)}(x) \succeq 0\}_{j=1}^q$, is the k th constraint $A^{(k)}(x) \succeq 0$ redundant ?

The following theorem gives a necessary and sufficient condition for testing the k th semidefinite constraint to be redundant. It turns a redundancy problem into an eigenvalue problem.

Theorem 2.1.1 $A^{(k)}(x) \succeq 0$ is redundant if and only if the semidefinite program SDP_k :

$$\begin{aligned} \min \quad & \lambda_{\min}(A^{(k)}(x)) \\ & x \in \mathcal{R}_k \end{aligned} \tag{2.1.4}$$

has an optimal solution, say x^* , satisfying $\min \lambda_{\min}(A^{(k)}(x^*)) \geq 0$.

Proof: If $A^{(k)}(x) \succeq 0$ is redundant, then for every $x \in \mathcal{R}_k$, $A^{(k)}(x) \succeq 0$. This implies that for every $x \in \mathcal{R}_k$, $\lambda_{\min}(A^{(k)}(x)) \geq 0$. Hence, the SDP (2.1.4) has an optimal solution x^* satisfying $\lambda_{\min}(A^{(k)}(x^*)) \geq 0$.

Conversely, suppose x^* is an optimal solution of the SDP (2.1.4) and that $A^{(k)}(x) \succeq 0$ is nonredundant. Then, for some $y \in \mathcal{R}_k$, we have $\lambda_{\min}(A^{(k)}(y)) < 0$. Therefore $\lambda_{\min}(A^{(k)}(x^*)) < 0$, and $A^{(k)}(x^*) \not\succeq 0$. $\square$

Corollary 2.1.1 $A^{(k)}(x) \succeq 0$ is nonredundant if the semidefinite program

$$\begin{aligned} \min \quad & \text{Tr}(A^{(k)}(x)) \\ & x \in \mathcal{R}_k \end{aligned} \tag{2.1.5}$$

has an optimal value of the objective function which is strictly negative.

Proof: If $\text{Tr}(A^{(k)}(x^*)) = \lambda_1(A^{(k)}(x^*)) + \lambda_2(A^{(k)}(x^*)) + \dots + \lambda_{m_k}(A^{(k)}(x^*)) < 0$, then $\lambda_{\min}(A^{(k)}(x^*)) = \lambda_{m_k}(A^{(k)}(x^*)) < 0$. Now apply Theorem 2.1.1. $\square$

The function $\text{Tr}(A^{(k)}(x))$ is linear, so (2.1.5) can be solved by SDPSOL code [46]. According to Corollary 2.1.1, if the optimal objective value of (2.1.5) is negative, then constraint k is redundant. If the objective value is nonnegative, then the constraint k may or may not be redundant.

Lemma 2.1.1 *Let $A(x) = A_0 + \sum_{i=1}^n x_i A_i$, where A_0, A_i are given symmetric $m \times m$ matrices. The minimum eigenvalue function, $\lambda_{\min}(A(x))$, is a continuous, concave and nondifferentiable function of x .*

Proof: The concavity and continuity of $\lambda_{\min}(A(x))$ has been demonstrated in [1]. Concavity is verified as follows: For each α , $0 \leq \alpha \leq 1$,

$$\begin{aligned} \lambda_{\min}(A(\alpha x + (1 - \alpha)y)) &= \lambda_{\min}(\alpha A(x) + (1 - \alpha)A(y)) \\ &\geq \lambda_{\min}(\alpha A(x)) + \lambda_{\min}((1 - \alpha)A(y)) \\ &= \alpha \lambda_{\min}(A(x)) + (1 - \alpha) \lambda_{\min}(A(y)) \end{aligned}$$

The above inequality holds by the fact that $\min(C + D) \geq \min(C) + \min(D)$ for any bounded subsets of real numbers C and D . Recall that by the Rayleigh-Ritz theorem [23], for any symmetric matrix E

$$\lambda_{\min}(E) = \min\{x^T E x \mid x^T x = 1\}.$$

Therefore, $\lambda_{\min}(A(x))$ is concave. Recall that, by definition, $\lambda_m(A(x)) = \lambda_{\min}(A(x))$, and that $\lambda_{m-1}(A(x))$ is the second smallest eigenvalue of $A(x)$. It is known [18] that

points x where $\lambda_m(A(x)) = \lambda_{m-1}(A(x))$ are possible points where $\lambda_{\min}(A(x))$ can be nondifferentiable, but $\lambda_{\min}(A(x))$ is differentiable at all other points. $\square$

The objective function of SDP_k (2.1.4) is thus concave and nondifferentiable and therefore, difficult to solve. In the next section we show that this SDP_k is NP-hard.

2.2 The Semidefinite Redundancy Problem is NP-hard

Lemma 2.2.1 *Let M be an $m \times m$ matrix*

$$M = \begin{bmatrix} I & y \\ y^T & r \end{bmatrix}$$

where r is a scalar and y an $m \times 1$ vector. The minimum eigenvalue of M is

$$\lambda_{\min}(M) = \frac{(1+r) - \sqrt{((r-1)^2 + 4y^Ty)}}{2} \quad (2.2.1)$$

Proof: Observe that $\text{Tr}(M) = m + r - 1$ and that

$$\begin{aligned} \det \begin{bmatrix} I & y \\ y^T & r \end{bmatrix} &= \det \left(\begin{bmatrix} I & 0 \\ -y^T & 1 \end{bmatrix} \begin{bmatrix} I & y \\ y^T & r \end{bmatrix} \begin{bmatrix} I & 0 \\ y^T & 1 \end{bmatrix} \right) \\ &= \det \left(\begin{bmatrix} I & 0 \\ -y^T & 1 \end{bmatrix} \begin{bmatrix} I & y \\ y^T & r \end{bmatrix} \begin{bmatrix} I & y \\ 0 & 1 \end{bmatrix} \right) \\ &= r - y^Ty \end{aligned}$$

By the interlacing theorem for bordered matrices [[23], p.185], the eigenvalues $\lambda_{\max} = \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m = \lambda_{\min}$ of M satisfy $\lambda_i = 1$, $i = 2, 3, \dots, m-1$. Hence,

$\text{Tr}(M) = m + r - 1$ and $\det(M) = r - y^T y$ give

$$\lambda_{\min} + \lambda_{\max} = 1 + r \quad (2.2.2)$$

$$\lambda_{\min} \lambda_{\max} = r - y^T y \quad (2.2.3)$$

Solving the two equations simultaneously gives the required result. $\square$

Corollary 2.2.1 *Let M be the $m \times m$ matrix*

$$M = \begin{bmatrix} I & y \\ y^T & 1 \end{bmatrix}$$

Then $M \succeq 0 \iff y^T y \leq 1$.

Proof: Use Lemma 2.2.1 with $r=1$ to get

$$\begin{aligned} M \succeq 0 &\iff \lambda_{\min}(M) \geq 0 \\ &\iff \frac{(1+1) - \sqrt{((1-1)^2 + 4y^T y)}}{2} \geq 0 \\ &\iff y^T y \leq 1. \quad \square \end{aligned}$$

The following quadratic programming problem QP whose objective function contains a positive definite quadratic term is NP-hard [35].

QP :

$$\begin{aligned} \max \quad & \sum_i x_i(x_i - 1) + \sum_i x_i s_i \\ & \sum_i x_i s_i \leq K \\ & 0 \leq x_i \leq 1, \quad i = 1, 2, \dots, n, \end{aligned} \quad (2.2.4)$$

where $K, s_i, i = 1, 2, \dots, n$ are given positive integers. Sartaj [35] has shown that QP (2.2.4) to be NP-hard by establishing its equivalence with the well known

NP-hard problem *sum of subsets problem* : that is, is there a subset of $S = \{s_1, s_2, \dots, s_n\}$ that sums to K ? Note that from the QP (2.2.4) we see that the objective function $\sum_i x_i(x_i - 1) + \sum_i x_i s_i < \sum_i x_i s_i \leq K$ if some i , $0 < x_i < 1$. Thus, the objective function $\sum_i x_i(x_i - 1) + \sum_i x_i s_i$ equals K if and only if S has a subset that sums to K by picking $x_i = 0$ or 1 as needed. Pardalos [33] has shown that QP (maximum) problem with a single positive definite quadratic term is NP-hard.

Theorem 2.2.1 *The semidefinite redundancy problem is NP-hard.*

Proof: In view of Theorem 2.2.2, the semidefinite redundancy problem is equivalent to the decision problem: Given an instance of SDP_k , is the optimal objective value greater than or equal to 0? It suffices to show to that SDP_k is NP-hard. We show below that QP (2.2.4) is a special case of SDP_k :

Let $c = (s_1 - 1, s_2 - 1, \dots, s_n - 1)$. The QP (2.2.4) then becomes

$$\begin{aligned} \max \quad & x^T x + c^T x \\ \text{s.t.} \quad & \sum_i x_i(c_i + 1) \leq K \\ & 0 \leq x_i \leq 1, \quad i = 1, 2, \dots, n. \end{aligned}$$

Now, consider the semidefinite constraint

$$A^{(0)}(x) = \begin{bmatrix} I & x + \frac{\varepsilon}{2} \\ (x + \frac{\varepsilon}{2})^T & 1 \end{bmatrix} \succeq 0.$$

By Lemma 2.2.1, the minimum eigenvalue of $A^{(0)}(x)$

$$\lambda_{\min}(A^{(0)}(x)) = \lambda_{\min}\left(\begin{bmatrix} I & x + \frac{\varepsilon}{2} \\ (x + \frac{\varepsilon}{2})^T & 1 \end{bmatrix}\right)$$

$$\begin{aligned}
&= \frac{2 - \sqrt{4(x + \frac{\varepsilon}{2})^T(x + \frac{\varepsilon}{2})}}{2} \\
&= 1 - \sqrt{(x^T x + c^T x + \frac{c^T c}{4})}.
\end{aligned}$$

Clearly, minimizing $\lambda_{\min}(A^{(0)})(x)$ is equivalent to maximizing $x^T x + c^T x + \frac{c^T c}{4}$, which in turn is equivalent to maximizing $x^T x + c^T x$ over the same region. Therefore, the QP becomes

$$\begin{aligned}
&\min \lambda_{\min}(A^{(0)}(x)) \\
&s.t. \quad K - \sum_i x_i(c_i + 1) \geq 0 \\
&\quad \quad x_i \geq 0, \quad i = 1, 2, \dots, n \\
&\quad \quad 1 - x_i \geq 0, \quad i = 1, 2, \dots, n
\end{aligned}$$

This is SDP_k of Theorem 2.1.1 for the problem that arises in determining the redundancy status of $A^{(0)} \succeq 0$ in the feasible region $\mathcal{R}$ determined by the system

$$\begin{aligned}
&K - \sum_i x_i(c_i + 1) \geq 0 \\
&A^{(0)}(x) \succeq 0 \\
&x_i \geq 0, \quad 0 \leq i \leq n \\
&1 - x_i \geq 0, \quad 0 \leq i \leq n.
\end{aligned}$$

Thus, QP (2.2.4) is a special case of SDP_k . $\square$

In the special case of linear constraints, Theorem 2.1.1 forms the foundation for LP based redundancy methods [9]. Theorem 2.2.2 shows why LP based methods for redundancy in linear constraints can not be extended to general semidefinite

constraints. Even LP redundancy methods that require only solving LP feasibility problems do not always extend to semidefinite constraints. For example, the *Turnover Lemma* method of Boot [10], replaces a given inequality $a^T x + b \geq 0$ by the reverse strict inequality $a^T x + b < 0$. If the new system is infeasible, then constraint $a^T x + b \geq 0$ is redundant, otherwise it is nonredundant. For a semidefinite constraint, $A_0 + \sum_{i=1}^n x_i A_i \not\geq 0$ does not imply $-(A_0 + \sum_{i=1}^n x_i A_i) \succ 0$. So, the *Turnover Lemma* does not generalize to semidefinite constraints.

Because of the problems of extending LP based methods, we begin the examination of an alternative approach to the problem. The *SSH method* to be introduced later, seems to be an attractive alternative.

2.3 Exact Penalty Functions for Semidefinite Programming

The following is guided by our desire to compute the SSH *hitting step* efficiently. We look for an easily computable penalty function for a semidefinite program. Because of the fundamental role of penalty functions in the development of interior-point methods it is reasonable to consider their role here. We shall show that they are unlikely to play a significant role in developing an efficient method for solving the SSH *hitting step*.

Consider SDP (1.2.3) given in the form

$$\min\{c^T x \mid x \in \mathcal{R}\} \quad (2.3.1)$$

where c is a given n -vector and the region

$$\mathcal{R} = \{x \in \mathbb{R}^n \mid A^{(j)}(x) \succeq 0, 1 \leq j \leq q\}.$$

Definition 2.3.1 A function $S : \mathbb{R}^n \longrightarrow \mathbb{R}$ is called a penalty function for the feasible region $\mathcal{R}$ if

$$\begin{aligned} S(x) &= 0 \text{ if } x \in \mathcal{R} \\ S(x) &> 0 \text{ if } x \notin \mathcal{R} \end{aligned}$$

For fixed $\gamma > 0$, let $x_\gamma = \operatorname{argmin}\{c^T x + \gamma S(x) \mid x \in \mathbb{R}^n\}$. It is intuitively clear that $x_\gamma \rightarrow$ an optimal point of (2.3.1), as $\gamma \rightarrow \infty$. $S(x)$ is called an *exact penalty* function if it is a penalty function and for some $\gamma > 0$ we have

$$\min\{c^T x \mid x \in \mathcal{R}\} = \min\{c^T x + \gamma S(x) \mid x \in \mathbb{R}^n\}.$$

In this case, the SDP problem is transformed into the unconstrained problem $\min\{c^T x + \gamma S(x) \mid x \in \mathbb{R}^n\}$. A complete discussion of exact penalty functions can be found in [5]. The basic steps of a penalty method to solve problem (2.3.1), using the penalty function $S(x)$, are

Initialization

Choose a tolerance value $\alpha > 0$, an initial point x_1 in $\mathbb{R}^n$, a penalty parameter $\gamma_1 > 0$, and a scalar $\beta > 1$. Let $k = 1$, and go to the main step.

Main Step

1. Using x_k as a starting point, solve $\min \{c^T x + \gamma_k S(x) \mid x \in \mathbb{R}^n\}$.
Set $x_{k+1} = \operatorname{argmin} \{c^T x + \gamma_k S(x) \mid x \in \mathbb{R}^n\}$.
2. If $\gamma_k S(x_{k+1}) < \alpha$, STOP: otherwise, set $\gamma_{k+1} = \beta \gamma_k$, replace k by $k + 1$, and return to step 1.

The *hitting step* of the SSH algorithm is a one-dimensional optimization problem. When dimension $n = 1$, step 1 of the penalty method can be solved using a derivative-free line search method [[5], ch.8].

Since $A^{(j)}(x) \succeq 0 \iff \lambda_{\min}(A^{(j)}(x)) \geq 0$, then region $\mathcal{R}$ of (2.3.1) can also be described as

$$\mathcal{R} = \{x \in \mathbb{R}^n \mid -\lambda_{\min}(A^{(j)}(x)) \leq 0, \ 1 \leq j \leq q\}.$$

Definition 2.3.2 Let $\bar{x}$ be an optimal point of (2.3.1), and $I = \{j \mid \lambda_{\min}(A^{(j)}(\bar{x})) = 0\}$. The point $\bar{x}$ is called a *Karush-Kuhn-Tucker (KKT) point* if for all $j \in I$, $\lambda_{\min}(A^{(j)}(x))$ is differentiable at $\bar{x}$ and if the following system is feasible:

$$c + \sum_{j \in I} u_j \nabla \lambda_{\min}(A^{(j)}(\bar{x})) = 0 \tag{2.3.2}$$

$$u_j \geq 0 \text{ for all } j \in I. \tag{2.3.3}$$

A feasible solution $\{u_j \mid j \in I\}$ of equations (2.3.2) and (2.3.3) are called *Lagrange multipliers*. Define the function $S : \mathbb{R}^n \longrightarrow \mathbb{R}$ by

$$S(x) = \sum_{j=1}^q \max\{0, -\lambda_{\min}(A^{(j)}(x))\}$$

Notice that $S(x)$ is zero in $\mathcal{R}$ and positive outside $\mathcal{R}$.

Theorem 2.3.1 *If the SDP (2.3.1) has a KKT point $\bar{x}$, then $S(x)$ is an exact penalty function for the SDP. That is, for some $\gamma > 0$, the optimal solution of the unconstrained problem*

$$\min\{c^T x + \gamma \sum_{j=1}^s \max\{0, -\lambda_{\min}(A^{(j)}(x))\}\}$$

is equal to that of the SDP (2.3.1), where $\lambda_{\min}(A^{(j)}(x))$ denotes the minimum eigenvalue of $A^{(j)}(x)$.

Proof: By Lemma 2.1.1, $-\lambda_{m_j}(A^{(j)}(x))$ is convex. Let $\{u_j\}$ be the Lagrange multipliers associated with $\bar{x}$. It follows from [5, Theorem 9.3.1], that for $\gamma \geq \max\{u_j \mid j \in I\}$, we have

$$\min\{c^T x \mid x \in \mathcal{R}\} = \min\{c^T x + \gamma S(x) \mid x \in \mathbb{R}^n\}. \quad \square$$

One drawback of this particular penalty function $S(x)$ is that one must compute eigenvalues repeatedly, because step 1 of the penalty method consists of several computations of $S(x)$. In the appendix, we consider a more economical penalty function $V(x)$ for low rank matrices.

2.4 Semidefinite Stand-and-Hit (SSH) Algorithm

We consider the set of semidefinite constraints $A^{(j)}(x) \succeq 0$, ($1 \leq j \leq q$) of (2.1.1) and its feasible region $\mathcal{R}$ defined in (2.1.2). It is clear from Theorem 2.1.1 that the complete classification of the set of semidefinite constraints as redundant or nonredundant is difficult. Therefore, methods that detect almost all nonredundant constraints would be useful. In this section, we introduce probabilistic methods to

attack this problem.

An interior point x_0 of $\mathcal{R}$ and a random direction s together define two line segments $\{x_0 + \sigma s \mid \sigma \geq 0\}$ and $\{x_0 - \sigma s \mid \sigma \geq 0\}$. The intersection points of each of the two line segments with the boundary of $\mathcal{R}$ are called *hit points*. The two hit points identify one or two nonredundant semidefinite constraints. This is the basic idea of the SSH. The point x_0 is called the *hitting point* of $\mathcal{R}$. In the next section we discuss how to find a *good* hitting point for SSH.

The hitting step of the algorithm (that is, SSH) requires the solution of two SDP sub-problems. Fortunately, they can be expressed as a single eigenvalue problem which can be solved using the exact penalty function $S(x)$ introduced in the previous section. This requires several computations of eigenvalues. We discuss the stopping rules for SSH in the next section.

Basic Steps of the SSH Algorithm

Define the unit hypersphere $S^{n-1} = \{(x_1, x_2, \dots, x_n) \mid x_1^2 + \dots + x_n^2 = 1\}$. The basic steps of the SSH algorithm are:

Initialization

Choose an interior point $x_0 \in \mathcal{R}$, i.e., choose a hitting point.

Repeat

(**Search Direction:**) Select a random point s on S^{n-1} .

(**Hitting Step**)

Solve **SDP1**: $\max \sigma$

$$\text{s.t. } A^{(j)}(x_0 + \sigma s) \succeq 0, \quad j = 1, 2, \dots, q.$$

Solve **SDP2**: $\max \sigma$

$$\text{s.t. } A^{(j)}(x_0 - \sigma s) \succeq 0, \quad j = 1, 2, \dots, q.$$

Let σ_1 and σ_2 respectively denote the solutions of SDP1 and SDP2. The points $x_0 + \sigma_1 s$ and $x_0 - \sigma_2 s$ identify one or two nonredundant semidefinite constraint(s).

Until a stopping rule holds.

The points $x_0 + \sigma_1 s$ and $x_0 - \sigma_2 s$ are the *hit points* of the SSH iteration. It is clear that constraints which are *duplicate* of one another, or that are coincident on some open subset of the boundary of $\mathcal{R}$ might be incorrectly identified as nonredundant. A *duplicate* is redundant and so, it can be removed without changing the feasible region $\mathcal{R}$. We assume that all extraneous duplicate constraints have been removed from our system $A^{(j)}(x)$, $j = 1, 2, \dots, q$.

Let $\mathcal{S}$ be the boundary of $\mathcal{R}$. The set $\mathcal{S}$ has dimension $n - 1$. The above assumption implies that on the boundary of $\mathcal{R}$, two or more constraints can coincide on at most a $n - 2$ dimensional subset of $\mathcal{S}$. Now, any subset of $\mathcal{S}$ with dimension $\leq n - 2$ has Lebesgue measure zero. Therefore, in one SSH iteration, the probability of detecting two constraints simultaneously in one direction is zero.

The cost of the SSH algorithm is related to the hitting step which requires the solution of two SDP problems. They can be solved using the exact penalty function $S(x)$ introduced in the previous section. This requires several computations of eigenvalues. Fortunately, these SDP's can be viewed as eigenvalue problems for symmetric matrices that can be solved efficiently.

Let x_0 be the hitting point of SSH. Since x_0 is an interior point of $\mathcal{R}$, then $A^{(j)}(x_0) \succ 0$ for each constraint j .

Definition 2.4.1 *Given a search direction vector s , define*

$$B_j(s) = -A^{(j)}(x_0)^{-\frac{1}{2}} \left(\sum_{i=1}^n s_i A_i^{(j)} \right) A^{(j)}(x_0)^{-\frac{1}{2}} \text{ for } 1 \leq j \leq q$$

Note that $B_j(s)$ is symmetric, and so, its eigenvalues are real.

Theorem 2.4.1 *Let σ_1 and σ_2 denote, respectively, the solutions of SDP1 and SDP2. Then*

$$\begin{aligned} \sigma_1 &= \min\left\{\frac{1}{\lambda_j} \mid \lambda_j \text{ is a maximum positive eigenvalue of } B_j(s), j = 1, 2, \dots, q\right\} \\ \sigma_2 &= \min\left\{-\frac{1}{\lambda_j} \mid \lambda_j \text{ is a minimum negative eigenvalue of } B_j(s), j = 1, 2, \dots, q\right\}. \end{aligned}$$

In computing σ_1 , if $B_j(s)$ has no positive eigenvalue, then its term $\frac{1}{\lambda_j}$ is not considered. Similarly for σ_2 .

Proof: We know that the feasible region $\mathcal{R}$ is bounded and that x_0 is an interior point. Since σ_1 and σ_2 are distances from x_0 to the boundary of $\mathcal{R}$, it is clear that

$$\sigma_1 = \min\{\lambda_j \mid \lambda_j > 0, \det[A^{(j)}(x_0 + \lambda_j s)] = 0, j = 1, 2, \dots, q\} \quad (2.4.1)$$

$$\sigma_2 = \min\{\lambda_j \mid \lambda_j > 0, \det[A^{(j)}(x_0 - \lambda_j s)] = 0, j = 1, 2, \dots, q\} \quad (2.4.2)$$

Now,

$$\begin{aligned}
\det[A^{(j)}(x_0 + \lambda_j s)] = 0 &\iff \det[A^{(j)}(x_0) + \lambda_j \sum_{i=1}^n s_i A_i] = 0 \\
&\iff \det\left[\frac{1}{\lambda_j} I + A^{(j)}(x_0)^{-\frac{1}{2}} \left(\sum_{i=1}^n s_i A_i^{(j)}\right) A^{(j)}(x_0)^{-\frac{1}{2}}\right] = 0 \\
&\iff \det\left[\frac{1}{\lambda_j} I - B_j(s)\right] = 0 \\
&\iff \frac{1}{\lambda_j} \text{ is an eigenvalue of } B_j(s).
\end{aligned}$$

Combining this with equations (2.4.1) and (2.4.2) gives the theorem. $\square$

An eigenvalue of $B_j(s)$ is called a *generalized eigenvalue* of the ordered pair [11].

$$\left(-\sum_{i=1}^n s_i A_i^{(j)}, A^{(j)}(x_0)\right).$$

A less stable but cheaper way to compute $B_j(s)$ is obtained by replacing the term $A^j(x_0)^{-\frac{1}{2}}$ by the inverse of the Cholesky factor of $A^j(x_0)$ [21]. Theorem 2.4.1 shows that a hitting step of SSH only requires computing the maximum and the minimum eigenvalues of $B_j(s)$. Note that because x_0 is fixed, we need only compute $A^j(x_0)^{-\frac{1}{2}}$ once throughout the detection process. With these observations, we give the SSH algorithm to find a set $\mathcal{J}$ of nonredundant constraints.

SSH Algorithm

Initialization

Set $\mathcal{J} = \emptyset$.

Choose an interior (hitting) point x_0 of $\mathcal{R}$.

Calculate $A^j(x_0)^{-1/2}$ for each $j = 1, 2, \dots, q$.

Repeat

Search Direction: Generate a random point s uniformly on the surface of S^{n-1} as follows: choose $\tilde{s}$ with n random entries, chosen from the $N(0,1)$ normal distribution, and then compute

$$s = \frac{\tilde{s}}{\|\tilde{s}\|_2}.$$

Hitting Step: Calculate $B_j(s)$ for all $j = 1, 2, \dots, q$, and find

$$\sigma_1^{(j)} = \{\frac{1}{\lambda_j} \mid \lambda_j \text{ is a max positive eigenvalue of } B_j(s), j = 1, 2, \dots, q\}$$

$$\sigma_2^{(j)} = \{-\frac{1}{\lambda_j} \mid \lambda_j \text{ is a min negative eigenvalue of } B_j(s), j = 1, 2, \dots, q\}.$$

Calculate

$$\sigma_1 = \min\{\sigma_1^{(j)}, j \in \{1, 2, \dots, q\}\}$$

$$\sigma_2 = \min\{\sigma_2^{(j)}, j \in \{1, 2, \dots, q\}\}.$$

For $1 \leq k \leq q$, if $\sigma_1^{(k)} = \sigma_1$ or $\sigma_2^{(k)} = \sigma_2$ and $k \notin \mathcal{J}$, set $\mathcal{J} = \mathcal{J} \cup \{k\}$.

Until a stopping rule holds.

After the termination of the SSH algorithm, the set $\mathcal{J}$ contains the detected nonredundant constraints.

Theorem 2.4.2 *Let x_0 be the hitting point, and s the search direction in an iteration of the SSH algorithm. Let σ_1 and σ_2 respectively denote the solutions of SDP1 and SDP2 after one iteration. For $k = 1, 2, \dots, q$,*

if $\sigma_1^{(k)} = \sigma_1$ or $\sigma_2^{(k)} = \sigma_2$, then $A^{(k)}(x) \succeq 0$ is nonredundant.

Proof: Let us assume the case $\sigma_1^{(k)} = \sigma_1$; the other case is similar. Since, the probability of simultaneously hitting two constraints in one direction is zero, constraint k satisfies $\sigma_1^{(k)} = \sigma_1$ uniquely. Then, from the definition of σ_1 , $x_0 + \sigma_1 s$ is

on the boundary of the k th constraint and in the interior of each of the remaining constraints. That is,

$$\begin{aligned}\lambda_{\min}(A^{(k)}(x_0 + \sigma_1 s)) &= 0 \\ \lambda_{\min}(A^{(j)}(x_0 + \sigma_1 s)) &> 0, \quad j = 1, 2, \dots, q, \quad j \neq k.\end{aligned}$$

For sufficiently small $\epsilon > 0$, we have

$$\begin{aligned}\lambda_{\min}(A^{(k)}(x_0 + (\sigma_1 + \epsilon)s)) &< 0 \\ \lambda_{\min}(A^{(j)}(x_0 + (\sigma_1 + \epsilon)s)) &> 0, \quad j = 1, 2, \dots, q, \quad j \neq k.\end{aligned}$$

Then $x_0 + (\sigma_1 + \epsilon)s \in \mathcal{R}_k$, but $x_0 + (\sigma_1 + \epsilon)s \notin \mathcal{R}$. Hence, $\mathcal{R}_k \neq \mathcal{R}$, which means that constraint k is nonredundant. $\square$

In Chapter 3, we give an improvement of the SSH algorithm using the *Undetected First Rule* introduced in [24].

2.5 Stopping Rules for SSH

If the number of nonredundant constraints is known, then an obvious stopping rule for SSH is to stop when all nonredundant constraints are detected. In general, the number of nonredundant constraints is unknown.

Recall that in every SSH iteration, the boundary of $\mathcal{R}$ is hit in two directions. For simplicity, we assume that the boundary of $\mathcal{R}$ is hit in one direction only. This is called *one-hit* analysis. Suppose x_0 is an interior point of the feasible region $\mathcal{R}$. Let $p_j(x_0)$, $j = 1, 2, \dots, q$ denote the probability that constraint j is detected from

x_0 in an SSH iteration. In Figure 2.1, which depicts a subset of $\mathbb{R}^2$, $p_j(x_0)$ is given by $\frac{\theta_j}{2\pi}$. If we make a large number, say n , SSH iterations at x_0 to get f_j detections of constraint j , then $p_j(x_0) \approx f_j/n$. For general n , consider a ball $B(x_0, r)$ in the interior of $\mathcal{R}$. Let S_j be the projection of the region corresponding to constraint j on the boundary of $\mathcal{R}$, onto the ball $B(x_0, r)$. Then $p_j(x_0)$ is given by the ratio of the area of S_j to the surface area of $B(x_0, r)$.

Figure 2.1: The angle of sight subtended at x_0 by the j th constraint

Since x_0 is fixed in SSH, the detection probabilities $p_j(x_0)$ are constant and independent of the iteration number. With these assumptions, several stopping rules have been given for hit-and-run methods ([7], [9], [13]).

The implementation of the stopping rules is a problem, mainly because *a priori*

knowledge of the detection probabilities $p_j(x_0)$ is required and these probabilities are generally hard to compute. We describe one stopping rule given in [13]. It is obtained by computing the number of iterations for detecting the nonredundant constraints.

For simplicity, we assume here that all of the q constraints that define $\mathcal{R}$ are nonredundant. Since hit-and-run methods such as SSH only detect nonredundant constraints, the probabilities of hitting the redundant constraints are all zero. Including them in the analysis simply introduces zero probabilities, having no impact on the results. Let Y = the number of iterations required to detect all q nonredundant constraints. Then by [13],

$$\begin{aligned}
 E_{x_0} = E(Y) &= \sum_{y=1}^{\infty} y \text{Prob}(Y = y) \\
 &= \sum_{y=1}^{\infty} \sum_{k=1}^y 1 \text{Prob}(Y = y) \\
 &= \sum_{k=1}^{\infty} \sum_{y=k}^{\infty} \text{Prob}(Y = y) \\
 &= \sum_{k=1}^{\infty} \text{Prob}(Y \geq k) \\
 &= \sum_{k=0}^{\infty} \text{Prob}(Y > k) \\
 &= \sum_{k=0}^{\infty} \text{Prob}(\text{fail to detect all nonredundant constraints in } k \text{ iterations})
 \end{aligned}$$

Using the inclusion-exclusion principle, the above simplifies to

$$E_{x_0} = \sum_{j=1}^q \frac{1}{p_j(x_0)} - \sum_{k=2}^q (-1)^k \sum_{j_1 < j_2 < \dots < j_k} \frac{1}{(\sum_{r=1}^k p_{j_r}(x_0))}. \quad (2.5.1)$$

If E_{x_0} is known then one stopping rule is to terminate after E_{x_0} iterations. However, we need to know $p_j(x_0)$ in order to calculate E_{x_0} , and these probabilities are

usually unknown. Even if $p_j(x_0)$ are known, the work to compute E_{x_0} increases exponentially with the number of nonredundant constraints. For example if $q = 5$,

$$\begin{aligned}
E_{x_0} = & \frac{1}{p_1} + \frac{1}{p_2} + \frac{1}{p_3} + \frac{1}{p_4} + \frac{1}{p_5} - \left(\frac{1}{p_1 + p_2} + \frac{1}{p_1 + p_3} + \frac{1}{p_1 + p_4} + \frac{1}{p_1 + p_5} + \frac{1}{p_2 + p_3} \right. \\
& + \frac{1}{p_2 + p_4} + \frac{1}{p_2 + p_5} + \frac{1}{p_3 + p_4} + \frac{1}{p_3 + p_5} + \frac{1}{p_4 + p_5} \left. \right) + \left(\frac{1}{p_1 + p_2 + p_3} + \frac{1}{p_1 + p_2 + p_4} \right. \\
& + \frac{1}{p_1 + p_2 + p_5} + \frac{1}{p_1 + p_3 + p_4} + \frac{1}{p_1 + p_3 + p_5} + \frac{1}{p_1 + p_4 + p_5} + \frac{1}{p_2 + p_3 + p_4} + \frac{1}{p_2 + p_3 + p_5} \\
& + \frac{1}{p_2 + p_4 + p_5} + \frac{1}{p_3 + p_4 + p_5} \left. \right) - \left(\frac{1}{p_1 + p_2 + p_3 + p_4} + \frac{1}{p_1 + p_2 + p_3 + p_5} + \frac{1}{p_1 + p_2 + p_4 + p_5} \right. \\
& \left. + \frac{1}{p_1 + p_3 + p_4 + p_5} + \frac{1}{p_2 + p_3 + p_4 + p_5} \right) + \frac{1}{p_1 + p_2 + p_3 + p_4 + p_5}
\end{aligned}$$

A few iterations of SSH can be used to give a rough estimate of the probabilities $p_j(x_0)$, and then use it to estimate the expected value E_{x_0} . We should do at least that number of iterations before SSH is terminated. With only a rough estimate of E_{x_0} , it is difficult to make any conclusion about the solution. The expression for E_{x_0} has a useful theoretical value. The dominant term in E_{x_0} is the first term $\sum_{j=1}^q \frac{1}{p_j(x_0)}$, which is dominated by the smallest values for $p_j(x_0)$ [8]. Hence, the number of iterations required to detect all nonredundant constraints depends on the constraints whose detection probabilities from the hitting point x_0 are smallest.

2.6 Complexity Estimates for SSH

In this section, we study the effect of the sizes m_j of the matrices and that of the ambient space n on SSH. First, we estimate the amount of work required to execute one iteration of the SSH. The SSH code and all other programs were written using the MATLAB© language. Since, the hitting point of SSH is fixed, the main computational effort is in the hitting step where we compute the maximum and min-

imum eigenvalues of a possibly singular matrix. Now, according to the EISPACK guide [36]: “with regard to considerations of accuracy and reliability, the methods of computing partial eigensystem fall short of the complete eigensystem methods.”

The *eig* function of MATLAB is more accurate than the *Rayleigh quotient* method for computing the largest absolute eigenvalue of a symmetric matrix. MATLAB uses EISPACK routines in its *eig* command to calculate the set of all eigenvalues of a real matrix. The function *eig* uses the *QR*-algorithm, together with accuracy-enhancement features, to find the eigenvalues of a real $m \times m$ matrix A [31]. The basic steps of the *QR*-algorithm are:

Step 1. Use at most $m - 2$ Householder matrices to get the Hessenberg form $H = Q^T A Q$, where Q is orthogonal. This reduces the computational effort in obtaining the *QR*-decomposition of A . Set $A_1 = H$ and $i = 1$.

Step 2. Use at most m Householder matrices to compute the *QR*-decomposition $A_i = Q_i R_i$, where Q_i is orthogonal and R_i is upper-triangular.

Step 3. Define $A_{i+1} = R_i Q_i$. If A_{i+1} is upper triangular stop; otherwise, set $i \rightarrow i + 1$ and repeat step 2. Note that $A_{i+1} = Q_i^T Q_i R_i Q_i = Q_i^T A_i Q_i$. So, A_{i+1} is orthogonally similar to A and of Hessenberg form.

After a finite number of steps, A_i becomes upper triangular. Step 1 requires $O(m^3)$ flops and step 2, $O(m^2)$ flops. The algorithm converges in $O(m)$ iterations of step 2. In practice, the eigenvalues of A can be accurately approximated in at most $15m^3$ arithmetic flops [27]. Since A is symmetric in the case of SSH, the Hessenberg

matrix is tridiagonal and step 2 is $O(m)$. In this case, we need fewer than $15m^3$ flops to get all of the eigenvalues. In the case of large sparse matrices, the function *eigs*, in the recently released version Matlab 5.0, can be used to efficiently compute the largest and the smallest eigenvalues only. It is an implementation of the *Arnoldi method* described in [27], but it is less accurate than the *eig* function. Rough estimates of the extremum eigenvalues of a symmetric matrix can be found using a non-iterative method given by Dixon [19] in combination with the Gerschgorin circle theorem [23].

In our experiments, we used the *eig* function to calculate all the eigenvalues of $B_j(s)$ (Definition 2.4.1). Then, the minimum and the maximum of the eigenvalues were found using the *min* and the *max* commands respectively. We estimate the total work in each iteration of SSH. Recall that in each iteration a direction vector s is chosen by

$$s = \frac{\tilde{s}}{\|\tilde{s}\|_2},$$

where $\tilde{s} \in \mathbb{R}^n$ with entries chosen randomly from the $N(0,1)$ normal distribution. This takes about $3n - 1$ flops. The hitting step requires the computation of the eigenvalues of q $m_j \times m_j$ matrices

$$B_j(s) = -A^j(x_0)^{-\frac{1}{2}} \left(\sum_{i=1}^n s_i A_i^{(j)} \right) A^j(x_0)^{-\frac{1}{2}}, \quad j = 1, 2, \dots, q.$$

Since x_0 is fixed, to compute one $B_j(s)$ requires about $2m_j^2 + nm_j^2$ multiplications and $(n-1)m_j^2$ additions, giving a total of $(2n+1)m_j^2$ flops. From above, computing the eigenvalues of one $B_j(s)$ takes at most $15m_j^3$ flops. Hence, the work estimate

per step SSH iteration is at most

$$3n - 1 + (2n + 1) \sum_{j=1}^q m_j^2 + 15 \sum_{j=1}^q m_j^3. \quad (2.6.1)$$

flops, i.e., $O(qnm_j^2 + qm_j^3)$. This measures the effect of the sizes m_j of the matrices and that of the ambient space n on the cost per iteration of SSH.

Next, we consider the effect of the dimension n on the expected number of iterations E_{x_0} required for complete detection. In equation (2.5.1), in the previous section, we found that E_{x_0} is largely determined by the smallest detection probabilities p_j . Therefore, the question that arises is how does the dimension n affect the smallest probability p_j .

For linear constraints, since $\mathcal{R}$ is bounded, the minimum number of nonredundant constraints $n + 1$ increases with increasing n . Since $\sum_{j=1}^q p_j = 1$, as n increases, then on average all the p_j 's decrease and in particular, the smallest one decreases. Equivalently, the expected number of iterations E_{x_0} increases with increasing dimension.

In the case of general semidefinite constraints, the effect of dimension on E_{x_0} is difficult to describe due to the nonlinearity of the constraints. Using simulation trials on ellipsoid constraints, we have found that p_j decreases with increasing dimension n . Given n , a problem of the type given in (1.3.1) and (1.3.2) was randomly generated and its smallest p_j was estimated by making a large number of *SHD* iterations. The *SHD* method is introduced in the next section. For each $n = 2, 3, 4, 5, 6, 7, 8$, the average smallest p_j over 100 random problems was recorded. Figure 2.2 is a graph

Figure 2.2: Dimension n versus the smallest detection probability p_j .

of an exponential fit of this data that indicates how the smallest p_j decreases as the dimension n increases for this one problem type.

2.7 SSH vs. SHD and SSH vs. SCD

In this section we introduce the corresponding semidefinite extension of two well known hit-and-run methods for linearly constrained regions, HD and CD. The corresponding semidefinite extension of HD and CD are called SHD and SCD respectively; they are described as follows.

Semidefinite Hypersphere Direction (SHD) Method:

SHD is similar to SSH, but in the former, the hitting point is not fixed. In an

SSH iteration, the two hit points determine a line segment through the hitting point. In the next iteration of SHD, a new hitting point is chosen uniformly from the line segment of the current iteration. The idea is to move the hitting point randomly about the region $\mathcal{R}$ and to give a better chance of locating all the constraints. The SHD algorithm is given as follows:

Initialization

Set $\mathcal{J} = \emptyset$.

Choose an interior (hitting) point x_0 of $\mathcal{R}$.

Repeat

Search Direction: Generate a random point s uniformly on the surface of S^{n-1} as follows: choose $\tilde{s}$ with n random entries, chosen from the $N(0,1)$ normal distribution, and then compute

$$s = \tilde{s} / \|\tilde{s}\|_2.$$

Hitting Step: Calculate $B_j(s, x_0) = -A^j(x_0)^{-\frac{1}{2}}(\sum_{i=1}^n s_i A_i^{(j)})A^j(x_0)^{-\frac{1}{2}}$

for all $j = 1, 2, \dots, q$, and find

$$\sigma_1^{(j)} = \{ \frac{1}{\lambda_j} \mid \lambda_j \text{ is a max positive eigenvalue of } B_j(s), j = 1, 2, \dots, q \}$$

$$\sigma_2^{(j)} = \{ -\frac{1}{\lambda_j} \mid \lambda_j \text{ is a min negative eigenvalue of } B_j(s), j = 1, 2, \dots, q \}.$$

Calculate

$$\sigma_1 = \min\{\sigma_1^{(j)}, j \in \{1, 2, \dots, q\}\}$$

$$\sigma_2 = \min\{\sigma_2^{(j)}, j \in \{1, 2, \dots, q\}\}.$$

For $1 \leq k \leq q$, if $\sigma_1^{(k)} = \sigma_1$ or $\sigma_2^{(k)} = \sigma_2$ and $k \notin \mathcal{J}$, set $\mathcal{J} = \mathcal{J} \cup \{k\}$.

Run: choose u from the $U(0,1)$ uniform distribution, and replace x_0 by the next iteration point $x_0 + (\sigma_2 + u(\sigma_1 - \sigma_2))s$.

Until a stopping rule holds.

Since x_0 changes, $A^j(x_0)^{-\frac{1}{2}}$ must be computed in each iteration. Recall that with SSH, we only compute $A^j(x_0)^{-\frac{1}{2}}$ once. This makes a SHD iteration computationally more expensive than one of SSH. It is well known that it requires about m_j^3 multiplications/divisions and $m_j^3 - 2m_j^2 + m_j$ additions/subtractions to compute an inverse of an $m_j \times m_j$ matrix. So, in addition to the cost (2.6.1) of a SSH iteration, each iteration of SHD requires

$$\sum_{j=1}^q (2m_j^3 - 2m_j^2 + m_j)$$

more flops. In total, each SHD iteration requires

$$3n - 1 + (2n - 1) \sum_{j=1}^q m_j^2 + 17 \sum_{j=1}^q m_j^3 \text{ flops.} \quad (2.7.1)$$

Semidefinite Coordinate Direction (SCD) Method:

SCD differs from SHD only in the choice of the direction vector s . In SCD, s is chosen uniformly from the $2n$ directions parallel to the coordinate axes. Note that, if $s = e_i$, where e_i is the canonical unit vector, then

$$\begin{aligned} B_j(s, x_0) &= -A^j(x_0)^{-\frac{1}{2}} \left(\sum_{i=1}^n s_i A_i^{(j)} \right) A^j(x_0)^{-\frac{1}{2}} \\ &= -A^j(x_0)^{-\frac{1}{2}} A_i^{(j)} A^j(x_0)^{-\frac{1}{2}}. \end{aligned}$$

Hence, the cost per iteration is independent of the direction vector s . We already have an estimate of the cost of computing $A^j(x_0)^{-\frac{1}{2}}$. After $A^j(x_0)^{-\frac{1}{2}}$ is found, calcu-

lating $B_j(s, x_0)$ requires computing $2m_j^2$ multiplications. Computing the eigenvalues of an $m_j \times m_j$ requires at most $15m_j^3$ flops. Hence, the total work estimate per iteration of SCD is at most

$$\sum_{j=1}^7 17m_j^3 + m_j \text{ flops.} \quad (2.7.2)$$

From (2.6.1), (2.7.1) and (2.7.2), it is clear that both SCD and SSH iteration are less expensive than that of SHD. The cost per iteration of SCD may be higher or lower than that of SSH. Furthermore, the cost per iteration in SCD is independent of the ambient dimension n . In contrast, the cost per iteration in both SSH and SHD increases with increasing n . In the next chapter a comparison is made between SSH implemented from the *analytic center*, SCD and SSH. We compare the total time required to detect all the nonredundant constraints.

Chapter 3

SSH Complexity Reductions

In this chapter we discuss two ways of reducing the time complexity of the SSH method. The first is the choice of the analytic center to be the hitting point. The second way uses the *Undetected-First Rule* to reduce the computational effort in an iteration.

3.1 Weighted Analytic Center

Let $\mathcal{R}$ be the feasible region defined in equation (2.1.2), i.e.,

$$\mathcal{R} = \{x \in \mathbb{R}^n \mid A^{(j)}(x) \succeq 0, \ 1 \leq j \leq q\},$$

where,

$$A^{(j)}(x) = A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0, \quad j = 1, 2, \dots, q.$$

A nonnegative function $\phi : \mathbb{R}^n \longrightarrow \mathbb{R}$ is called a *barrier function* for $\mathcal{R}$ if it is continuous over the interior of $\mathcal{R}$, and becomes infinite as x approaches the boundary

of $\mathcal{R}$ from the interior. As in Chapter 2, we assume throughout that $\mathcal{R}$ is bounded and that $\mathcal{R}$ has a non-empty interior.

Define $\phi(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$ by

$$\phi(x) = \begin{cases} \sum_{j=1}^q \omega_j \log \det(A^{(j)}(x))^{-1} & \text{if } x \in \mathcal{R} \\ \infty & \text{otherwise} \end{cases}.$$

where $\omega = (\omega_1, \omega_2, \dots, \omega_q)$ is a given strictly positive vector. $\phi(x)$ is a barrier function for $\mathcal{R}$ because it becomes infinite as x approaches the boundary of $\mathcal{R}$.

Definition 3.1.1 A strictly positive vector $\omega = (\omega_1, \omega_2, \dots, \omega_q)$ is called a *weight vector* for $\mathcal{R}$.

Definition 3.1.2 Given a weight vector ω , the *weighted analytic center* of $\mathcal{R}$ is defined by

$$x_{(ac)}(\omega) = \operatorname{argmin}\{\phi(x) \mid x \in \mathcal{R}\}.$$

If all $\omega_j = 1$, $x_{(ac)}(\omega)$ or simply $x_{(ac)}$ is called the *analytic center* of $\mathcal{R}$.

As the weight ω_k increases and all the other weights are held fixed, the weighted analytic center pushes away from the boundary of $A^{(k)}(x) \succeq 0$. Clearly,

$$\begin{aligned} x_{(ac)}(\omega) &= \operatorname{argmin}\left\{\sum_{j=1}^q \omega_j \log \det(A^{(j)}(x))^{-1} \mid x \in \mathcal{R}\right\} \\ &= \operatorname{argmax}\left\{\sum_{j=1}^q \omega_j \log \det(A^{(j)}(x)) \mid x \in \mathcal{R}\right\} \\ &= \operatorname{argmax}\left\{\prod_{j=1}^q \omega_j \det(A^{(j)}(x)) \mid x \in \mathcal{R}\right\} \end{aligned} \tag{3.1.1}$$

The last equality follows by the well known facts that the sum of logarithms is the logarithm of the product and that $\log$ is an increasing function. Note that if $A^{(j)}(x) \succeq 0$ is the linear inequality $a_j^T x - b_j \geq 0$ ($1 \leq j \leq q$), then

$$x_{(\text{ac})}(\omega) = \operatorname{argmax} \left\{ \sum_{j=1}^q \omega_j \log(a_j^T x - b_j) \mid x \in \mathcal{R} \right\}$$

This shows the definition 3.1.2 of weighted analytic center is consistent with the usual definition of weighted analytic centers for linear inequalities [4].

Lemma 3.1.1 (*Boyd and El Ghaoui [11]*) *Let $\{F_i\}_{i=0}^n$ be symmetric matrices such that $\{F_i\}_{i=1}^n$ are linearly independent, and let*

$$F(x) = F_0 + \sum_{i=1}^n x_i F_i \succeq 0.$$

Then, the barrier function

$$\phi_1(x) = \begin{cases} \log \det(F(x))^{-1} & \text{if } F(x) \succeq 0 \\ \infty & \text{otherwise} \end{cases},$$

is strictly convex over the feasible region determined by $F(x) \succeq 0$, and

$$\begin{aligned} \nabla_i \phi_1(x) &= -\operatorname{Tr}[(F(x))^{-1} F_i], \quad i = 1, 2, \dots, n \\ \nabla^2 \phi_1(x) &= H_{ij}(x) \\ &= \operatorname{Tr}[(F(x))^{-1} F_i (F(x))^{-1} F_j], \quad i, j = 1, 2, \dots, n \end{aligned}$$

In the following lemma, we now add an assumption to guarantee that the barrier function $\phi(x)$ is strictly convex.

Lemma 3.1.2 *Assume that the matrices $A_1^{(j)}, A_2^{(j)}, \dots, A_n^{(j)}$ are linearly independent for all j . Then, the barrier function $\phi(x)$ is strictly convex over $\mathcal{R}$ and the gradient*

$\nabla\phi(x)$ and the Hessian $H(x) = \nabla^2\phi(x)$ of ϕ are given respectively by

$$\begin{aligned}\nabla_i\phi(x) &= -\sum_{j=1}^q \omega_j \operatorname{Tr}[(A^{(j)}(x))^{-1}A_i^{(j)}], \quad i = 1, 2, \dots, n \\ H_{ij}(x) &= \sum_{k=1}^q \omega_k \operatorname{Tr}[(A^{(k)}(x))^{-1}A_i^{(k)}(A^{(k)}(x))^{-1}A_j^{(k)}], \quad i, j = 1, 2, \dots, n\end{aligned}$$

Proof Since the matrices $A_1^{(j)}, A_2^{(j)}, \dots, A_n^{(j)}$ are linearly independent for each j , then by the above lemma we obtain

$$\begin{aligned}\nabla_i\phi(x) &= -\sum_{j=1}^q \omega_j \nabla_i \log \det(A^{(j)}(x)) \\ &= -\sum_{j=1}^q \omega_j \operatorname{Tr}[(A^{(j)}(x))^{-1}A_i^{(j)}] \\ H_{ij}(x) &= \sum_{k=1}^q \omega_k \nabla_{ij}^2 \log \det(A^{(k)}(x)) \\ &= \sum_{k=1}^q \omega_k \operatorname{Tr}[(A^{(k)}(x))^{-1}A_i^{(k)}(A^{(k)}(x))^{-1}A_j^{(k)}].\end{aligned}$$

By Lemma 3.1.1 and since $w_j > 0$, we $\phi(x)$ is a sum of strictly convex functions over $\mathcal{R}$. so $\phi(x)$ is convex. $\square$

Lemma 3.1.2 guarantees the existence and uniqueness of the weighted analytic center, which is computed efficiently using the Damped Newton's Method of [30]. Recall that we assume throughout that $\mathcal{R}$ is bounded and has a non-empty interior.

Lemma 3.1.3 $\mathcal{R}$ is unbounded if and only if there is an $s \neq 0$ such that $\sum_{i=1}^n s_i A_i^{(j)} \succeq 0$ for all $j = 1, 2, \dots, q$.

Proof: Let x_0 be an interior point of $\mathcal{R}$. If for some $s \neq 0$, $\sum_{i=1}^n s_i A_i^{(j)} \succeq 0$ for all $j = 1, 2, \dots, q$, then the ray $\{x_0 + \sigma s \mid \sigma \geq 0\}$ is feasible. Hence, $\mathcal{R}$ is unbounded.

Conversely, suppose that $\mathcal{R}$ is unbounded. By the convexity of $\mathcal{R}$, for some direction $s \neq 0$ the ray $\{x_0 + \sigma s \mid \sigma \geq 0\}$ is feasible for all $j = 1, 2, \dots, q$. Then,

$$A^{(j)}(x_0) + \sigma \sum_{i=1}^n s_i A_i^{(j)} \succeq 0 \text{ for all } \sigma \geq 0 \quad (3.1.2)$$

$$v^T A^{(j)}(x_0) v + \sigma v^T \left(\sum_{i=1}^n s_i A_i^{(j)} \right) v \geq 0 \text{ for all } \sigma \geq 0 \text{ and } v \quad (3.1.3)$$

Hence, if $v^T (\sum_{i=1}^n s_i A_i^{(j)}) v < 0$ for some vector v , we could make the left hand side of (3.1.3) negative and contradict the inequality, by choosing σ sufficiently large. Therefore, $\sum_{i=1}^n s_i A_i^{(j)} \succeq 0$ for all $j = 1, 2, \dots, q$. $\square$

The following lemma gives a criterion for the infeasibility of the semidefinite constraint $A_0 + \sum_{i=1}^n x_i A_i \succ 0$.

Lemma 3.1.4 (*Extended Farkas Lemma*) $A_0 + \sum_{i=1}^n x_i A_i \succ 0$ is infeasible if and only if there is a nonzero matrix $Z \succeq 0$ such that

$$Z \bullet A_0 \leq 0,$$

$$Z \bullet A_i = 0 \quad (1 \leq i \leq n),$$

$$Z \bullet B > 0 \text{ for all } B \succ 0.$$

Proof: Assume each A_i is $m \times m$. We shall work in the space $\mathcal{H}_m$ of all symmetric matrices. Suppose $A(x) = A_0 + \sum_{i=1}^n x_i A_i \succ 0$ is infeasible, and let $\mathcal{P}$ be the cone of all positive semidefinite matrices in $\mathcal{H}_m$. The interior of $\mathcal{P}$, $\text{int}(\mathcal{P})$, is the set of all positive definite matrices [1], and the set

$$\{A(x) \mid x \in \mathbb{R}^n\} \quad (3.1.4)$$

does not intersect $\text{int}(\mathcal{P})$. By [11], since $\text{int}(\mathcal{P})$ is convex, there is a separating hyperplane

$$Z \bullet X = 0, \quad (3.1.5)$$

for some fixed $Z \succeq 0$, $Z \neq 0$ such that

$$Z \bullet A(x) \leq 0 \text{ for all } x \in \mathbb{R}^n \quad (3.1.6)$$

$$Z \bullet B > 0 \text{ for all } B \succ 0. \quad (3.1.7)$$

The sets A_0 , $A_0 + A_i$ and $A_0 - A_i$ ($1 \leq i \leq n$) belong to the set (3.1.4). The equation (3.1.6) gives $Z \bullet A_0 \leq 0$, and for each i ,

$$Z \bullet A_0 + x_i(Z \bullet A_i) \leq 0$$

for all $x_i \in \mathbb{R}$. It follows that for each i , $Z \bullet A_i = 0$.

Conversely, suppose the right hand side of the lemma holds. Then $Z \bullet (A_0 + \sum_{i=1}^n x_i A_i) \leq 0$ for all $x \in \mathbb{R}^n$, and $Z \bullet B > 0$ for all $B \succ 0$. It follows that $A_0 + \sum_{i=1}^n x_i A_i \not\succ 0$ for each $x \in \mathbb{R}^n$. Thus $A_0 + \sum_{i=1}^n x_i A_i \succ 0$ is infeasible. $\square$

The region $\mathcal{E} = \{x_{(\text{ac})}(\omega) \mid \omega > 0\}$ is called the *the region of weighted analytic centers*. Since the barrier function $\phi(x)$ is strictly convex over $\mathcal{R}$, each weighted analytic center $x_{(\text{ac})}(\omega)$ is an interior point. We next investigate whether or not $\mathcal{E}$ contains *all* of the interior points of $\mathcal{R}$.

Proposition 3.1.1 *There exist $m_j \times m_j$ matrices $Z^{(j)}$, $j = 1, 2, \dots, q$ such that*

$$\sum_{j=1}^q \text{Tr}[A_i^{(j)} Z^{(j)}] = 0, \quad i = 1, 2, \dots, n. \quad (3.1.8)$$

Proof: Since $\mathcal{R}$ is assumed to be bounded, by Lemma 3.1.3, the system

$$\sum_{i=1}^n s_i A_i^{(j)} \succ 0 \quad (1 \leq j \leq q)$$

is infeasible. Consider the LMI

$$\begin{aligned} & \text{diag}(A^{(1)}(x), A^{(2)}(x), \dots, A^{(q)}(x)) = \\ & \text{diag}(A_0^{(1)}, A_0^{(2)}, \dots, A_0^{(q)}) + \sum_{i=1}^n x_i \text{diag}(A_i^{(1)}, A_i^{(2)}, \dots, A_i^{(q)}) \succ 0. \end{aligned}$$

By Lemma 3.1.4, there exist $m_j \times m_j$ matrices $Z^{(j)}$, $j = 1, 2, \dots, q$ such that

$$0 \neq \text{diag } Z^{(1)}, Z^{(2)}, \dots, Z^{(q)} \succeq 0, \quad (3.1.9)$$

$$A_i^{(j)} \bullet Z^{(j)} = 0, \quad (1 \leq i \leq n),$$

$$A_0^{(j)} \bullet Z^{(j)} \leq 0, \quad (1 \leq j \leq q).$$

It follows that

$$\sum_{j=1}^q \text{Tr}[A_i^{(j)} Z^{(j)}] = 0, \quad i = 1, 2, \dots, n. \quad \square$$

The next theorem gives a sufficient condition for an interior point of $\mathcal{R}$ to be a weighted analytic center.

Theorem 3.1.1 *Let $Z^{(j)}$ ($1 \leq j \leq q$) be the matrices given by Proposition 3.1.1 and let x^* be an interior point of $\mathcal{R}$. If the equations*

$$\omega_j [A^{(j)}(x^*)]^{-1} = Z^{(j)}, \quad j = 1, 2, \dots, q \quad (3.1.10)$$

have a solution $\omega = (\omega_1, \omega_2, \dots, \omega_q)$, then $x^ = x_{ac}(\omega)$.*

Proof: Since x^* is an interior point, then $A^{(j)}(x^*) \succ 0$ and $[A^{(j)}(x^*)]^{-1} \succ 0$. The diagonal elements of a positive semidefinite matrix are nonnegative and $Z^{(j)} \succeq 0$. By comparing diagonal elements, (3.1.10) shows that $\omega_j \geq 0$. By (3.1.8), $Z^{(j)} \neq 0$ for some j . From (3.1.10) it follows that $\omega \neq 0$. So, $\omega \geq 0$ and $\omega \neq 0$. Using (3.1.9), (3.1.10) and Lemma 3.1.2 for the barrier function ϕ , we obtain

$$\begin{aligned}
 \nabla_i \phi(x^*) &= - \sum_{j=1}^q \omega_j \text{Tr}[(A^{(j)}(x^*))^{-1} A_i^{(j)}] \\
 &= - \sum_{j=1}^q \text{Tr}[\omega_j (A^{(j)}(x^*))^{-1} A_i^{(j)}] \\
 &= - \sum_{j=1}^q \text{Tr}[Z^{(j)} A_i^{(j)}] \\
 &= - \sum_{j=1}^q \text{Tr}[A_i^{(j)} Z^{(j)}] \\
 &= 0, \quad i = 1, 2, \dots, n
 \end{aligned}$$

So, the first order conditions are satisfied at x^* . Since, $\phi(x)$ is strictly convex, we get $x^* = x^{(c)}(\omega)$ [5]. $\square$

The following corollary to Theorem 3.1.1 shows that for linear constraints the region of weighted analytic centers equals the set of all interior points of $\mathcal{R}$. Later, we give an example showing that for general semidefinite constraints, the region of analytic centers *is not* all of interior points.

Corollary 3.1.1 *If the matrices $A^{(j)}(x)$ are 1×1 , i.e. $A^{(j)}(x^*) \succeq 0$ are ordinary linear inequality constraints, then there exists $\omega > 0$ such that $x^* = x_{ac}(\omega)$ for any interior point $x^* \in \mathcal{R}$.*

Proof: In this case $Z^{(j)}$ and $A^{(j)}(x^*)$ are real numbers, and equation (3.1.10) has

the required solution

$$\omega_j = Z^{(j)} A^{(j)}(x^*), \quad j = 1, 2, \dots, q. \quad \square$$

3.2 Is the analytic center a *good* hitting point for SSH ?

From previous discussions in Chapter 2, we know that the performance of SSH depends on the choice of the hitting point. As in section 5 of Chapter 2, let

$$p_j(x), \quad j = 1, 2, \dots, q$$

be the probability that constraint j is detected in an iteration of the SSH algorithm shooting from point x . Clearly, $p_j = 0$ if and only if constraint j is redundant. We would like to select an optimal hitting point from which to hit the boundary of $\mathcal{R}$. An optimal hitting point x_{opt} of $\mathcal{R}$ is one that minimizes the expected number of iterations E_x given in equation (2.5.1). That is $x_{opt} = \operatorname{argmin}\{E_x \mid x \in \mathcal{R}\}$.

To investigate x_{opt} , we consider the region SDP1 whose feasible region is given in Figure 3.1.

SDP 1

$$\begin{aligned} A^{(1)}(x) &= \begin{bmatrix} 5 & -1 \\ -1 & 2 \end{bmatrix} + x_1 \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \succeq 0 \\ A^{(2)}(x) &= \begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix} + x_1 \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \succeq 0 \end{aligned}$$

$$\begin{aligned}
A^{(3)}(x) &= \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} + x_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \succeq 0 \\
A^{(4)}(x) &= \begin{bmatrix} 3.8 & 0 \\ 0 & 3.8 \end{bmatrix} + x_1 \begin{bmatrix} -0.4 & 0 \\ 0 & -0.04 \end{bmatrix} + x_2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \succeq 0 \\
A^{(5)}(x) &= \begin{bmatrix} 2.6 & 0 \\ 0 & 2.6 \end{bmatrix} + x_1 \begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix} + x_2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \succeq 0.
\end{aligned}$$

Figure 3.1: The feasible region of SDP1. This figure shows how $x_{ac} = (1.3292, 0.4530)$ moves to $x_{ac}(\omega) = (1.7764, -0.1568)$ under the influence of the weight vector $\omega = (50000, 1, 48000, 1, 2000)$.

Figure 3.3 indicates that the point marked (*), i.e., $(3.31, -0.74)$ in Figure 3.2, is the minimum of E_x over SDP1. The corresponding optimal objective value is 116.4.

Figure 3.2: Contours of the graph of E_x over $SDP1$.

Figure 3.3: The graph of E_x over $SDP1$

Recall from section 5 of Chapter 2 that E_x is based on *one-hit* analysis. If SSH were a *one-hit* method, the expected number of SSH iterations to detect all the four constraints, hitting from (3.31,-0.74) is 116. Since SSH is *two-hit*, by [24], the expected number of iterations required is reduced by half, i.e., it is 58. From our experiments the required number of iterations is 58.06 which is approximately 58.

The work of Caron et al [13] and our conjecture in [24] suggests that an optimal hitting point is a point of $\mathcal{R}$ that balances the detection probabilities $p_j(x)$ of the nonredundant constraints, i.e., it is an equilibrium point of the $p_j(x)$'s. This opens the possibility of using other functions to estimate x_{opt} .

Recall that $p_j(x)$ is the probability that constraint j is detected in an SSH iteration. In connection with information theory, the *information content* of constraint j is defined as $-\log(p_j(x))$. A constraint with low $p_j(x)$ has a high information content. The *average information* of all the constraints is given by the *entropy* function

$$H(x) = -\sum_{j=1}^q p_j(x) \log(p_j(x)).$$

We choose $p_j(x)$'s so as to maximize the entropy function. The *Shannon point* of $\mathcal{R}$ is defined here as

$$\operatorname{argmax}\{H(x) \mid x \in \mathcal{R}\}$$

A *Shannon point* is a point of $\mathcal{R}$ that maximizes the *average* information. Figures 3.4 and 3.5 are graphs of the entropy function over SDP1. We find that (4.51,-1.94) is a *Shannon point*. This was not as good a result as we would have liked. This experiment was repeated many times with other SDP problems.

Figure 3.4: Contours of of the graph of $H(x)$ over $SDP1$.

Figure 3.5: The graph of $H(x)$ over $SDP1$

A simple transformation of the *entropy* function to $\tilde{H}(x) = -\sum_{j=1}^q \frac{1}{p_j(x)} \log(\frac{1}{p_j(x)})$ was tried. Maximizing it over the feasible region of SDP1 gave a more interesting point. In order to compare $\tilde{H}(x)$ with E_x , we graph the function $-\tilde{H}(x)$, instead of $\tilde{H}(x)$. Figures 3.6 and 3.7 show the graph of $-\tilde{H}(x)$. Its maximum occurs at (3.01,-0.59), which is close to the optimal hitting point (3.31,-0.74). The similarity between the graph of the function and that of E_x is striking. We found similar observations using other SDP feasible regions. The optimal point of $\tilde{H}(x)$ is called the *co-Shannon point*. Because the probabilities $p_j(x)$ are usually unknown, it is

Figure 3.6: Contours of of the graph of $-\tilde{H}(x)$ over SDP1

difficult to determine the optimal hitting point. The expression for E_x in equation (2.5.1) shows that the number of terms in E_x increases exponentially as the number of the constraints q increases. Therefore, even when the detection probabilities $p_j(x)$ are known, finding an optimal hitting point becomes more difficult with increasing

Figure 3.7: The graph of $-\tilde{H}(x)$ over SDP1

q . It is relatively easy to determine the *co-Shannon point* because $-\tilde{H}(x)$ is easy to compute using estimates of $p_j(x)$ based on relative frequencies of hits taken from a finite sample of iterations.

In the case of linearly constrained regions $\mathcal{R}$, the iteration points generated by the SHD algorithm are asymptotically uniformly distributed over the interior of the bounded region $\mathcal{R}$ [7]. Since their result only uses the convexity and boundedness $\mathcal{R}$, it extends at once to semidefinite constrained regions of $\mathcal{R}$.

An interior point $\hat{x}$ is called a *good point* if, with $\hat{x}$ as its hitting point, SSH takes *equal* or *less* time than SHD to detect all the nonredundant constraints. The questions that arise are whether or not *good points* exist and whether they can be computed easily. The existence of a *good point*, in the case of linearly constrained

regions, has been established in ([15], [24]). Their proofs show that the result directly extends to semidefinite constrained regions of $\mathcal{R}$. In the case of linearly constrained regions, the analytic center of $\mathcal{R}$ is a *good point* for many problems [24]. We examine the choice of the analytic center as the hitting point of the SSH algorithm.

Problem	SHD		SCD		SSH	
	Iter	Time(sec)	Iter	Time(sec)	Iter	Time(sec)
SDP1	51	3.3	110	6.7	67	1.6
SDP2	216	55.3	125	29.6	62	3.4
SDP3	774	371.5	620	286.1	230	15.1
SDP4	2,274	9,175.2	2,840	10,212.0	19,687	10,155.0
SDP5	2,278	1,398.1	1,846	1,132.2	1,102	87.4
SDP6	1,627	8,349.0	505	1,134.4	58	25.7
SDP7	999	6,907.7	1,169	6,103.7	667	1875.8
SDP8	997	6,727.6	1,123	5,643.8	468	1031.8
SDP9	839	3,513.3	1,072	4,486.5	718	637.8
SDP10	995	2,183.9	687	1,557.3	719	307.4
SDP11	990	22,374.0	817	8,237.2	252	201.7

Table 3.1: Iterations/time to find all nonredundant constraints

The preceding table compares SSH hitting from the analytic center with SHD and SCD where several simulations of each method are performed on each problem. The average number of iterations and the average time (in seconds) for complete detection of the nonredundant constraints are recorded.

We see that with the exception of SDP4, SSH hitting from the analytic center took significantly less time to detect all the nonredundant constraints. This indicates that x_{ac} is often a *good point* and it seems SSH is a better detection algorithm than both SHD and SCD. The reasons for these savings were explained in Section 2.7. Our data indicates that often SCD takes less time than SHD to detect all nonredundant constraints. This is not always the case when the problem is hard, i.e., when many iterations are required. From Section 2.7, we know that SCD has lower cost per iteration than SHD. However, SHD *looks* in every direction while SCD *looks* only in the $2n$ coordinate directions.

We investigate the use of a strategy discussed in [24] and [15] to improve the approximation of an optimal hitting point by the analytic center. It is clear that moving closer to the boundary of a constraint increases its detection probability. Our goal is to find a detection-balance point. We can move away from the detected constraints at x_{ac} by giving higher weights to constraints in proportion to the number of times they are detected. This strategy is problematic here.

Example: Consider SDP1. The analytic center of this problem is $x_{ac} = (1.3292, 0.4530)$. After 50 iterations of SSH at x_{ac} , the number of hits of constraints (1,2,3,4,5) were (50,0,48,0,2) respectively (recall that SSH hits the boundary in two directions during each iteration). Table 3.2 shows $x_{ac}(\omega)$ as the weight ω increases proportionally to (50,0,48,0,2). The undetected constraints are given weights 1 so that the region $\mathcal{R}$ remains unchanged.

weight ω	$x_{ac}(\omega)$
(1 1 1 1 1)	(1.3292 0.4530)
(500 1 480 1 20)	(1.7718 -0.1558)
(5000 1 4800 1 200)	(1.7760 -0.1567)
(50000 1 48000 1 2000)	(1.7764 -0.1568)
(500000 1 480000 1 20000)	(1.7764 -0.1568)
(5000000 1 4800000 1 200000)	(1.7764 -0.1568)

Table 3.2: Weight vs. Weighted analytic center in SDP1

The table and Figure 3.1 show that increasing ω in the above manner moves the weighted center $x_{ac}(\omega)$ closer to the undetected constraint (5). But, $x_{ac}(\omega)$ does not move close enough to the boundary to enable us to easily detect constraint (5). However, in the special case of stand-and-hit for (SH) linearly constrained regions, as noted in [24] and [15], $x_{ac}(\omega)$ can be moved close enough to the boundary by increasing the weights ω in the above manner. So, by dynamically changing the weights ω and recomputing $x_{ac}(\omega)$, we can get close to an optimal hitting point. An important fact which makes the strategy work in this special case is that every point in a linearly constrained region is a weighted analytic center, as shown in Corollary 3.1.1. Not every point of a bounded semidefinite constrained region $\mathcal{R}$ is a weighted analytic center.

Example: In the above example the point (4,-1.5) is not a weighted analytic center. Suppose that there is a weight vector $\omega = (\omega_1, \omega_2, \omega_2, \omega_4, \omega_5)$ such that $x_{ac}(\omega) =$

$(4, -1.5)^T$. The first order conditions $(\nabla\phi(x))_i = 0$, $i = 1, 2$ are satisfied at the weighted analytic center [5]. By Lemma 3.1.2, the first order conditions $(\nabla\phi(x))_i = 0$, $i = 1, 2$ give

$$\sum_{j=1}^5 \omega_j \operatorname{Tr}[(A^{(j)}(x))^{-1} A_i^{(j)}] = 0, \quad i = 1, 2.$$

This gives

$$\begin{aligned} -1.4\omega_1 - 1\omega_2 + 0.2\omega_3 - 1.1429\omega_4 + 0.3721\omega_5 &= 0 \\ -0.4\omega_1 - 0.2857\omega_2 + 2\omega_3 + 2.8571\omega_4 + 0.4651\omega_5 &= 0 \end{aligned}$$

The solution space of the system is given by

$$\begin{pmatrix} \omega_3 \\ \omega_5 \end{pmatrix} = \omega_1 \begin{pmatrix} -0.5510 \\ 2.9836 \end{pmatrix} + \omega_2 \begin{pmatrix} -0.7714 \\ 4.1770 \end{pmatrix} + \omega_4 \begin{pmatrix} -2.4489 \\ 4.3878 \end{pmatrix} \quad (3.2.1)$$

Since, the weight vector ω is nonnegative, (3.2.1) implies that $\omega_1 = \omega_2 = \omega_4 = \omega_3 = \omega_5 = 0$. Hence, there is no weight ω for which $x_{ac}(\omega) = (4, -1.5)^T$.

3.3 SSH with Undetected-First Rule

Let s be the direction vector of an SSH iteration. The direction s is called a *successful* direction if a new constraint can be detected in this direction, otherwise s is called an *unsuccessful* direction.

Recall that to determine the constraint detected in the direction s , we calculate all the hit points of the number q of constraints in the s direction, and then

choose the smallest of the hit points. The constraint corresponding to the smallest hit point is the detected constraint. To determine whether or not s is a successful direction, it is not always necessary to calculate the hit points of all the constraints in the s direction. It is clear that if a hit point of some nonredundant constraint is smaller than the smallest of the hit points of all the undetected constraints then s is an unsuccessful direction. That is, no new constraint can be detected in the s direction. This is the basic idea of the *Undetected-First Rule (UFR)* ([8],[24],[15]).

According to the rule, we first calculate the hit points of all the undetected constraints. We then calculate the hit points of the detected constraints one by one. At any time, if the hit point of some detected constraint is smaller than the smallest hit point of the undetected constraints, then s is unsuccessful; there is no need to calculate the hit points of the remaining detected constraints. The following example illustrates the idea.

Consider Figure 3.8. Suppose at the beginning of an SSH iteration constraints $\{5, 1, 2, 4\}$ have been detected in the given order, and the remaining constraints $\{3, 6, 7, 8, 9\}$ are undetected. Suppose s is the current search direction. Start calculating hit points in the direction s for the undetected constraints only. We see that constraint 9 has the closest hit point to the hitting point x_0 . Now, consider calculating hit points for the detected constraints in the given order. Constraint 5 is hit in the direction s and its hit point is closer to x_0 than constraint 9. Therefore, after calculating the hit point for 5, it is clear that no new constraint can be detected in the direction s . There is no need to calculate hit points of the other detected

constraints 1,2,5 and this reduces the computational efforts needed in the iteration.

Figure 3.8: SSH(UFR) iteration from the hitting point x_0 in direction s

We can rank order the detected constraints in descending order according to their relative frequencies and save even further. UFR was first suggested in [8] to improve the HD algorithm. It is difficult to implement UFR in HD or in CD since the next iteration requires the computation of the hit points of the boundary. Similarly, it is difficult to implement UFR in SHD and SCD. The determination of the two hit points of $\mathcal{R}$ requires that all of the constraints be considered.

In an SSH iteration, the hit points in the direction s and $-s$ are calculated simultaneously. We apply the UFR technique simultaneously in the direction s and $-s$. If both of the directions are *successful*, it is not necessary to calculate all the

hit points of the constraints, and time savings are possible. The SSH method with UFR is denoted by SSH(UFR). The following is the SSH(UFR) method to find the nonredundant constraints of the set $\{A^{(j)}(x) \succeq 0\}_{j=1}^q$. Recall from Definition 2.4.1 that

$$B_j(s) = -A^{(j)}(x_0)^{-\frac{1}{2}} \left(\sum_{i=1}^n s_i A_i^{(j)} \right) A^{(j)}(x_0)^{-\frac{1}{2}} \text{ for } 1 \leq j \leq q.$$

In the algorithm, $\sigma_1^{(j)}$ is evaluated only when $B_j(s)$ has a positive eigenvalue. A similar assumption is made in the case of $\sigma_2^{(j)}$. Let

$$\mathcal{J} = \{\text{detec}(1), \text{detec}(2), \dots, \text{detec}(N)\}$$

be the set of detected nonredundant constraints.

SSH(UFR) Algorithm

Initialization

Set $\mathcal{J} = \emptyset$.

Choose an interior (hitting) point x_0 of $\mathcal{R}$.

Calculate $A^{(j)}(x_0)^{-1/2}$ for each $j = 1, 2, \dots, q$.

Repeat

1. **Search Direction:** Generate a random point s uniformly on the surface of S^{n-1} as follows: choose $\tilde{s}$ with n random entries, chosen from the $N(0,1)$ normal distribution, and then compute

$$s = \frac{\tilde{s}}{\|\tilde{s}\|_2}.$$

2. **Hitting Step:** Calculate $B_j(s)$ for all $j \in \{1, 2, \dots, q\} - \mathcal{J}$, and find

$$\begin{aligned} \sigma_1^{(j)} &= \frac{1}{\lambda_j}, \text{ where } \lambda_j \text{ is a maximum positive eigenvalue of } B_j(s) \\ \sigma_2^{(j)} &= -\frac{1}{\lambda_j}, \text{ where } \lambda_j \text{ is a minimum negative eigenvalue of } B_j(s) \end{aligned}$$

Calculate

$$u_1 = \min\{\sigma_1^{(j)}, j \in \{1, 2, \dots, q\} - \mathcal{J}\}$$

$$u_2 = \min\{\sigma_2^{(j)}, j \in \{1, 2, \dots, q\} - \mathcal{J}\}.$$

Let $\mathcal{J} = \{\text{detec}(1), \text{detec}(2), \dots, \text{detec}(N)\}$. Set $r = 1$, STOP=false.

Repeat

Calculate $B_r(s)$, and find

$$\sigma_1^{(r)} = \frac{1}{\lambda_r}, \text{ where } \lambda_r \text{ is a maximum positive eigenvalue of } B_r(s)$$

$$\sigma_2^{(r)} = \frac{-1}{\lambda_r}, \text{ where } \lambda_r \text{ is a minimum negative eigenvalue of } B_r(s)$$

If (($\sigma_1^{(r)} < u_1$) and ($\sigma_2^{(r)} < u_2$))), then STOP=true

Set $r=r+1$

until (($r=N+1$) or (STOP=true))

If STOP=true, then exit hitting step and begin new iteration. else go to the next step

Calculate

$$\sigma_1 = \min\{u_1, \sigma_1^{(r)}, r \in \mathcal{J}\}$$

$$\sigma_2 = \min\{u_2, \sigma_2^{(r)}, r \in \mathcal{J}\}.$$

For $1 \leq k \leq q$, if $\sigma_1^{(k)} = \sigma_1$ or $\sigma_2^{(k)} = \sigma_2$ and $k \notin \mathcal{J}$, set $\mathcal{J} = \mathcal{J} \cup \{k\}$,.

Until a stopping rule holds.

Table 3.3 compares SSH with SSH(UFR) where both are implemented from the analytic center. The table shows SSH(UFR) took less time than SSH to detect all the nonredundant constraints, except for SDP9. The time saving of SSH(UFR) over SSH is small with SDP9, SDP10 and SDP10, partly because the special structure

of the Lyapunov inequalities was exploited in implementing the methods. In fact, in SDP9, the cost for implementing UFR in SSH exceeded the cost reduction. By comparing with Table 3.1, we see that SSH(UFR) took less time than both SHD and SCD in all the problems.

Problem	SSH		SSH(UFR)	
	Iter	Time(sec)	Iter	Time(sec)
SDP1	67	1.6	67	1.5
SDP2	62	3.4	62	2.5
SDP3	230	15.1	230	8.7
SDP4	19,687	10,155.0	19,687	5,941.3
SDP5	1,102	87.4	1,102	34.5
SDP6	58	25.7	58	17.9
SDP7	667	1875.8	667	1769.5
SDP8	468	1031.8	468	891.5
SDP9	718	637.8	718	716.3
SDP10	719	307.4	719	292.5
SDP11	252	201.7	252	197.8

Table 3.3: Iterations/time to find all nonredundant constraints

From Table 3.1, SCD outperformed SSH in SDP4. Interestingly, SSH(UFR) performed better with SDP4 than with any of the remaining problems. From Table 1.1, SDP4 has many nonredundant constraints and it has the largest detection time for each of the methods. These suggests that the advantage of SSH(UFR) over SSH

increases as the number of nonredundant constraints increases and as the problem gets harder.

Recall that an SSH iteration requires computing the hit points of all of the constraints. In an SSH(UFR) iteration, where r constraints have been detected in previous iterations, it is possible to avoid computing up to $r - 1$ hit points of the constraints detected previously. This is significant considering that calculating the hit point of a constraint j requires computing the eigenvalues of the matrix $B_j(s)$ (Definition 2.4.1).

After several SSH(UFR) iterations, most nonredundant constraints are detected. The remaining iterations are used to try to detect the hard to find constraints. Therefore time savings per iteration of the SSH(UFR) method increase with increasing difficulty of the problem. Our experiments (Table 3.1 and Table 3.3) indicate that the difficulty of the semidefinite detection problem increases with increasing n . From here onwards, SSH(UFR) will be denoted by SSH.

Chapter 4

Redundant Diagonalizable Semidefinite Constraints and $S1$ -redundant Ellipsoids

In the previous chapters, we presented the *probabilistic* SSH method for detecting *nonredundant* semidefinite constraints. Here, we give a *deterministic* method of detecting *redundant* diagonalizable semidefinite constraints in a system of semidefinite constraints. This is a first step towards the goal of reducing the size of the LMI's in an SDP problem. We also provide a means of detecting certain types of *redundant* ellipsoids in a system of ellipsoids.

4.1 Diagonalizable Semidefinite Constraints

We present a method for removing redundant *diagonalizable* semidefinite constraints in a bounded semidefinite constrained region. The method described here applies to SDP problems with some linear constraints. This is a generalization of the problem of removing redundant linear constraints from a system of linear inequalities ([25], [3]). This restriction still represents a significant range of problems.

We consider the semidefinite constraints (2.2.1)

$$A^{(j)}(x) := A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0, \quad j = 1, 2, \dots, q$$

By Theorem 2.2.1, the problem SDP_k (2.1.4) of determining whether or not the k th constraint is redundant is NP-hard. We will show how to handle this problem in the case where the matrices $A_i^{(k)}$, $i = 0, 1, \dots, n$ are simultaneously diagonalizable. The k th constraint

$$A^{(k)}(x) = A_0^{(k)} + \sum_{i=1}^n x_i A_i^{(k)} \succeq 0,$$

is called *diagonalizable* if the matrices $A_i^{(k)}$, $i = 0, 1, \dots, n$ are simultaneously diagonalizable. Since a symmetric matrix is diagonalizable, by [[23], Theorem 1.3.19], the k th constraint is diagonalizable if and only if the matrices $A_i^{(k)}$ are commutative. A linear constraint $\sum_{i=1}^n x_i a_i + b \geq 0$ is diagonalizable, since every a_i is a scalar.

Suppose that a subset of the semidefinite constraints $A^{(j)}(x) \succeq 0$ ($1 \leq j \leq q_0$) are diagonalizable. We present a method to remove or reduce the size of each redundant diagonalizable semidefinite constraint as follows:

Suppose the k th constraint $A^{(k)}(x) \succeq 0$ is diagonalizable. There is an orthogonal matrix Q such that $QA_i^{(k)}Q^T$ is a diagonal $D_i^{(k)}$ for each $i = 0, 1, \dots, n$. Since Q is nonsingular,

$$A_0^{(k)} + \sum_{i=1}^n x_i A_i^{(k)} \succeq 0 \iff D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)} \succeq 0.$$

Then $A_0^{(k)} + \sum_{i=1}^n x_i A_i^{(k)} \succeq 0$ is redundant in $\mathcal{R}$ if and only if $D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)} \succeq 0$

is redundant in $\mathcal{R}$. Replace the k th constraint of (4.1.1) $A^{(k)}(x) \succeq 0$ by

$$D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)} \succeq 0 \quad (4.1.1)$$

to get

$$\mathcal{R} = \mathcal{R}_k \cap \{x \in \mathbb{R}^n \mid D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)} \succeq 0\}. \quad (4.1.2)$$

Also, SDP_k becomes

SDP_k :

$$\begin{aligned} \min \quad & \lambda_{\min}(D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)}) \\ & x \in \mathcal{R}_k. \end{aligned} \quad (4.1.3)$$

For each i , let $D_i^{(k)}(1, 1), D_i^{(k)}(2, 2), \dots, D_i^{(k)}(m_k, m_k)$ denote the diagonal elements of $D_i^{(k)}$. Then for each x , $D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)}$ is a diagonal matrix with eigenvalues

$$D_0^{(k)}(r, r) + \sum_{i=1}^n x_i D_i^{(k)}(r, r), \quad r = 1, 2, \dots, m_k. \quad (4.1.4)$$

Therefore, the objective function of SDP_k (4.1.3) is a minimum of the m_k linear constraints in (4.1.4). Hence, the objective function is convex and piecewise-linear. We can handle this case by breaking the problem into m_k subproblems. First, note that constraint (4.1.1) is redundant in $\mathcal{R}$ if and only if all of its m_k diagonal entries (4.1.4) are redundant in $\mathcal{R}$. So, instead of solving SDP_k (4.1.3) to

check for redundancy of (4.1.1), we solve for each r , the problem of whether or not $D_0^{(k)}(r, r) + \sum_{i=1}^n x_i D_i^{(k)}(r, r)$ is redundant in $\mathcal{R}$. An advantage of this approach is that even when constraint (4.1.1) is nonredundant, it is possible to reduce its size. If $D_0^{(k)}(r, r) + \sum_{i=1}^n x_i D_i^{(k)}(r, r)$ is redundant, then we can delete the r th row and r th column of (4.1.1), thereby reducing the size of (4.1.1) from m_k to $m_k - 1$.

Given constraint k (4.1.1), to check if its r th diagonal entry is redundant, consider the following SDP problem: SDP_k^r :

$$\begin{aligned} \min \quad & \sum_{i=1}^n x_i D_i^{(k)}(r, r) \\ \text{st.} \quad & D_0^{(k)}(l, l) + \sum_{i=1}^n x_i D_i^{(k)}(l, l) \geq 0, \quad l = 1, 2, \dots, r-1, r+1, \dots, m_k \\ & x \in \mathcal{R}_k, \end{aligned}$$

The constraints $D_0^{(k)}(l, l) + \sum_{i=1}^n x_i D_i^{(k)}(l, l) \geq 0$, $l = 1, 2, \dots, r-1, r+1, \dots, m_k$ are ordinary linear constraints, one for each row of $D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)} \succeq 0$, excluding the r th. Our discussion shows that a diagonalizable semidefinite constraint actually defines a polyhedron in $\mathbb{R}^n$.

Theorem 4.1.1 *The r th diagonal element of $D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)} \succeq 0$ is redundant in $\mathcal{R}$ if and only if the above semidefinite program SDP_k^r has an optimal solution, say at x^* , such that $D_0^{(k)}(r, r) + \sum_{i=1}^n x_i^* D_i^{(k)}(r, r) \geq 0$.*

Proof: From (4.1.2), the feasible region of SDP_k^r together with the r th diagonal element of the new k th constraint, $D_0^{(k)}(r, r) + \sum_{i=1}^n x_i D_i^{(k)}(r, r) \geq 0$ defines the region $\mathcal{R}$. The result follows by applying Theorem 2.1.1 to the problem of determining whether or not $D_0^{(k)}(r, r) + \sum_{i=1}^n x_i D_i^{(k)}(r, r) \geq 0$ is redundant in $\mathcal{R}$. $\square$

For a diagonalized constraint $D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)} \succeq 0$, our goal is to remove it from the system or to reduce its size m_k by as much as possible. By Theorem 4.1.1, this can be done by solving all the programs SDP_k^r that arise from each diagonalized semidefinite constraint.

The Diagonalizable Semidefinite Constraint (DSC) Algorithm.

Let $A^{(j)}(x) \succeq 0$, $j = 1, 2, \dots, q_0$ be the diagonalizable constraints of $\mathcal{R}$. Set $k = 1$.

Repeat

Diagonalize $A^{(k)}(x) \succeq 0$ to get $D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)} \succeq 0$.

Set $r = 1$

Repeat

Solve SDP_k^r to get optimal solution x^* .

If $D_0^{(k)}(r, r) + \sum_{i=1}^n x_i^* D_i^{(k)}(r, r) \geq 0$, remove the r th diagonal element of $D_0^{(k)} + \sum_{i=1}^n x_i D_i^{(k)} \succeq 0$.

$r = r + 1$

Until $r = m_k + 1$

$k = k + 1$

Until $k = q_0 + 1$.

To remove all redundant linear constraints arising from the diagonalizable semidefinite constraints of $\mathcal{R}$, the DSC algorithm requires that we have to solve SDP_k^r for $(1 \leq r \leq m_k)$ and $(1 \leq k \leq q)$. This would be a significant amount of computation in a large problem. In the next two sections, we give two methods that can be used

to reduce the number of SDP_k^r s to be solved. We assume that all diagonalizable constraints have been already diagonalized and rewritten as linear constraints. The resulting system that defines the region $\mathcal{R}$ is now

$$A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0, \quad j = 1, 2, \dots, q_1 \quad (4.1.5)$$

$$a_l^T x + b_l \geq 0, \quad l = 1, 2, \dots, q_2 \quad (4.1.6)$$

By Theorem 4.1.1, $a_k^T x + b_k \geq 0$ is redundant in $\mathcal{R}$, if and only if the problem

SDP_k

$$\min \quad a_k^T x \quad (4.1.7)$$

$$s.t. \quad A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0, \quad j = 1, 2, \dots, q_1 \quad (4.1.8)$$

$$a_l^T x + b_l \geq 0, \quad l = 1, 2, \dots, k-1, k+1, \dots, q_2 \quad (4.1.9)$$

has an optimal solution, say at x^* , such that $a_k^T x^* + b_k \geq 0$. To remove the redundant constraints from the linear system (4.1.6), the DSC algorithm requires us to solve SDP_k (4.1.7) for $1 \leq k \leq q_2$. From here onwards, SDP_k will refer to the SDP_k problem (4.1.7).

4.2 Computing SDP Dual Variables

We show here how to use SDP duality to reduce the number of SDP_k^r s to be solved in the previous section. Duality is the basis of many modern approaches to mathematical programming. Given that x^* is an optimal point of SDP_k , we compute its dual variables. The following theorem gives the optimality conditions of SDP_k at x^* .

Theorem 4.2.1 x^* is optimal for SDP^k if and only if there are dual variables Z_j , u_l which are dual optimal such that:

$$\begin{aligned}
 (1) \quad & \sum_{j=1}^{q_1} A_i^{(j)} \bullet Z_j + \sum_{l=1, l \neq k}^{q_2} u_l (a_l)_i = (a_k)_i, \quad i = 1, 2, \dots, n \\
 & Z_j \succeq 0, \quad j = 1, 2, \dots, q_1 \\
 & u_l \geq 0, \quad l = 1, 2, \dots, k-1, k+1, \dots, q_2 \\
 (2) \quad & (A_0^{(j)} + \sum_{i=1}^n x_i^* A_i^{(j)}) Z_j = 0, \quad j = 1, 2, \dots, q_1 \\
 & (a_l^T x^* + b_l) u_l = 0, \quad l = 1, 2, \dots, k-1, k+1, \dots, q_2.
 \end{aligned}$$

Condition (1) gives dual feasibility of the variables Z_j , u_l ; condition (2) is the complementary slackness condition.

Proof: By taking the dual of SDP_k and using the optimality conditions of Theorem 1.2.1, we see that the j th semidefinite constraint is associated with a dual variable symmetric matrix Z_j and the l th linear constraint is associated with a dual real variable u_l for $l \neq k$. $\square$

Suppose x^* is optimal for SDP_k . The computation of the corresponding dual variables for SDP_k is simplified when x^* is *nondegenerate*. In that case the dual variables are unique [2]. The notion of degeneracy in SDP is not as simple as for LP problems.

For each $j = 1, 2, \dots, q_1$ let Q_j be $m_j \times m_j$ orthogonal matrices such that

$$A^{(j)}(x^*) = Q_j \text{diag}(\lambda_1^{(j)}, \lambda_2^{(j)}, \dots, \lambda_{r_j}^{(j)}, 0, \dots, 0) Q_j^T. \quad (4.2.1)$$

where $\lambda_1^{(j)}, \lambda_2^{(j)}, \dots, \lambda_{r_j}^{(j)}$ are the strictly positive eigenvalues of $A^{(j)}(x^*)$. So,

$$\text{rank}(A^{(j)}(x^*)) = r_j. \quad (4.2.2)$$

For each $j = 1, 2, \dots, q_1$ let

$$\tilde{Q}_j = \text{the submatrix given by the last } m_j - r_j \text{ columns of } Q_j. \quad (4.2.3)$$

Let $\mathcal{A}_1$ be the subset of the constraints $A^{(j)}(x) \succeq 0$ (4.1.8) that are binding at x^* , i.e.,

$$\begin{aligned} \mathcal{A}_1 &= \{j \in \{1, 2, \dots, q_1\} \mid \lambda_{\min}(A^{(j)}(x^*)) = 0\} \\ &= \{j_1, j_2, \dots, j_s\}. \end{aligned} \quad (4.2.4)$$

Let $\mathcal{A}_2$ be the subset of linear constraints $a_l^T x^* + b_l \geq 0$ (4.1.9) that are binding at x^* , i.e.,

$$\begin{aligned} \mathcal{A}_2 &= \{l \in \{1, 2, \dots, k-1, k+1, \dots, q_2\} \mid a_l^T x^* + b_l = 0\} \\ &= \{l_1, l_2, \dots, l_t\}. \end{aligned} \quad (4.2.5)$$

Let

$$Z_j \text{ and } u_l \text{ be the dual variables of Theorem 4.2.1.} \quad (4.2.6)$$

If $j \notin \mathcal{A}_1$, then $A^{(j)}(x^*) = A_0^{(j)} + \sum_{i=1}^n x_i^* A_i^{(j)} \succ 0$. By the complementary slackness condition (2) of Theorem 4.2.1 we have

$$Z_j = 0 \text{ for } j \notin \mathcal{A}_1 \text{ and } u_l = 0 \text{ for } l \notin \mathcal{A}_2.$$

The complementary slackness conditions (2), $A^{(j)}(x^*)Z_j = 0$, together with the choice of Q_j (4.2.1) give

$$\begin{aligned} [\text{diag}(\lambda_1^{(j)}, \lambda_2^{(j)}, \dots, \lambda_{r_j}^{(j)}, 0, \dots, 0)][Q_j^T Z_j Q_j] &= Q_j^T [A^{(j)}(x^*)Z_j] Q_j \\ &= 0. \end{aligned}$$

This together with the fact that $Q_j^T Z_j Q_j$ is symmetric, imply that

$$Q_j^T Z_j Q_j = \text{diag}(0, \dots, 0, W_j), \quad (4.2.7)$$

where W_j is an $(m_j - r_j) \times (m_j - r_j)$ symmetric matrix. We see that the nonzero dual variables of SDP_k at x^* belong to a space of dimension

$$\begin{aligned} d &= \left[\sum_{j \in \mathcal{A}_1} (m_j - r_j)(m_j - r_j)/2 \right] + \left[\sum_{k \in \mathcal{A}_2} 1 \right] \\ &= \left[\sum_{i=1}^s (m_{j_i} - r_{j_i})(m_{j_i} - r_{j_i})/2 \right] + t. \end{aligned} \quad (4.2.8)$$

We use [[2], Theorem 3] for the following definition.

Definition 4.2.1 *The feasible point x^* of SDP_k is nondegenerate if and only if*

$$[\text{diag}(\tilde{Q}_{j_1}^T A_i^{(j_1)} \tilde{Q}_{j_1}, \dots, \tilde{Q}_{j_s}^T A_i^{(j_s)} \tilde{Q}_{j_s}, (a_{l_1})_i, \dots, (a_{l_t})_i)], \quad (1 \leq i \leq n)$$

span a space of dimension d .

Theorem 4.2.2 *If x^* is a nondegenerate optimal point of SDP_k^* , then the linear system*

$$(a_k)_i = \sum_{j \in \mathcal{A}_1} [\tilde{Q}_j^T A_i^{(j)} \tilde{Q}_j] \bullet W_j + \sum_{l \in \mathcal{A}_2} u_l (a_l)_i, \quad (1 \leq i \leq n) \quad (4.2.9)$$

has unique solution W_j, u_l with W_j symmetric, $W_j \succeq 0$ and $u_l \geq 0$.

Proof: Let Z_j and u_l be the dual variables of SDP_k at x^* mentioned earlier (4.2.6). We know that $Z_j = 0$ for $j \notin \mathcal{A}_1$ and that $u_l = 0$ for $l \notin \mathcal{A}_2$. This and the dual feasibility condition (1) give

$$\sum_{j \in \mathcal{A}_1} A_i^{(j)} \bullet Z_j + \sum_{l \in \mathcal{A}_2} u_l (a_l)_i = (a_k)_i, \quad (1 \leq i \leq n). \quad (4.2.10)$$

By the orthogonality of Q_j and (4.2.7), we have

$$A_i^{(j)} \bullet Z_j = [Q_j^T A_i^{(j)} Q_j] \bullet [Q_j^T Z_j Q_j] \quad (4.2.11)$$

$$= [Q_j^T A_i^{(j)} Q_j] \bullet [\text{diag}(0, \dots, 0, W_j)] \quad (4.2.12)$$

$$= [\tilde{Q}_j^T A_i^{(j)} \tilde{Q}_j] \bullet [\tilde{Q}_j^T Z_j \tilde{Q}_j] \quad (4.2.13)$$

$$= [\tilde{Q}_j^T A_i^{(j)} \tilde{Q}_j] \bullet W_j \quad (4.2.14)$$

By (4.2.10), we have

$$\sum_{j \in \mathcal{A}_1} [\tilde{Q}_j^T A_i^{(j)} \tilde{Q}_j] \bullet W_j + \sum_{l \in \mathcal{A}_2} u_l (a_l)_i = (a_k)_i, \quad (1 \leq i \leq n). \quad (4.2.15)$$

Therefore, W_j and u_l is a solution of (4.2.9). Next, we establish the uniqueness of this solution. Since SDP_k is nondegenerate at x^* , the matrices

$$[\text{diag}(\tilde{Q}_{j_1}^T A_i^{(j_1)} \tilde{Q}_{j_1}, \dots, \tilde{Q}_{j_s}^T A_i^{(j_s)} \tilde{Q}_{j_s}, (a_{l_1})_i, \dots, (a_{l_t})_i)], \quad (1 \leq i \leq n)$$

span a space of dimension d (4.2.8). Therefore at least d of the linear equations of (4.2.9) are linearly independent. The linear system (4.2.9) has exactly d unknowns. Hence, the solution W_j and u_l obtained earlier is unique. $\square$

Theorem 4.2.3 *Let x^* be a feasible solution of SDP_k^r , and define $\tilde{Q}_j$ as in (4.2.1) and (4.2.3). If the system (4.2.9) has a feasible solution satisfying $W_j \succeq 0$, $u_l \geq 0$, then x^* is an optimal solution of SDP_k^r .*

Proof: Let $W_j \succeq 0$, $u_l \geq 0$ be a feasible solution of (4.2.9). Recall that W_j is $(m_j - r_j) \times (m_j - r_j)$, where $\text{rank}(A^{(j)}(x^*)) = r_j$. Let Z_j be the $m_j \times m_j$ matrix

$$Z_j = \begin{cases} \text{diag}(0, 0, \dots, 0, W_j) & \text{if } j \in \mathcal{A}_1 \\ 0 & \text{if } j \notin \mathcal{A}_1 \end{cases} \quad (4.2.16)$$

$$u_l = 0 \text{ for } l \notin \mathcal{A}_2. \quad (4.2.17)$$

It suffices to show that Z_j , u_l satisfy the optimality conditions of Theorem 4.2.1. Clearly,

$$Z_j \succeq 0 \quad , \quad j = 1, 2, \dots, q_1 \quad (4.2.18)$$

$$u_l \geq 0 \quad , \quad l = 1, 2, \dots, k-1, k+1, \dots, q_2 \quad (4.2.19)$$

By (4.2.16) and the same argument as in (4.2.11) through (4.2.14), we obtain

$$A_i^{(j)} \bullet Z_j = [\tilde{Q}_j^T A_i^{(j)} \tilde{Q}_j] \bullet W_j \text{ for } j \in \mathcal{A}_1. \quad (4.2.20)$$

Since W_j , u_l satisfy (4.2.9), we have

$$(a_k)_i = \sum_{j \in \mathcal{A}_1} [\tilde{Q}_j^T A_i^{(j)} \tilde{Q}_j] \bullet W_j + \sum_{l \in \mathcal{A}_2} u_l (a_l)_i, \quad (1 \leq i \leq n). \quad (4.2.21)$$

By (4.2.16), (4.2.17) and (4.2.20) we have

$$\sum_{j=1}^{q_1} A_i^{(j)} \bullet Z_j + \sum_{l=1, l \neq k}^{q_2} u_l (a_l)_i = (a_k)_i, \quad i = 1, 2, \dots, n. \quad (4.2.22)$$

Equations (4.2.18), (4.2.19) and (4.2.22) show that Z_j and u_l satisfy the dual feasibility condition (1) of Theorem 4.2.1.

By (4.2.1) and (4.2.16), and the orthogonality of Q_j , we get

$$\begin{aligned} A^{(j)}(x^*) \bullet Z_j &= [Q_j^T A^{(j)}(x^*) Q_j] \bullet [Q_j^T Z_j Q_j] \\ &= \text{diag}(\lambda_1^{(j)}, \lambda_2^{(j)}, \dots, \lambda_{r_j}^{(j)}, 0, \dots, 0) \bullet \text{diag}(0, 0, \dots, 0, \tilde{Q}_j^T W_j \tilde{Q}_j) \\ &= 0. \end{aligned} \quad (4.2.23)$$

Since $A^{(j)}(x^*) \succeq 0$, $Z_j \succeq 0$, from (4.2.23) we have

$$A^{(j)}(x^*) Z_j = 0 \text{ for } j = 1, 2, \dots, q_1. \quad (4.2.24)$$

By the definition of $\mathcal{A}_2$ and (4.2.17),

$$(a_l^T x^* + b_l)u_l = 0, \quad l = 1, 2, \dots, k-1, k+1, \dots, q_2. \quad (4.2.25)$$

Equations (4.2.24) and (4.2.25) show that the complementary slackness condition (2) of Theorem 4.2.1 holds. $\square$

Figure 4.1: The feasible region of SDP12

We provide an example using **SDP12**:

SDP12:

$$A^{(1)}(x) = \begin{bmatrix} 5 & -1 \\ -1 & 2 \end{bmatrix} + x_1 \begin{bmatrix} -1 & 2 \\ 2 & 3 \end{bmatrix} + x_2 \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \succeq 0$$

$$A^{(2)}(x) = \begin{bmatrix} 5.2 & 1.4 \\ 1.4 & 1 \end{bmatrix} + x_1 \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \succeq 0$$

$$A^{(3)}(x) = \begin{bmatrix} 52/9 & 0 & 0 \\ 0 & 44/7 & 0 \\ 0 & 0 & 44 \end{bmatrix} + x_1 \begin{bmatrix} -8/9 & 0 & 0 \\ 0 & -4/7 & 0 \\ 0 & 0 & -4 \end{bmatrix} + x_2 \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -7 \end{bmatrix} \succeq 0.$$

The feasible region of $SDP12$ is given in Figure 4.1. The third constraint is diagonalizable into three linear constraints. We want to determine whether or not each of the three linear constraints is redundant. We have three programs to solve, namely: $SDP12_1$, $SDP12_2$, and $SDP12_3$. Consider $SDP12_1$. Recall that $SDP12_1$ is the problem of determining whether or not $52/9 - 8/9x_1 - x_2 \geq 0$ is redundant in the feasible region of $SDP12$.

$SDP12_1$:

$$\begin{aligned} \min \quad & -8/9x_1 - x_2 \\ \text{st.} \quad & A^{(1)}(x) \succeq 0 \\ & A^{(2)}(x) \succeq 0 \\ & 44/7 - 4/7x_1 - x_2 \geq 0 \\ & 44 - 4x_1 - 7x_2 \geq 0 \end{aligned}$$

Using SDPSOL, $SDP12_1$ has an optimal solution $x^*=(2.80,0.1833)$, with objective value -2.67221. Since $-2.67221 + 52/9 = 3.1056 > 0$, then $52/9 - 8/9x_1 - x_2 \geq 0$ is redundant in $\mathcal{R}$ by Theorem 4.1.1. We can use equations (4.2.9) of Theorem 4.2.2 to check if any of $SDP12_2$ and $SDP12_3$ is also optimal at x^* . For example, consider

$SDP12_2$:

$$\begin{aligned}
\min \quad & -4/7x_1 - x_2 \\
st. \quad & A^{(1)}(x) \succeq 0 \\
& A^{(2)}(x) \succeq 0 \\
& 52/9 - 8/9x_1 - x_2 \geq 0 \\
& 44 - 4x_1 - 7x_2 \geq 0
\end{aligned}$$

We obtain

$$\begin{aligned}
A^{(1)}(x^*) &= \begin{bmatrix} 5 & -1 \\ -1 & 2 \end{bmatrix} + x_1^* \begin{bmatrix} -1 & 2 \\ 2 & 3 \end{bmatrix} + x_2^* \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 2.2000 & 4.7833 \\ 4.7833 & 10.4000 \end{bmatrix} \\
&= \begin{bmatrix} 0.4179 & 0.9085 \\ 0.9085 & -0.4179 \end{bmatrix} \begin{bmatrix} 12.6000 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.4179 & 0.9085 \\ 0.9085 & -0.4179 \end{bmatrix}^T \\
A^{(2)}(x^*) &= \begin{bmatrix} 5.2 & 1.4 \\ 1.4 & 1 \end{bmatrix} + x_1^* \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} + x_2^* \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \\
&= \begin{bmatrix} 2.4000 & 1.4000 \\ 1.4000 & 0.8167 \end{bmatrix} \\
&= \begin{bmatrix} -0.8638 & 0.5039 \\ -0.5039 & -0.8638 \end{bmatrix} \begin{bmatrix} 3.2167 & 0 \\ 0 & 0.0000 \end{bmatrix} \begin{bmatrix} -0.8638 & 0.5039 \\ -0.5039 & -0.8638 \end{bmatrix}^T
\end{aligned}$$

We have $\mathcal{A}_1 = \{1, 2\}$. The two linear constraints, at x^* , give

$$52/9 - 8/9x_1^* - x_2^* = 3.1056 \quad 44 - 4x_1^* - 7x_2^* = 31.5168.$$

So, they are not binding at x^* , and thus $\mathcal{A}_2 = \emptyset$. We have

$$\tilde{Q}_1 = [0.9085, -0.4179]^T, \quad \tilde{Q}_2 = [0.5039, -0.8638]^T.$$

The linear system (4.2.9) corresponding to $SDP12_2$ at $x^*=(2.80, 0.1833)$ is

$$\begin{aligned} -1.8201w^{(1)} - 0.2539w^{(2)} &= -4/7 \\ -0.7593w^{(1)} - 0.7462w^{(2)} &= -1 \end{aligned}$$

The system has the unique solution $w^{(1)} = 0.1480$ and $w^{(2)} = 1.1895$. Both are nonnegative. Hence, by Theorem 4.2.3, $x^*=(2.8000, 0.1833)$ is an optimal solution for SDP_2^3 . Because $44/7 - 4/7x_1^* - x_2^* = 4.5024 > 0$, then $44/7 - 4/7x_1 - x_2 \geq 0$ is redundant in $\mathcal{R}$.

The constraints (4.1.5) contain the *hidden* linear constraints given by their diagonal elements. For, instance, $A_0^{(j)} + \sum_{i=1}^n x_i A_i^{(j)} \succeq 0$ has the linear constraints

$$(A_0)_{kk}^{(j)} + \sum_{i=1}^n x_i (A_i)_{kk}^{(j)} \geq 0, \quad k = 1, 2, \dots, m_j$$

where $(A_0)_{kk}^{(j)}$ means the k th diagonal element of $(A_0)^{(j)}$, etc. A less expensive way to partially detect redundant linear constraints $a_l^T x + b_l \geq 0$ from the region defined by (4.1.5) and (4.1.6) is to replace (4.1.5) by their diagonal linear constraints. In that case, we can apply known linear methods to tackle the problem. But, removal of redundant constraints must be restricted to the linear constraints (4.1.6), and for the diagonal constraints determined by (4.1.5).

4.3 Removing Duplicate Linear Constraints

Two linear constraints are called *duplicate* if they are identical except for a scalar multiplier [40]. It is suggested in [15] that all duplicate linear constraints can be removed in the first iteration of SH. We show how this idea works in the case of the system defined by (4.1.5) and (4.1.6). We show that all (redundant) duplicates in the linear system (4.1.6), i.e.,

$$a_l^T x + b_l \geq 0, \quad l = 1, 2, \dots, q_2$$

can be removed by the application of a single hitting step of the SSH algorithm.

Let x_0 be an interior point of the region $\mathcal{R}$ and let s a random direction. The probability that a random line from the interior hits a given linear constraint is 1. For each $l = 1, 2, \dots, q_2$, the distance from the interior point x_0 to the boundary of $a_l^T x + b_l \geq 0$ is

$$\sigma_l = \frac{a_l^T x_0 + b_l}{a_l^T s}.$$

Algorithm

0. Set $\mathcal{I} = \{1, 2, \dots, q_2\}$
1. Generate a random direction s uniformly on the unit hypersphere.
2. For any pair (l^1, l^2) , $1 \leq l^1 < l^2 \leq q_2$, calculate σ_{l^1} and σ_{l^2} . If $\sigma_{l^1} = \sigma_{l^2}$, delete σ_{l^2} from $\mathcal{I}$.

Remark:

For any pair (l^1, l^2) , $\sigma_{l^1} = \sigma_{l^2}$ if and only if the linear constraints l^1 and l^2 are duplicates of each other. If $\sigma_{l^1} = \sigma_{l^2}$, the distances from x_0 to the boundaries of constraints l^1 and l^2 are equal. The probability of hitting two different linear constraints in the s direction from x_0 , is zero. Hence, we conclude that constraints l^1 and l^2 are duplicates of each other.

At the end of the algorithm, the remaining constraints are free of duplicates. Hence, the (redundant) duplicate constraints will have been removed. Note that due to finite precision of computers, the equality $\sigma_{l^1} = \sigma_{l^2}$ is correct only to a finite number of significant digits.

The method of detecting duplicate constraints discussed in [40] treats the constraints pair-wise, giving a large number of cases to be considered. This indicates that the algorithm presented here is more efficient.

Example: Consider the linear constraints corresponding to constraint $A^{(3)}(x) \succeq 0$ of SDP12:

$$52/9 - 8/9x_1 - x_2 \geq 0 \quad (1)$$

$$44/7 - 4/7x_1 - x_2 \geq 0 \quad (2)$$

$$44 - 4x_1 - 7x_2 \geq 0. \quad (3)$$

We generate a random direction $s = (0.8806, 0.4738)$ and choose interior point of SDP12, $x_0 = (2, -4)$. The distances from $x_0 = (2, -4)$ to the boundaries of the

constraints 1, 2, 3 above, are respectively $\{-140.8451, 48.3, 48.3\}$. This means constraints 2 and 3 are duplicates, and constraint 3 can be removed. By using a single random direction we have removed all the duplicates.

4.4 $S1$ -redundant Convex Quadratic Constraints

In this section we introduce $S1$ -redundant convex quadratic constraints (CQC) and give a method based on the S -procedure to detect them in a system of convex quadratic constraints. In the special case of ellipsoids, the method is improved significantly by solving a $MAXDET$ problem.

Let $F_j : x^T A_j x + 2b_j^T x + c_j \leq 0$, ($1 \leq j \leq q$) be a system of quadratic inequalities, where $x \in \mathbb{R}^n$.

Definition 4.4.1 *The k th constraint F_k is called $S1$ -redundant if for some $j \neq k$*

$$\{x \in \mathbb{R}^n \mid x^T A_j x + 2b_j^T x + c_j \leq 0\} \subseteq \{x \in \mathbb{R}^n \mid x^T A_k x + 2b_k^T x + c_k \leq 0\}. \quad (4.4.1)$$

When (4.4.1) holds, we say that F_k is $S1$ -redundant with respect to F_j . A quadratic inequality F_k is redundant if it is $S1$ -redundant.

Not every quadratic constraint is a positive semidefinite constraint. The feasible region of a semidefinite constraint is convex, but a quadratic constraint $x^T A_j x + 2b_j^T x + c_j \leq 0$ determines a convex region if and only if $A_j \succeq 0$. Every convex quadratic constraint is a semidefinite constraint. To show this, we begin by introducing the well-known Schur Lemma. The proof of the Schur Lemma is straightforward and well-known [23].

Lemma 4.4.1 (Schur Lemma). *Let Q be a real symmetric matrix of the form*

$$Q = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix},$$

and $A \succ 0$. Define $S = C - B^T A^{-1} B$. Then the following hold:

1. $Q \succ 0$ iff $S \succ 0$
2. $Q \succeq 0$ iff $S \succeq 0$. $\square$

S is called the *Schur complement of matrix Q* . It is easily shown that if we are given $C \succ 0$ instead of $A \succ 0$ and define $\bar{S} = A - B C^{-1} B^T$, then we can write

1. $Q \succ 0$ iff $\bar{S} \succ 0$
2. $Q \succeq 0$ iff $\bar{S} \succeq 0$.

Let $x^T A_j x + 2b_j^T x + c_j \leq 0$ be a convex quadratic constraint (CQC), i.e., $A_j \succeq 0$. Since $A_j \succeq 0$, $A_j^{\frac{1}{2}}$ exists. Using the Schur Lemma, we have

$$\begin{aligned} x^T A_j x + 2b_j^T x + c_j \leq 0 &\iff -2b_j^T x - c_j - x^T A_j x \geq 0 \\ &\iff \beta_j(x) = \begin{bmatrix} I & A_j^{\frac{1}{2}} x \\ x^T A_j^{\frac{1}{2}} & -2b_j^T x - c_j \end{bmatrix} \succeq 0. \end{aligned} \quad (4.4.2)$$

Equation (4.4.2) is a positive semidefinite constraint. Hence, $x^T A_j x + 2b_j^T x + c_j \leq 0$ is equivalent to the positive semidefinite constraint $\beta_j(x) \succeq 0$.

We are concerned with detecting $S1$ -redundant convex quadratic constraints (CQC) from the following system of CQC

$$F_j : x^T A_j x + 2b_j^T x + c_j \leq 0, \quad A_j \succeq 0 \quad (1 \leq j \leq q). \quad (4.4.3)$$

We describe the *S-procedure* for detecting CQC constraints.

Lemma 4.4.2 (S-Procedure) [20] *Consider the sets*

$$F_j = \{x \mid x^T A_j x + 2b_j^T x + c_j \leq 0\}, \quad j = 1, 2, \dots, q. \quad (4.4.4)$$

If there exists σ_j ($2 \leq j \leq q$) satisfying

$$\sum_{j=2}^q (\sigma_j \begin{bmatrix} A_j & b_j \\ b_j^T & c_j \end{bmatrix}) - \begin{bmatrix} A_1 & b_1 \\ b_1^T & c_1 \end{bmatrix} \succeq 0 \quad (4.4.5)$$

$$\sigma_j \geq 0, \quad j = 2, 3, \dots, q \quad (4.4.6)$$

then $\bigcap_{j=2}^q F_j \subseteq F_1$.

The condition is necessary and sufficient when $q = 2$ and F_2 has nonempty interior.

If $\bigcap_{j=2}^q F_j \subseteq F_1$ then F_1 is redundant. So, the *S-procedure* provides a condition for finding redundant quadratic constraints. The following corollary to Lemma 4.4.2 gives necessary and sufficient condition for determining whether or not F_k is *S1-redundant* in the system (4.4.3).

Corollary 4.4.1 *F_k is S1-redundant in $\mathcal{R}$ if and only if for some j*

$$\sigma \begin{bmatrix} A_j & b_j \\ b_j^T & c_j \end{bmatrix} - \begin{bmatrix} A_k & b_k \\ b_k^T & c_k \end{bmatrix} \succeq 0 \quad (4.4.7)$$

$$\sigma \geq 0 \quad (4.4.8)$$

is feasible.

Proof:

$$\begin{aligned}
 F_k \text{ is } S1\text{-redundant in } \mathcal{R} &\iff F_j \subseteq F_k \text{ for some } j \neq k \\
 &\iff \sigma \begin{bmatrix} A_j & b_j \\ b_j^T & c_j \end{bmatrix} - \begin{bmatrix} A_k & b_k \\ b_k^T & c_k \end{bmatrix} \succeq 0, \quad \sigma \geq 0 \\
 &\quad \text{is feasible}
 \end{aligned}$$

The last equivalence follows from the S -Procedure. $\square$

Using Corollary 4.4.1, we can determine all $S1$ -redundant convex quadratic constraints in (4.4.3) by solving 2^q SDP feasibility problems. This is too large in general. In the case of ellipsoids, this method can be improved considerably as follows.

Ellipsoids

We next consider the system of ellipsoids $\mathcal{E}_j$ defined by the form

$$\mathcal{E}_j = \{x \in \mathbb{R}^n \mid (x - x_j)^T H_j^{-2} (x - x_j) \leq 1\}, \quad H_j \succ 0, \quad j = 1, 2, \dots, q. \quad (4.4.9)$$

Corollary 4.4.2 $\mathcal{E}_k$ is $S1$ -redundant in the system (4.4.9) if and only if there is a j and a $\sigma \geq 0$ for which

$$\sigma \begin{bmatrix} -H_j^{-2} & -H_j^{-2} x_j \\ -x_j^T H_j^{-2} & x_j^T H_j^{-2} x_j - 1 \end{bmatrix} - \begin{bmatrix} -H_k^{-2} & -H_k^{-2} x_k \\ -x_k^T H_k^{-2} & x_k^T H_k^{-2} x_k - 1 \end{bmatrix} \succeq 0 \quad (4.4.10)$$

$$\sigma \geq 0 \quad (4.4.11)$$

Proof: Observe that each $\mathcal{E}_j$ can be written in the form

$$\mathcal{E}_j = \{x \in \mathbb{R}^n \mid x^T A_j x + 2b_j^T x + c_j \leq 0\},$$

where $A_j = H_j^{-2}$, $b_j = -H_j^{-2}x_j$, $c_j = x_j^T H_j^{-2}x_j - 1$. The result follows from Corollary 4.4.1. $\square$

Let $\mathcal{R}$ be the feasible region determined by the system (4.4.9). $\mathcal{R}$ is convex, and it is well-known that every convex body is contained in a unique ellipsoid of minimal volume called the Löwner ellipsoid [22]. Any constraint in (4.4.9) that is $S1$ -redundant with respect to the Löwner ellipsoid is redundant in $\mathcal{R}$. To check for this redundancy, there are q SDP feasibility problems to solve as given in Corollary 4.4.2.

Unfortunately, it is hard to compute Löwner ellipsoids [22]. This is an NP-hard problem even for polytopes [11]. In the case of convex regions determined by semidefinite constraints, there are two known polynomial-time methods for computing an approximation of the Löwner ellipsoid. We introduce the idea of a $S1$ -redundant ellipsoid as one that is $S1$ -redundant with respect to a given approximation of the Löwner ellipsoid .

The first method for computing a Löwner ellipsoid approximation of $\mathcal{R}$ uses its analytic center [11]. This method applies to any bounded semidefinite constrained region. The analytic center of a semidefinite constrained region can be computed in polynomial-time using the damped Newton method [30]. Unfortunately, the volume of the Löwner ellipsoid approximation obtained via the analytic center is proportional to the total number of redundant and noredundant constraints [11]. The more redundant constraints there are, the larger the volume of the approximation. This reduces the number of $S1$ -redundant constraints found. Hence, the approximation of

the Löwner ellipsoid via the analytic center is not ideal for detecting $S1$ -redundant ellipsoids, and so we reject this approach.

A second Löwner ellipsoid approximation whose volume depends *only* on the dimension of the space is obtained via the maximum volume ellipsoid $\mathcal{E}_0$ contained inside the feasible region $\mathcal{R}$.

$$\mathcal{E}_0 = \{x \in \mathbb{R}^n \mid (x - x_0)^T H_0^{-2} (x - x_0) \leq 1\} \quad (4.4.12)$$

The ellipsoid $\mathcal{E}_0$ is given by the solution of the MAXDET problem [11], which is polynomial-time.

$$\begin{aligned} \max \quad & -\log \det(H_0) \\ \text{s.t.} \quad & \begin{bmatrix} H_j^2 & x_0 - x_j & -H_0 \\ (x_0 - x_j)^T & 1 - \sigma_j & 0 \\ -H_0 & 0 & \sigma_j I \end{bmatrix} \succeq 0 \end{aligned} \quad (4.4.13)$$

$$H_0 \succ 0 \quad (4.4.14)$$

$$\sigma_j \geq 0, \quad j = 1, 2, \dots, q. \quad (4.4.15)$$

The ambient space of this problem is $\mathbb{R}^n$, i.e., $x \in \mathbb{R}^n$. Given $\mathcal{E}_0$, it is known [19] that:

$$\mathcal{R} \subseteq n(\mathcal{E}_0 - x_0) + x_0, \quad (4.4.16)$$

where $\mathcal{E}_0 - x_0$ is a translation of $\mathcal{E}_0$ so that its center is the origin. The ellipsoid $n(\mathcal{E}_0 - x_0) + x_0$ has the same center as $\mathcal{E}_0$ and its volume is n times that of $\mathcal{E}_0$. Call $n(\mathcal{E}_0 - x_0) + x_0$ the *killer* ellipsoid, because any ellipsoid containing it can be

removed from (4.4.9). For a *centrally symmetric* feasible region $\mathcal{R}$, the factor n can be improved to $\sqrt{n}$ [26], i.e.,

$$\mathcal{R} \subseteq \sqrt{n}(\mathcal{E}_0 - x_0) + x_0.$$

Since $\mathcal{E}_0$ is unique, it is clear that the volume of the approximation $n(\mathcal{E}_0 - x_0) + x_0$ to the Löwner ellipsoid does not depend on the number of constraints that defines $\mathcal{R}$. Therefore, for redundancy detection, this approximation is better than the one obtained via the analytic center of $\mathcal{R}$. The following is the (S1-RE) algorithm for detecting S1-redundant ellipsoids from the system of ellipsoids $\mathcal{E}_j$, $j = 1, 2, \dots, q_0$ of the form (4.4.9.)

The S1-redundant Ellipsoid (S1-RE) Algorithm.

Set $\mathcal{I} = \{1, 2, \dots, q\}$. Set $k = 1$.

Solve MAXDET problem (4.4.13) to obtain the maximum volume ellipsoid $\mathcal{E}_0$ contained in $\mathcal{R}$.

Repeat

Use Corollary 4.4.2 to determine $n(\mathcal{E}_0 - x_0) + x_0 \subseteq \mathcal{E}_k$.

If $n(\mathcal{E}_0 - x_0) + x_0 \subseteq \mathcal{E}_k$, then $\mathcal{E}_k$ is S1-redundant, and remove it from $\mathcal{I}$.

$k = k + 1$

Until $k = q + 1$

Figure 4.2: The feasible region of SDP13. The inner ellipsoid is $\mathcal{E}_0$, and the outer dotted ellipsoid is the killer ellipsoid.

Example: Consider the region defined by the given ellipsoids below. They have the form $\mathcal{E}_j = \{x \in \mathbb{R}^n \mid (x - x_j)^T H_j^{-2} (x - x_j) \leq 1\}$, $j = 1, 2, \dots, 7$ given below

SDP13:

$$\mathcal{E}_1 : [x - (0.4962, 0.4226)^T]^T \begin{bmatrix} 0.5073 & 0.1151 \\ 0.1151 & 0.4357 \end{bmatrix}^{-2} [x - (0.4962, 0.4226)^T] \leq 1$$

$$\mathcal{E}_2 : [x - (0.6000, 0.6000)^T]^T \begin{bmatrix} 1.5038 & -0.7865 \\ -0.7865 & 1.5629 \end{bmatrix}^{-2} [x - (0.6000, 0.6000)^T] \leq 1$$

$$\mathcal{E}_3 : [(x - (0.5000, 0.6000)^T)^T]^T \begin{bmatrix} 2.4019 & -1.6020 \\ -1.6020 & 3.7910 \end{bmatrix}^{-2} [x - (0.5000, 0.6000)^T] \leq 1$$

$$\begin{aligned}
\mathcal{E}_4 & : [x - (0.5000, 0.4000)^T]^T \begin{bmatrix} 2.3952 & 0.4059 \\ 0.4059 & 1.6181 \end{bmatrix}^{-2} [x - (0.5000, 0.4000)^T] \leq 1 \\
\mathcal{E}_5 & : [x - (0.4000, 0.4000)^T]^T \begin{bmatrix} 2.9621 & 0.6517 \\ 0.6517 & 1.6642 \end{bmatrix}^{-2} [x - (0.4000, 0.4000)^T] \leq 1 \\
\mathcal{E}_6 & : [x - (0.3000, 0.4000)^T]^T \begin{bmatrix} 2.5397 & -1.0714 \\ -1.0714 & 2.7436 \end{bmatrix}^{-2} [x - (0.3000, 0.4000)^T] \leq 1 \\
\mathcal{E}_7 & : [x - (0.3000, 0.6000)^T]^T \begin{bmatrix} 2.7483 & -1.4789 \\ -1.4789 & 2.7715 \end{bmatrix}^{-2} [x - (0.3000, 0.6000)^T] \leq 1.
\end{aligned}$$

The optimal solution of MAXDET (4.4.1) corresponding to this example is

$$\begin{aligned}
H_0 & = \begin{bmatrix} 0.4929 & 0.1055 \\ 0.1055 & 0.4250 \end{bmatrix} \\
x_0 & = (0.5102, 0.4339)^T.
\end{aligned}$$

Since we are in $\mathbb{R}^2$, volume means area. Thus, the maximum area ellipsoid inside the feasible region of the example SDP13 is:

$$\mathcal{E}_0 : [x - (0.5102, 0.4339)^T]^T \begin{bmatrix} 0.4929 & 0.1055 \\ 0.1055 & 0.4250 \end{bmatrix}^{-2} [x - (0.5102, 0.4339)^T] \leq 1.$$

The area of the optimal ellipsoid $\mathcal{E}_0$ is directly proportional to $\sqrt{\det(H_0^2)}$ [22]. The Löwner ellipsoid approximation (4.4.16), i.e., the *killer* ellipsoid, is

$$\begin{aligned}
2(\mathcal{E}_0 - x_0) + x_0 & = \{x \mid (x - x_0)^T (2H_0)^{-2} (x - x_0) \leq 1\} \\
& = \{x \mid [x - \begin{pmatrix} 0.5102 \\ 0.4339 \end{pmatrix}]^T \begin{bmatrix} 0.9858 & 0.2110 \\ 0.2110 & 0.8500 \end{bmatrix}^{-2} [x - \begin{pmatrix} 0.5102 \\ 0.4339 \end{pmatrix}] \leq 1\}.
\end{aligned}$$

Figure 4.2 shows the feasible region $SDP13$, the maximum volume ellipsoid $\mathcal{E}_0$, and the *killer* ellipsoid $2(\mathcal{E}_0 - x_0) + x_0$ of the feasible region. We use Corollary 4.4.2 to solve 7 SDP feasibility problems to determine whether or not

$$2(\mathcal{E}_0 - x_0) + x_0 \subseteq \mathcal{E}_j, \quad j = 1, 2, \dots, 7.$$

We find that constraints 3,4,5,6,7 to be $S1$ -redundant, and hence redundant. For instance, $2(\mathcal{E}_0 - x_0) + x_0 \subseteq \mathcal{E}_3$ holds with $\sigma = 0.9047$ in equations (4.4.10) and (4.4.11).

Chapter 5

Applications to the Löwner Ellipsoid Problems and Other Problems

In this chapter, we show how the SSH method can be applied to the Löwner ellipsoid problem for a set of points. Theorem 5.1.2 shows that all the points that lie on the boundary of the Löwner ellipsoid can be identified using SSH. In the last section, we apply SSH to improve the complexity of the $S1$ -redundant ellipsoid method of Chapter 4.

5.1 Löwner Ellipsoid for $\{a^j\}_{j=1}^q$

Let $\{a^j\}_{j=1}^q$ be a set of distinct data points in $\mathbb{R}^n$. The minimum volume ellipsoid containing these data points is called the *Löwner ellipsoid for $\{a^j\}_{j=1}^q$* [22]. Given $X \succ 0$ and $y \in \mathbb{R}^n$, define the ellipsoid

$$\mathcal{E}(X, y) = \{a \in \mathbb{R}^n \mid (Xa + y)^T(Xa + y) \leq 1\}. \quad (5.1.1)$$

By Corollary 2.2.1, we have

$$(Xa + y)^T(Xa + y) \leq 1 \iff \begin{bmatrix} I & Xa + y \\ (Xa + y)^T & 1 \end{bmatrix} \succeq 0, \quad (1 \leq j \leq q) \quad (5.1.2)$$

The volume of the ellipsoid $\mathcal{E}$ is proportional to $\det(X^{-1})$. Since $\log(x)$ is an increasing function, minimizing $\det(X^{-1})$ is equivalent to minimizing $\log \det(X^{-1}) = -\log \det(X)$. By (5.1.1) and (5.1.2), the minimum volume ellipsoid containing $\{a^j\}_{j=1}^q$, i.e., the Löwner ellipsoid for $\{a^j\}_{j=1}^q$, is given by the solution of the MAXDET problem

$$\begin{aligned} \max \quad & \log \det(X) \\ & A^{(j)}(X, y) = \begin{bmatrix} I & Xa^j + y \\ (Xa^j + y)^T & 1 \end{bmatrix} \succeq 0, \quad (1 \leq j \leq q) \quad (5.1.3) \\ & X \succ 0. \quad (5.1.4) \end{aligned}$$

where the variable X denotes a $n \times n$ symmetric matrix and the variable y belongs to $\mathbb{R}^n$. The dimension of the ambient space of the problem is $n(n+3)/2$.

Let (X_0, y_0) be the optimal solution of this problem. The Löwner ellipsoid $\mathcal{E}_0$ is given by:

$$\mathcal{E}_0 = \{x \in \mathbb{R}^n \mid (X_0 x + y_0)^T (X_0 x + y_0) \leq 1\}. \quad (5.1.5)$$

Let $\mathcal{R}$ be the feasible region of the MAXDET problem described by (5.1.3) and (5.1.4). This is a semidefinite constrained region. We are interested in determining the nonredundant constraints of $\mathcal{R}$. The SSH method can be applied to detect the nonredundant constraints. In order to apply SSH to remove the redundancies in

(5.1.3) and (5.1.4), we need to establish the boundedness of the region $\mathcal{R}$. The following theorem addresses this.

Theorem 5.1.1 *The feasible region $\mathcal{R}$ given by (5.1.3) and (5.1.4) is bounded if a subset of $n + 1$ of the data points $\{a^j\}_{j=1}^q$ is an affinely independent set.*

Proof: Suppose $\mathcal{R}$ is unbounded. By Lemma 3.1.3 with $A^{(j)}(X, y)$ given by (5.1.3), a necessary condition for the unboundedness of $\mathcal{R}$ is the existence of a $n \times n$ matrix S and a vector $s \in \mathbb{R}^n$, not both zero, such that

$$\begin{bmatrix} 0 & Sa^j + s \\ (Sa^j + s)^T & 0 \end{bmatrix} \succeq 0, \quad j = 1, 2, \dots, q \quad (5.1.6)$$

$$S \succeq 0. \quad (5.1.7)$$

Now, the q equations (5.1.6) imply that

$$\begin{bmatrix} I & Sa^j + s \\ (Sa^j + s)^T & 0 \end{bmatrix} \succeq 0, \quad j = 1, 2, \dots, q$$

By applying the Schur Lemma 4.4.1, we obtain

$$-(Sa^j + s)^T(Sa^j + s) \geq 0 \quad j = 1, 2, \dots, q.$$

Hence,

$$Sa^j + s = 0, \quad j = 1, 2, \dots, q. \quad (5.1.8)$$

$$S(a^i - a^j) = 0, \quad i, j = 1, 2, \dots, q. \quad (5.1.9)$$

By the assumption of the affine independence of $\{a^j\}_{j=1}^q$, n of the vectors $a^i - a^j$ are linearly independent and form a basis of $\mathbb{R}^n$. The equation (5.1.9) yields $S=0$.

By (5.1.8) it follows that $s=0$. This is a contradiction of the assumption that S , s are not both zero. Thus, the region $\mathcal{R}$ is bounded. $\square$

Our experiments indicate that, for q large, many of the constraints in (5.1.3) are redundant. The following theorem shows that the constraints in (5.1.3) given by the points a^j on the boundary of the Löwner ellipsoid are nonredundant.

Theorem 5.1.2 *If a^k is a point that lies on the boundary of the Löwner ellipsoid $\mathcal{E}_0$, then the k th constraint of (5.1.3) is nonredundant.*

Proof: By definition, the k th constraint is redundant if and only if for all ellipsoids $\mathcal{E}$ (5.1.1) we have $(a^j \in \mathcal{E} \text{ for } j = 1, 2, \dots, k-1, k+1, \dots, q \implies a^k \in \mathcal{E})$. Let

$$\hat{a}_k = \{a^1, a^2, \dots, a^{k-1}, a^{k+1}, \dots, a^q\}.$$

On the surface of the Löwner ellipsoid $\mathcal{E}_0$, choose a small closed ball centered at a^k that does not intersect the convex hull of $\hat{a}_k$. Then, the hyperplane passing through the boundary of this ball separates a^k from the remaining points $\hat{a}_k$. We can then find an ellipsoid containing the remaining points $\hat{a}_k$, but not containing a^k . Hence, the k th constraint of (5.1.3) is nonredundant. $\square$

Example: We use the points in Figure 5.1 and apply SSH to the feasible region $\mathcal{R}$ determined by these points. Figure 5.2 gives the points that corresponds to the constraints of (5.1.3) that are detected. From Figure 5.2, we see that most of the interior points of the Löwner ellipsoid $\mathcal{E}_0$ give rise to redundant constraints in (5.1.3). We also see that some of the interior points give nonredundant constraints.

Figure 5.1: A set of 1000 points in $\mathbb{R}^2$ and their Löwner ellipsoid

Figure 5.2: The detected nonredundant points of Figure 5.1, and their Löwner ellipsoid.

Since the constraints that define $\mathcal{R}$ are ellipsoids, the hitting step of the SSH algorithm can be implemented more efficiently. Let (X_0, y_0) be the hitting point, and let (S, s) be the search direction of an SSH iteration. Recall that the hitting step involves the computation of the distances from (X_0, y_0) to the boundary of

$$\begin{bmatrix} I & Xa^j + y \\ (Xa^j + y)^T & 1 \end{bmatrix} \succeq 0 \quad (5.1.10)$$

in the directions (S, s) and $-(S, s)$ for each constraint. From Theorem 2.4.1 and the SSH algorithm, these distances in the directions (S, s) and $-(S, s)$, are given respectively by

$$d_1 = 1/\lambda_{\max}(B_j) \quad (5.1.11)$$

$$d_2 = 1/\lambda_{\min}(B_j), \quad (5.1.12)$$

where

$$v^j = X_0 a^j + y_0 \quad (5.1.13)$$

$$w^j = S a^j + s \quad (5.1.14)$$

$$B_j = - \begin{bmatrix} I & v^j \\ (v^j)^T & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & w^j \\ (w^j)^T & 0 \end{bmatrix} \quad (5.1.15)$$

$$= \frac{1}{(v^j)^T v^j - 1} \begin{bmatrix} -v^j (w^j)^T & [(-v^j)^T v^j + 1]I + v^j (v^j)^T \\ (w^j)^T & -(v^j)^T w^j \end{bmatrix} \quad (5.1.16)$$

Note that $\|v^j\|_2 < 1$ since (X_0, y_0) is an interior point.

The following proposition gives an efficient way of computing the distances.

Proposition 5.1.1 (a) *The distances in the direction (S, s) of (X_0, y_0) to the boundary of the j th constraint (5.1.10) are given by*

$$d_1 = \frac{-(v^j)^T w^j + \sqrt{((v^j)^T w^j)^2 - \|w^j\|_2^2 (\|v^j\|_2^2 - 1)}}{\|w^j\|_2^2} \quad (5.1.17)$$

$$d_2 = \frac{-(v^j)^T w^j - \sqrt{((v^j)^T w^j)^2 - \|w^j\|_2^2 (\|v^j\|_2^2 - 1)}}{\|w^j\|_2^2} \quad (5.1.18)$$

(b) *Let d_1 and d_2 be defined by (5.1.17) and (5.1.18). For any $v_j, w_j \in \mathbb{R}^n$ with $w_j \neq 0$ and $\|v^j\|_2 < 1$ the maximum and the minimum eigenvalues of the matrix B_j (5.1.16) are given by $1/d_1$ and $1/d_2$ respectively.*

Proof:(a) By (5.1.2), the distances are given by the two solutions of the system

$$[X_0 a^j + y_0 + d(Sa^j + s)]^T [X_0 a^j + y_0 + d(Sa^j + s)] = 1, \quad d \in \mathbb{R}.$$

Solving this system and using the substitution (5.1.13) and (5.1.14), gives (5.1.17) and (5.1.18).

(b) In the result of (5.1.11), (5.1.12) and part (a), the only important restriction on w^j and v^j is that $w_j \neq 0$ and $\|v^j\|_2 < 1$. In general, w_j and v^j can be any vectors in $\mathbb{R}^n$ satisfying $w_j \neq 0$ and $\|v^j\|_2 < 1$. Therefore, (b) follows from part (a), (5.1.11) and (5.1.12). $\square$

Proposition 5.1.1 gives a more efficient way of implementing the SSH hitting step. Since the hitting point is fixed, so is v^j . Computing w^j requires n^2 multiplications. From (5.1.14), (5.1.17) and (5.1.18), computing the two distances requires $n^2 + 4n$ multiplications, 2 divisions and 1 square root. Computing each SSH iteration is at most $q(n^2 + 4n + 3)$ flops.

The problem of detecting the nonredundant constraints of the system (5.1.3) and (5.1.4) has significant potential in finding outliers in statistics, optimal design, and cluster analysis. In statistical problem, the points on the boundary of the Löwner ellipsoid are possible outliers.

The dual of the *D-experimental design* problem is the problem is of determining the minimum volume ellipsoid centered at the origin $\{x \mid x^t \hat{W} x \leq p\}$ containing a set of points $\{v^j\}_{j=1}^q$ [43]. That is, the problem

$$\begin{aligned} \max \quad & \log \det(W) \\ & (v^j)^T W v^j \leq p \quad j = 1, 2, \dots, q \end{aligned} \quad (5.1.19)$$

$$W \succ 0 \quad (5.1.20)$$

In this problem, the useful points v^j are those that lie on the boundary of the ellipsoid $\{x \mid x^t \hat{W} x \leq p\}$ [43]. These important points can be found without having to solve the dual problem, by applying SSH to the dual feasible region. Recall that by Theorem 5.1.2, any point v^j that lies on the boundary of the ellipsoid $\{x \mid x^t \hat{W} x \leq p\}$ gives rise to a nonredundant constraint (4.1.19).

5.2 Applying SSH to improve the *S1-RE* method

The *S1-RE* method for removing *S1*-redundant ellipsoids described in Section 4 of Chapter 4 can be improved by the application of the SSH method. Consider, the system of ellipsoids (4.4.9). First, we show that $\mathcal{E}_j$ can be expressed as a semidefinite

constraint. The ellipsoid $\mathcal{E}_j$ can be written as

$$\mathcal{E}_j = \{x \in \mathbb{R}^n \mid x^T A_j x + 2b_j^T x + c_j \leq 0\},$$

$$\begin{aligned} \text{where} \quad A_j &= H_j^{-2} \\ b_j &= -H_j^{-2} x_j \\ c_j &= x_j^T H_j^{-2} x_j - 1 \end{aligned}$$

By (4.4.2), $\mathcal{E}_j$ is equivalent to the semidefinite constraint $\beta_j \succeq 0$, where

$$\beta_j(x) = \begin{bmatrix} I & A_j^{\frac{1}{2}} x \\ x^T A_j^{\frac{1}{2}} & -2b_j^T x - c_j \end{bmatrix} \succeq 0.$$

We can apply the SSH algorithm to detect the nonredundant constraints. The feasible region $\mathcal{R}$ determined by (4.4.9) is suitable for the application of SSH because it is bounded. When all the ellipsoids have the same center, by symmetry, the center of the region is an optimal hitting point for implementing SSH. The hitting step can be done here more efficiently as in the Löwner ellipsoid problem of the previous section.

Recall that given an interior point x_0 and a direction s , the SSH hitting step computes the distance from x_0 to the boundary of $\mathcal{E}_j : x^T A_j x + 2b_j^T x + c_j \leq 0$ in the direction $s, -s$. These distances are given by the solution of

$$(x + ds)^T A_j (x + ds) + 2b_j^T (x + ds) + c_j = 0, \quad d \in \mathbb{R}.$$

The distances are given by

$$d_1 = \frac{v_j + \sqrt{w_j^2 - 4vz}}{2z_j} \tag{5.2.1}$$

$$d_2 = \frac{v_j - \sqrt{w_j^2 - 4vz}}{2z_j} \quad (5.2.2)$$

where

$$v_j = s^T A_j s \quad (5.2.3)$$

$$w_j = 2(s^T A_j x_0 + b_j^T s) \quad (5.2.4)$$

$$z_j = x_0^T A_j x_0 + 2b_j^T x_0 + c_j \quad (5.2.5)$$

Because x_0 is an interior point, $z_j < 0$. Calculating v_j , w_j and z_j requires $2n^2 + 5n + 2$ multiplications. Computing d_1 and d_2 takes $2n^2 + 5n + 6$ multiplications, 2 divisions and one square root. Therefore, each SSH iteration requires at most $q(2n^2 + 5n + 9)$ flops.

Now, we refer to the method of removing S1-redundant constraints given in Section 4 of Chapter 4. Recall that the *killer* ellipsoid $n(\mathcal{E}_0 - x_0) + x_0$ of (4.4.9) is an approximation of the Löwner ellipsoid of $\mathcal{R}$, and it is an expansion of the maximum volume ellipsoid $\mathcal{E}_0$ contained in $\mathcal{R}$.

To find the maximum volume ellipsoid $\mathcal{E}_0$ contained in $\mathcal{R}$, only the nonredundant constraints are needed. Let $\mathcal{R}_{detec}$ be the feasible region determined by the constraints detected after the application of the SSH method to (4.4.9). Clearly $\mathcal{R} \subseteq \mathcal{R}_{detec}$.

Since the *killer* ellipsoid $n(\mathcal{E}_0 - x_0) + x_0$ is only an approximation of the Löwner ellipsoid of $\mathcal{R}$, we can disregard the hard to detect nonredundant constraints in

finding $\mathcal{E}_0$. Therefore, it is reasonable to approximate the *killer* ellipsoid with $n(\mathcal{E}_{detec} - x_0) + x_0$, where $\mathcal{E}_{detec}$ is the Löwner ellipsoid of $\mathcal{R}_{detec}$. There are two benefits of this approximation. Let q_0 be the number of nonredundant constraints detected after the application of SSH to region $\mathcal{R}$. First, the $2q$ constraints (4.4.13) and (4.4.15) of the *MAXDET* problem reduce to $2q_0$. This greatly reduces the work per iteration to solve it if $q \gg q_0$. Secondly, in using Corollary 4.4.1 to detect redundancy, we only have to check

$$n(\mathcal{E}_{detec} - x_0) + x_0 \subseteq \mathcal{E}_j$$

for the $q - q_0$ undetected constraints j . The q_0 constraints already detected are nonredundant. This is a benefit if the difference $q - q_0$ is small.

As an example consider SDP13 given in Section 4 of Chapter 4. The SSH algorithm detected ellipsoids 1 and 2, i.e., $detec = \{1, 2\}$. Using only ellipsoids 1 and 2 in (4.4.13) and (4.4.15) the *MAXDET* problem is solved to get the optimal solution

$$H_0 = \begin{bmatrix} 0.4929 & 0.1055 \\ 0.1055 & 0.4250 \end{bmatrix}$$

$$x_0 = (0.5102, 0.4339)^T.$$

With only two ellipsoids, we obtain the same maximum volume inner ellipsoid $\mathcal{E}_0$ as before, i.e.,

$$\mathcal{E}_0 : [x - (0.5102, 0.4339)^T]^T \begin{bmatrix} 0.4929 & 0.1055 \\ 0.1055 & 0.4250 \end{bmatrix}^{-2} [x - (0.5102, 0.4339)^T] \leq 1.$$

We use Corollary 4.4.2 to solve 5 (instead of 7) SDP feasibility problems to determine

whether or not

$$2(\mathcal{E}_0 - x_0) + x_0 \subseteq \mathcal{E}_j, \quad j = 3, 4, 5, 6, 7.$$

We already know that ellipsoids 1 and 2 are nonredundant.

Chapter 6

Conclusion

We have shown that the problem of checking whether or not a semidefinite constraint is redundant is NP-hard. Our work demonstrated that the SSH method is useful for detecting nonredundant semidefinite constraints. We found that the analytic center is a useful hitting point for implementing the SSH method. The use of the Undetected-First Rule (UFR) reduced the number of computations in the hitting step and the savings rose with increasing number of nonredundant constraints (Table 3.3). From our experiments, we noted that SSH implemented with UFR outperformed both SCD and SHD (Table 3.1 and Table 3.3). A problem requiring further research is the development of new stopping rules for SSH.

The performance of SSH depends on the choice of the hitting point. A way to improve SSH is to find a way of moving the analytic center towards an optimal hitting point. It can be done by repeatedly adding weights to the constraints proportionally to their detection frequencies and finding the weighted analytic center. This particular technique is expensive and it does not always work. In section 2 of

chapter 3, we described the *co-Shannon point* as an approximation of an optimal hitting point. It was obtained by maximizing the function

$$\tilde{H}(x) = - \sum_{j=1}^q \frac{1}{p_j(x)} \log\left(\frac{1}{p_j(x)}\right).$$

Further research is needed to study how the *co-Shannon point* is related to an optimal hitting point.

The *S1 – RE* method introduced in chapter 4 for detecting *S1*-redundant ellipsoids might be improved by finding a better approximation of the Löwner ellipsoid to get a smaller *killer* ellipsoid. The smaller the *killer* ellipsoid, the more constraints that become *S1*-redundant. As discussed in Section 5.2, SSH can be applied to further improve the method.

Appendix A

A Penalty function for low rank matrices

Here, we give a penalty function $V(x)$ for low rank matrices, for which we need the following two Lemmas.

Lemma A.0.1 (*Wolkowicz and Styan [44]*) *Given a symmetric $m \times m$ matrix A and let*

$$p = \frac{(\text{Tr}(A))^2}{\text{Tr}(A^2)} - m + 1.$$

Then, $A \succ 0$ if $\text{Tr}(A) > 0$ and $p > 0$.

In the next lemma we extend this result to positive semidefinite matrices. We gives a simple criterion in the form of a necessary and sufficient condition for determining whether or not a rank two symmetric matrix is positive semidefinite.

Lemma A.0.2 *Suppose $A \neq 0$ and let*

$$w = \frac{(\text{Tr}(A))^2}{\text{Tr}(A^2)} - \text{rank}(A) + 1.$$

If $\text{Tr}(A) > 0$ and $w > 0$, then $A \succeq 0$. Furthermore, if in addition $\text{rank}(A) \leq 2$, then $A \succeq 0$ if and only if $\text{Tr}(A) > 0$ and $w > 0$.

Proof: Let $A = Q \operatorname{diag}(\lambda_1, \lambda_2, \dots, \lambda_r, 0, \dots, 0)Q^T$ be a *spectral decomposition* of A , where $\{\lambda_i\}_{i=1}^r$ are the nonzero eigenvalues of A , and Q is orthogonal. Also, let $D = \operatorname{diag}(\lambda_1, \lambda_2, \dots, \lambda_r)$. Then clearly, $\operatorname{Tr}(A) = \operatorname{Tr}(D)$, $\operatorname{Tr}(A^2) = \operatorname{Tr}(D^2)$, $\operatorname{rank}(A) = \operatorname{rank}(D) = r$, and $A \succeq 0 \iff D \succ 0$. Therefore,

$$\begin{aligned} \operatorname{Tr}(A) > 0, \quad \frac{(\operatorname{Tr}(A))^2}{\operatorname{Tr}(A^2)} - \operatorname{rank}(A) + 1 > 0 &\implies \operatorname{Tr}(D) > 0, \quad \frac{(\operatorname{Tr}(D))^2}{\operatorname{Tr}(D^2)} - r + 1 > 0 \\ &\implies D \succ 0 \text{ (Lemma (A.0.1))} \\ &\implies A \succeq 0. \end{aligned}$$

Now, suppose $r = \operatorname{rank}(A) \leq 2$ and $A \succeq 0$. We have $\operatorname{Tr}(A) > 0$, since $A \neq 0$; and

$$\begin{aligned} \frac{(\operatorname{Tr}(A))^2}{\operatorname{Tr}(A^2)} - \operatorname{rank}(A) + 1 &= \frac{(\operatorname{Tr}(D))^2}{\operatorname{Tr}(D^2)} - r + 1 \\ &\geq \frac{(\operatorname{Tr}(D))^2}{\operatorname{Tr}(D^2)} - 2 + 1 \text{ (since we have } r \leq 2) \\ &> 0. \end{aligned}$$

For the case $r = 2$, we have used the fact that λ_1 and λ_2 are strictly positive. $\square$

Theorem A.0.1 Suppose that $r_j = \operatorname{rank}(A^{(j)}(x)) = 2$ for all $x \in \mathfrak{R}^n$ ($1 \leq j \leq q$). Then

$$V(x) := \sum_{j=1}^q \max\left\{0, -\frac{[\operatorname{Tr}(A^{(j)}(x))]^2}{\operatorname{Tr}([A^{(j)}(x)]^2)} + r_j - 1\right\} + \max\{0, -\operatorname{Tr}(A^{(j)}(x))\}$$

is a *penalty function* for the feasible region $\mathcal{R}$ of SDP (2.3.1).

Proof: If $x \in \mathcal{R}$, then by Lemma A.0.2

$$-\frac{[\operatorname{Tr}(A^{(j)}(x))]^2}{\operatorname{Tr}([A^{(j)}(x)]^2)} + r_j - 1 < 0 \text{ and } -\operatorname{Tr}(A^{(j)}(x)) < 0.$$

If $x \notin \mathcal{R}$, then by Lemma A.0.2

$$-\frac{[\text{Tr}(A^{(j)}(x))]^2}{\text{Tr}([A^{(j)}(x)]^2)} + r_j - 1 \geq 0 \text{ or } -\text{Tr}(A^{(j)}(x)) \geq 0 \text{ for some } j.$$

This together with the fact that $r_j = 2$ imply that

$$-\frac{[\text{Tr}(A^{(j)}(x))]^2}{\text{Tr}([A^{(j)}(x)]^2)} + r_j - 1 > 0 \text{ or } -\text{Tr}(A^{(j)}(x)) > 0.$$

Therefore, $V(x) = 0$ for all $x \in \mathcal{R}$, and $V(x) > 0$ for all $x \notin \mathcal{R}$. Hence, $V(x)$ is a penalty function. $\square$

In Lemma A.0.2 the function w can be negative for arbitrary $A \succeq 0$, even when $\text{rank}(A) = 3$. The following Theorem gives the average value of w over the subset $\mathcal{D} = \{\text{diag}(x_1, x_2, \dots, x_m) \mid x_1^2 + x_2^2 + \dots + x_m^2 = 1, x_i \geq 0\}$ of $m \times m$ positive semidefinite matrices.

Theorem A.0.2 *The average value $\bar{w}$ of*

$$w = \frac{(x_1 + \dots + x_m)^2}{x_1^2 + \dots + x_m^2} - m + 1$$

over the subset $\mathcal{D}$ (defined above) is given by

$$\bar{w} = \begin{cases} \frac{2(m-1)}{\pi} \Gamma(\frac{m}{2} + 1) \prod_{j=3}^m \frac{\Gamma(\frac{j+1}{2})}{\Gamma(\frac{j}{2} + 1)} - m + 2 & \text{if } m \geq 3 \\ \frac{2}{\pi} & \text{if } m = 2 \end{cases},$$

where Γ is the Gamma function.

Proof: Note that $\mathcal{D}$ is *homeomorphic* to the first orthant of the unit sphere. The average value of w over all positive definite matrices is equivalent to the average

value of w over the first orthant. Let

$$z = \frac{(x_1 + \dots + x_m)^2}{x_1^2 + \dots + x_m^2}, \quad S_{m-1}^* = \{x \mid x_1^2 + \dots + x_m^2 = 1, x_i \geq 0\}.$$

Then

$$w = z - m + 1. \quad (1.0.1)$$

The total value of z over the first orthant is

$$z_{tot} = \int_{S_{m-1}^*} (x_1 + \dots + x_m)^2 ds \quad (1.0.2)$$

$$= \int_{S_{m-1}^*} ds + 2 \sum_{i \neq j} \int_{S_{m-1}^*} x_i x_j ds \quad (1.0.3)$$

$$= \int_{S_{m-1}^*} ds + 2 \binom{m}{2} \int_{S_{m-1}^*} x_1 x_2 ds \quad (1.0.4)$$

We transform the integrals using the spherical coordinates [38]

$$x_1 = r \cos \theta_{m-1} \cos \theta_{m-2} \dots \cos \theta_2 \cos \theta_1$$

$$x_j = r \cos \theta_{m-1} \cos \theta_{m-2} \dots \cos \theta_j \sin \theta_{j-1}, \quad 2 \leq j \leq m-1$$

$$x_m = r \sin \theta_{m-1}$$

where $-\pi \leq \theta_1 \leq \pi$, $-\frac{\pi}{2} \leq \theta_j \leq \frac{\pi}{2}$ for $j \geq 2$. The Jacobian of the transformation is

$$J = r^{m-1} (\cos \theta_{m-1})^{m-2} (\cos \theta_{m-2})^{m-3} \dots (\cos \theta_3)^2 \cos \theta_2.$$

In the integral (1.0.4), $r = 1$. Using this transformation and the known values $\Gamma(1) = 1$, $\Gamma(1/2) = \sqrt{\pi}$, $\Gamma(x+1) = x\Gamma(x)$ for $x > 0$, and

$$\int_0^{\frac{\pi}{2}} (\cos \theta)^j d\theta = \frac{\frac{1}{2} \Gamma(\frac{j+1}{2}) \sqrt{\pi}}{\Gamma(\frac{j+2}{2})},$$

we obtain

$$\begin{aligned} \int_{S_{m-1}^*} x_1 x_2 ds &= \int_0^{\frac{\pi}{2}} \sin \theta_1 \cos \theta_1 d\theta_1 \prod_{j=2}^{m-1} \int_0^{\frac{\pi}{2}} (\cos \theta_j)^{j+1} d\theta_j \\ &= \frac{1}{2} \prod_{j=2}^{m-1} \frac{\frac{1}{2} \Gamma(\frac{j+2}{2}) \sqrt{\pi}}{\Gamma(\frac{j+3}{2})} \end{aligned}$$

By adjusting the index j and simplifying give

$$\int_{S_{m-1}^*} x_1 x_2 ds = \frac{1}{2} 2^{-m+2} \pi^{\frac{m-2}{2}} \prod_{j=3}^m \frac{\Gamma(\frac{j+1}{2})}{\Gamma(\frac{j+2}{2})}. \quad (1.0.5)$$

Equation (1.0.5) includes the case $m = 2$ by setting the product

$$\prod_{j=3}^2 \frac{\Gamma(\frac{j+1}{2})}{\Gamma(\frac{j+2}{2})} = 1.$$

The surface area of the first orthant S_{m-1}^* is given by [38]

$$A(S_{m-1}^*) = \frac{1}{2^m} \frac{m \pi^{\frac{m}{2}}}{\Gamma(\frac{m}{2} + 1)}.$$

From equations (1.0.1), (1.0.4) and (1.0.5), we get the average value of as

$$\begin{aligned} \bar{w} &= \frac{z_{tot} + (-m+1)A(S_{m-1}^*)}{A(S_{m-1}^*)} \\ &= 1 + \frac{2 \binom{m}{2} \int_{S_{m-1}^*} x_1 x_2 ds}{A(S_{m-1}^*)} - m + 1 \\ &= \begin{cases} \frac{2(m-1)}{\pi} \Gamma(\frac{m}{2} + 1) \prod_{j=3}^m \frac{\Gamma(\frac{j+1}{2})}{\Gamma(\frac{j+2}{2})} - m + 2 & \text{if } m \geq 3 \\ \frac{2}{\pi} & \text{if } m = 2 \end{cases}. \quad \square \end{aligned}$$

Applying Theorem A.0.2, we provide the following table containing m versus the the average $\bar{w}$ value over $\mathcal{D}$.

m value	$\bar{w}$ value
2	0.6366
3	0.2732
4	-1.4195
5	-3.4589
6	-5.2520
7	-6.6192
8	-7.6148

Table A.1: Dimension m vs. Average $\bar{w}$ over $\mathcal{D}$

The table shows that $\bar{w}$ is negative for $m \geq 4$, and its magnitude increases with increasing m value. Note that any matrix $A \succ 0$ has the same eigenvalues as some multiple of an element of $\mathcal{D}$. Furthermore, $\mathcal{D}$ is just the set of all normalized eigenvalues of $m \times m$ matrices A , and $A \succeq 0$ if and only if its normalized eigenvalues belongs to $\mathcal{D}$.

Therefore, the above table gives us the average $\bar{w}$ value over the set of all normalized eigenvalues. It shows that on average an $m \times m$ positive semidefinite matrix has a negative $\bar{w}$ value, for $m \geq 4$.

Bibliography

- [1] F. Alizadeh. “*Semidefinite programming*”, Ph.D. Thesis, Dept. of Computer Science, University of Minnesota, 1991.
- [2] F. Alizadeh, J. A. Haeberly and M. Overton. “Complementarity and Nongeneracy in Semidefinite Programming”, *Math. Programming*, vol. 77, 1997, no. 2, Ser. B, pp. 111-128.
- [3] E. Andersen and K. Andersen. “Presolving in linear programming”. *Math. Programming*, vol. 71, 1995, pp. 221-245.
- [4] D. S. Atkinson and P. M. Vaidya. “A scaling technique for finding the weighted analytic center of a polytope”, *Math. Programming*, vol. 57, 1992, pp. 163-192.
- [5] M. S. Bazaraa, H. D. Sherali and C. M. Shetty, “*Nonlinear Programming: theory and algorithms*”, Second edition, John Wiley and Sons. Inc. 1993.
- [6] C. Belisle, A. Boneh and R. Caron. “*Convergence properties of hit-and-run samplers*”, Technical Report WMSR 95-13, University of Windsor, Windsor, Ontario 1995. to appear in *Stochastic Models*.

- [7] H. Berbee, C. Boender, A. Rinnooy Kan, C. Scheffer and R. Smith. "Hit-and-run algorithms for the identification of nonredundant linear inequalities". *Math. Programming*, vol. 37, 1987, pp. 184-207.
- [8] A. Boneh, Private Communication with R. J. Caron.
- [9] A. Boneh and A. Golan. "Constraint Redundancy and feasible region boundedness by a random feasible point generator (RFPG), contributed paper", EURO III. Amsterdam, 1989.
- [10] J.C.G Boot, "On trivial and binding constraints in programming problems". *Management Science*, vol 8, 1962, pp. 419-441.
- [11] S. Boyd and L. El Ghaoui, "Method of Centers for Minimizing Generalized Eigenvalues", *Linear Algebra and its Applications*, vol. 188, 1993, pp. 63-111.
- [12] S. Boyd and L. El Ghaoui. "*Linear Matrix Inequalities in System and Control Theory, volume 15 of Studies in Applied Mathematics*", SIAM. Philadelphia, PA, 1994.
- [13] R. Caron, M. Hlynka and J. F. McDonald, "On the best case performance of hit and run methods for detecting necessary constraints", *Math. Programming*, vol. 54, no. 2, 1992, Ser. A, pp. 233-249.
- [14] R. Caron, J. F. McDonald, R. L. Smith and J. Telgen. "*Stopping rules for a class of random methods for detecting necessary linear inequality constraints*". University of Windsor, Department of Mathematics Report WMR86-03, 1988.

- [15] R. Caron, A. Boneh and S. Boneh, "Redundancy" in *Advances in Sensitivity Analysis and Parametric Programming*, T. Gal and H. J. Greenberg (eds.), Kluwer Academic Publishers. International Series in Operations Research and Management Science. vol. 6. 1997 (invited).
- [16] J. W. Chinneck, "Locating Minimal Infeasible Constraint Sets in Linear Programs", *ORSA Journal on Computing*, vol. 3, 1991, pp. 157-168.
- [17] J. W. Chinneck, "Analyzing Infeasible Nonlinear Programs", *Computational Optimization and Applications.*, vol. 21, no. 2, 1995, pp. 167-179.
- [18] J. Cullum, W. E. Donath and P. Wolfe, "*The minimization of certain nondifferentiable sums of eigenvalue problems*", Math. Programming Stud.: vol. 3, 1975, pp. 35-55.
- [19] John D. Dixon, "Estimating Extremal Eigenvalues and Condition Numbers of Matrices", *SIAM J. NUMER. ANAL.*, vol. 20, no. 4, 1983, pp. 812-814.
- [20] A. L. Fradkov and V. A. Yakubovich, "The S-Procedure and duality relations in nonconvex problems of quadratic programming", *Vestnik Leningrad Univ. Math.*, 6(1):101-109, 1979. In Russian, 1973.
- [21] Gene H. Golub, "*Matrix Computations*", Johns Hopkins University Press, Baltimore, 1983.
- [22] M. Grotschel, L. Lovasz and A. Schrijver, "*Geometric Algorithms and Combinatorial Optimization*", Springer-Verlag 1988.

- [23] R. A. Horn and C. R. Johnson, "*Matrix Analysis*", Cambridge University Press 1996.
- [24] S. Jibrin, "*The Stand and Hit Algorithm for Detecting Necessary Linear Constraints*". Masters Thesis, University of Windsor, 1993.
- [25] M. H. Karwan, V. Lofti, J. Telgen and S. Zionts, "*Redundancy in Math. Programming*". Springer-Verlag, Berlin, 1983.
- [26] L. G. Khachiyan, "Rounding of Polytopes in the Real Number Model of Computation", *Math. Oper. Res.*, vol. 21, no. 2, May 1996, pp. 307-320.
- [27] R. B. Lehoucq and D. C. Sorensen, "Deflation Techniques within an Implicitly Restarted Arnoldi Iteration", *SIAM Journal On Matrix Analysis and Applications*, vol. 17, 1996, pp. 789-821.
- [28] I. Lustig, R. Marsten and D. Shannon, "Interior point methods for linear programming: Computational state of the art", *ORSA Journal on Computing*, vol. 6, 1994, pp. 123-135.
- [29] G. I. Nemhauser, L. A. Wolsey, "*Integer and Combinatorial Optimization*", John Wiley and Sons, Inc 1988.
- [30] Yu. Nesterov and A. Nemirovsky, "*Interior-point polynomial methods in convex programming, volume 13 of Studies in Applied Mathematics*", SIAM, Philadelphia, PA, 1994.
- [31] B. Noble and J. W. Daniel, "*Applied Linear Algebra*", Prentice-Hall 1988.

- [32] W. T. Obuchowska and R. J. Caron, "Minimal representation of quadratically constrained convex quadratic programmes", *Math. Programming*, vol. 68, 1995, pp. 169-186.
- [33] P. M. Pardalos, "Quadratic programming with one negative eigenvalue is NP-hard", *Journal of Global Optimization*, vol. 1, 1991, pp. 15-22.
- [34] M. Ramana, "An exact duality theory for semidefinite programming and its complexity implications", DIMACS Technical Report 95-02. RUTCOR, Rutgers University, 1995.
- [35] S. Sartaj, "Computationally related problems", *SIAM J. Comput.*, vol. 3, 1974, pp. 262-279.
- [36] B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema and C. B. Moler, "*Matrix Eigensystems Routines - EISPACK Guide*", Lecture Notes in Computer Science, Springer-Verlag Berlin, Heidelberg 1974.
- [37] R. L. Smith, "Monte Carlo procedures for generating random feasible solutions to mathematical programs", invited paper, ORSA/TIMS, Washington, DC, 1980.
- [38] A. H. Stroud, "*Approximate Calculation of Multiple Integrals*", Prentice-Hall, Inc., Englewood Cliffs, N. J., 1971.
- [39] J. Telgen, Private Communication with A. Boneh, 1980.
- [40] J. A. Tomlin and J. S. Welch, "Finding duplicate rows in a linear programming model", *Operations Research Letters*, vol. 5, no. 1, 1986, pp. 7-11.

- [41] L. Vandenberghe and S. Boyd, "Primal-dual potential reduction method for problems involving matrix inequalities", *Math. Programming*, vol. 69 , 1995, pp. 205-236.
- [42] L. Vandenberghe and S. Boyd, "Semidefinite Programming". *SIAM Review*, vol. 38, 1996, pp. 49-95.
- [43] L. Vandenberghe, S. Boyd and S. -P. Wu, "Determinant maximization with linear matrix inequality constraints", To appear *SIMAX*, April 1998.
- [44] H. Wolkowicz and P. H. Styan, "Bounds for Eigenvalues using Traces", *Linear Algebra and its Applications*, vol. 29, 1980, pp. 471-506.
- [45] H. Wolkowicz, "Some applications of optimization in matrix theory", *Linear Algebra and its Applications*, vol. 40, 1981, pp. 101-118.
- [46] S. -P. Wu and S. Boyd, "*SDPSOL - A Parser/Solver for Semidefinite Programming and Determinant Maximization Problems with Matrix Structure*". User's Guide, Version Beta. May 1996.

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved