

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.

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.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

RICE UNIVERSITY

Ellipsoidal Approximation to Polytopes and Computational Study of Lenstra's Algorithm

by

Liyan Gao

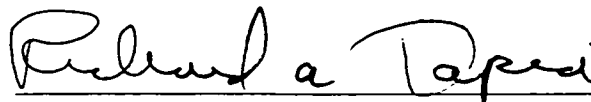
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

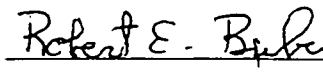
APPROVED, THESIS COMMITTEE:



Yin Zhang, Chairman
Associate Professor of Computational and
Applied Mathematics



Richard A. Tapia
Noah Harding Professor of Computational
and Applied Mathematics



Robert E. Bixby
Noah Harding Professor Emeritus of
Computational and Applied Mathematics



Richard A. Stong
Professor of Mathematics

Houston, Texas

December, 2001

UMI Number: 3047307

UMI[®]

UMI Microform 3047307

Copyright 2002 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Ellipsoidal Approximation to Polytopes and Computational Study of Lenstra's Algorithm

by

Liyan Gao

General integer programming is an important mathematical approach for many decision-making problems. In this field, a major theoretical breakthrough came in 1983 when H. W. Lenstra, Jr. proposed a polynomial-time algorithm for general integer programs while the number of variables is fixed. Two key ingredients of Lenstra's algorithm are ellipsoidal approximation of polytopes and lattice basis reduction. However, the lack of practically efficient algorithms and software for the ellipsoidal approximation of polytopes has made it difficult to study the computational properties of Lenstra's algorithm.

To bridge the gap between theory and computational practice for Lenstra's algorithm, we study both the ellipsoidal approximation to polytopes and computational properties of Lenstra's algorithm in this thesis. We have developed a reliable and efficient algorithm for computing the maximum volume ellipsoid inscribing a given polytope. This algorithm effectively exploits the problem-specific structures and utilizes a primal-dual type, interior point method. We show that this algorithm has a sound theoretical foundation, and demonstrate that it performs considerably better than a number of other algorithms through extensive numerical experiments.

Using our ellipsoidal approximation algorithm as a subroutine, we have implemented a version of Lenstra's algorithm for general integer programming feasibility problems. At each node, the method uses ellipsoidal approximation and lattice basis reduction to find a "thin" direction of the polytope, and branches on hyperplanes, rather than on variables as in the traditional branch-and-bound method. In this procedure, it is guaranteed that the number of branches at each node is bounded and small. Our numerical results on small- to medium-sized test instances suggest that Lenstra's algorithm examine much fewer nodes than the traditional branch-and-bound method. However, there is a tradeoff between many nodes and fast re-optimization as in the traditional branch-and-bound method and fewer nodes but more time-consuming decisions on branching as in Lenstra's algorithm. Currently, the main bottle-neck in the performance of the algorithm lies at the step of lattice basis reduction. If this step is sufficiently improved, then Lenstra's algorithm, when combined with other techniques such as cutting planes, promises to be efficient for certain classes of difficult problems.

Acknowledgments

This thesis would not have been possible without the help of many people. I am deeply grateful to all of them for sharing their knowledge and spending significant time and effort to make my stay at Rice fruitful and pleasant.

I am deeply grateful to my advisor Dr. Yin Zhang. Dr. Zhang has shown a genuine interest in my thesis work from its conception to completion. My research has benefited greatly from our animated discussions. Dr. Zhang always encourages me to present my work, and supports me to broaden my scientific knowledge and to improve myself by attending conferences. I also appreciate his patience and frankness throughout the years when I met difficulties in research. I thank Dr. Bill Cook who taught me to take the challenge of difficult research problems and set a high standard for me. I greatly appreciate his guidance in integer programming. Even after he left Rice, he still paid attention to my work and went through my draft and provided constructive comments to improve it. I truly appreciate his encouragement and enthusiasm on my thesis.

I would like to express my sincere appreciation to all my committee members for their guidance and support. I thank Dr. David Applegate, Dr. Bill Cook, Dr. Robert Bixby and Dr. Richard Tapia for their seminars and comments.

I would express my thanks to those who helped me during my research and thesis writing. Dr. Bixby pointed me to the set of test problems for integer programming. These test problems were very useful in my research. Dr. Karen Aardal has shared her interest in lattice basis reduction with me in the beginning of my research. Dr. Jan Hewitt edited a great deal of my proposal and my thesis. Keith Berrier critically read my thesis.

I would like to thank my officemates Keith Berrier, Maria Cristina Villalobos and Erica Zimmer Klampfl for sharing their experience with me and for their friendship. My special thanks go to Genetha Gray for her comments on my presentations. I thank all my friends and classmates and roommates at Rice for the time we spent together.

I thank all the faculty and staff of Computational and Applied Mathematics Department for providing such a stimulating environment. I thank our department coordinator Daria Lawrence for taking care of daily details, and Michael Pearlman and Eric Aune for their smooth work on our computer network.

I am thankful to my supportive family (Mom and Dad, uncles and aunts, and sisters) for their love, patience and belief in me. Finally, I thank my husband Jibin, for his love, encouragement and support.

Contents

Abstract	ii
Acknowledgments	iv
List of Illustrations	ix
List of Tables	x
1 Introduction	1
2 Background	7
2.1 Linear Programming and Integer Programming Concepts	7
2.2 Branch-and-Bound Algorithms	11
2.3 Cutting Plane Methods	14
2.4 Interior Point Methods for Linear Programming	15
2.5 Convex Programming	18
3 Ellipsoidal Approximation of Polytopes	21
3.1 Introduction	21
3.2 The Maximum Volume Ellipsoid Problem	25
3.3 Formulations and Primal-Dual Algorithms	28
3.3.1 Formulations without Matrix Variable	28
3.3.2 Primal-Dual Algorithmic Framework	30
3.4 Theoretical Results	35
3.4.1 Well-Definedness of Algorithms	35
3.4.2 Uniqueness of Solution	39
3.4.3 Existence and Convergence of Paths	41

3.4.4	Issues of Algorithmic Convergence	49
3.5	Khachiyan-Todd Algorithm and Modification	49
3.5.1	Khachiyan and Todd's Algorithm	50
3.5.2	A Modification of the KT Algorithm	52
3.6	Numerical Results	53
3.6.1	Implementation Details	54
3.6.2	Test Problems	56
3.6.3	Test Results	57
3.7	Concluding Remarks	60
4	Lenstra's Algorithm for Integer Programming	66
4.1	Introduction	66
4.2	Lenstra's Algorithm and Improvement	69
4.3	Lattices and Basis Reduction	72
4.3.1	Lattices and Dual Lattices	72
4.3.2	Lattice Basis Reduction	75
4.4	Generalized Basis Reduction Algorithm	81
5	Computational Study of Lenstra's Algorithm	84
5.1	Non-full-dimensional Polytopes	84
5.1.1	Identify Non-full-dimensional Cases and Implicit Equalities . .	85
5.1.2	Hermite Normal Form and Linear Diophantine Equations . . .	86
5.1.3	Projection Into Lower Full-Dimensional Case	88
5.2	Implementation	89
5.3	Choices of the Branching Direction	95
5.3.1	Babai's Direction Choice in Close Vector Algorithm	96

5.3.2	Direction of the Smallest Number of Intersecting Hyperplanes	96
5.3.3	Direction Closest to the Smallest Eigenvector of E	98
5.4	Results and Discussion	98
5.4.1	Test Problems	99
5.4.2	Numerical Results	101
5.4.3	Discussion	102
6	Conclusions	106
	Bibliography	109

Illustrations

2.1	Branch-and-Bound Tree on Binary IP Problem	12
2.2	LP Based Branch-and-Bound Method For MIP Problems	13
2.3	A Primal-Dual Interior Point Method Framework	17
4.1	A Thin Polytope in 2-D	70
5.1	Search Tree of Branching on Hyperplanes	90
5.2	Flowchart of Lenstra's Algorithm	93
5.3	Babai's Choice of Branching Direction in 2-D	97

Tables

3.1	Summary of Results for Tests 1 and 2	58
3.2	Results of Test Set 3: Problems 1-10	59
3.3	Results of Test Set 1: Problems 1-47	62
3.4	Results of Test Set 2: Problems 1-48	63
3.5	Results of Test Set 2: Problems 49-96	64
3.6	Results on Test Set 2: Problems 97-143	65
5.1	IP Test Problems	100
5.2	Computational Time	102
5.3	Node Counts	103
5.4	A Sample on Time of the Ellipsoidal Approximation and the Lattice Basis Reduction	105

Chapter 1

Introduction

As suggested by the title, this thesis has two main parts: ellipsoidal approximation to polytopes and computational study of Lenstra's algorithm.

Ellipsoidal approximation of convex bodies has long been studied and many theoretical results also use it as an analytic tool because ellipsoids have better geometric and algebraic properties than general convex bodies. Especially with the development of interior point methods since the middle of 1980s, several algorithms for finding the maximum volume ellipsoid (MaxVE) inside a polytope have been proposed [25, 34, 48, 33, 3]. However, those works are mainly concerned with the complexity issues and remain at the theoretical level. One contribution of this thesis is the design and development of a numerically efficient and stable algorithm and software for solving the MaxVE problem using the primal-dual interior point algorithm.

One application of ellipsoidal approximation of polytopes is Lenstra's algorithm [28] for general integer programming problems, proposed by H. W. Lenstra, Jr. in 1983. With this algorithm, Lenstra proved the polynomial-time solvability for the integer programming problem with a fixed number of variables. The algorithm is a breakthrough in the theory of integer programming. However, Lenstra's algorithm has been considered as a theoretical result and no computational results using ellipsoidal approximation has been previously reported. Therefore, other contributions of this thesis include the implementation of a version of Lenstra's algorithm using our result on ellipsoidal approximation as a subroutine and the insight we have gained on the computational properties of Lenstra's algorithm. We were interested in solving

small-sized to medium-sized hard problems because of the limited capacity currently available for lattice basis reduction.

Zoom Out Into A Bigger View

According to the Mathematical Programming Society, optimization or mathematical programming is a branch of applied mathematics concerning the problem of optimizing a function of many variables, often subject to a set of constraints. Included, along with the standard topics of linear, nonlinear, integer and stochastic programming, are computational testing, techniques for formulating and applying mathematical programming models, unconstrained optimization, convexity and the theory of polyhedra, and control and game theory viewed from the perspective of mathematical programming.

Depending on the property of the variables, optimization can be divided into two categories: continuous optimization versus discrete optimization. The characteristic feature of discrete, combinatorial or integer optimization (also integer programming or IP) is that some of the variables are required to belong to a discrete set, most commonly a subset of integers. The discrete restriction corresponds to economical indivisibility. Discrete optimization has widespread applications in reducing inventory and cost, increasing productivities and enhancing revenue in economics and management. For example, the scheduling problem, the transportation problem and the assignment problem are all discrete optimization problems. However, the integer programming problem belongs to a class of the most difficult problems, NP-Complete problems. Two most widely used general solution techniques for solving integer programming problems are the linear programming based branch-and-bound method and the recently developed branch-and-cut method. Even though the branch-and-bound method works well on most practical problems, it can become inefficient when the branch-and-bound tree grows extremely large. In 1983, H. W. Lenstra proposed a

new algorithm and proved that the integer programming problem is solvable in polynomial time for a fixed number of variables, a breakthrough in the theory of integer programming. Grötschel, Lovász and Schrijver [14] further developed his algorithm. What we are interested here is the version of Lenstra's algorithm using ellipsoidal approximation.

Karmarkar's work reported in 1984 [21] started a golden era for interior point methods. The development of this methodology was also a celebrated event in the history of linear programming. Interior point methods, particularly primal-dual interior point methods, emerged from theoretical beauty to becoming practical competitors of the simplex method. In the last decade, interior point methods have gained rapid development and been utilized not only in linear programming but also in convex programming, semidefinite programming and conic programming.

With the development of interior point methods, the ellipsoidal approximation problem has received more attention. However, most results obtained so far stay at the computational complexity level. As early as 1948, F. John [17] proved the existence of a pair of ellipsoids to approximate a convex body. This ellipsoidal approximation can be proven to be polynomial-time solvable by applying the ellipsoid method [30]. While the ellipsoid method has nice theoretical complexity, it was not practically efficient.

Motivations of This Thesis

Since 1984 when Karmarkar's remarkable work was introduced, dramatic progress has been made in interior point methods. Based on the primal-dual interior point methodology, we propose to develop and implement a practically efficient interior point algorithm for the specific convex programming problem: finding the maximum volume ellipsoid (MaxVE) inside a given polytope. A motivation for studying the MaxVE problem is that once the problem is solved, it can be utilized in many appli-

cations such as in linear control problems and in optimal design. So far most results on this problem have stayed at the theoretical level and the lack of practically efficient algorithms and software has hindered its applications.

Currently, the LP based branch-and-bound method is used in almost all commercial IP codes. It solves IP problems via solving a large number of LP problems. In the last ten years or so, improvements on the branch-and-bound method, i.e., cutting plane methods and the branch-and-cut method, have made it possible to solve IP and mixed integer programming (MIP) problems much more efficiently. However, the currently available software have limitations. On the one hand, the size of IP or MIP problems that can be solved by modern software has become larger and larger; on the other hand, some small but hard IP problems remain unsolved. Most of the unsolved problems fall into a category of general IP problems while most progress has been made on the 0-1 binary integer programming (BIP) problems. One difficulty with the current techniques on those small but hard problem is that the branch-and-bound tree grows too large even with the combination of cutting plane methods. As early as 1983, H. W. Lenstra proposed a different branching strategy that guarantees a bounded search tree, which is promising to work well on those small but hard general integer programming problems. He constructed an algorithm to branch on as few as possible hyperplanes instead of on variables, and proved that the IP problem can be solved in polynomial time for a fixed number of variables. The distinct feature of Lenstra's algorithm resides on the way to branch on hyperplanes. The algorithm measures the width of the polytope given by the LP relaxation of the original IP problem and chooses a flat direction that has a bounded number of hyperplanes to branch on. Thus the number of nodes on the search tree is dramatically reduced. Lenstra's algorithm is a breakthrough in IP theory; however, on the computational

side, the techniques involved, lattice basis reduction and ellipsoidal approximation of polytopes, were not quite ready.

As an effort toward solving the above problems, my thesis work consists of two parts: ellipsoidal approximation and computational study of Lenstra's algorithm for integer programming. We have developed and implemented a primal-dual interior point method for the MaxVE problem that takes advantage of the special structure of this convex programming problem. We have compared the numerical performance of four algorithms for the MaxVE problem: Khachiyan and Todd's algorithm, its modification and two of our direct primal-dual interior point methods. One of the direct primal-dual interior point methods for MaxVE outperforms the other three. On the IP side, I have implemented a version of Lenstra's algorithm using our results on ellipsoidal approximation and have tested it on a set of difficult IP problems. Our computational work on Lenstra's algorithm has provided insight for its further study.

Organization

In Chapter 2, we give the notation, terminology and other background information on integer programming, interior point methods and convex programming. In Chapter 3, we analyze the ellipsoidal approximation problem, one of the two interests of this thesis. We explore the special structure of the MaxVE problem and develop an numerically efficient and stable algorithm based on the primal-dual interior point methodology. We modify Khachiyan and Todd's algorithm, implement both the original and modified algorithms, and compare their numerical performance on 200 test problems.

Chapter 4 makes a connection between ellipsoidal approximation, lattice basis reduction and integer programming through Lenstra's algorithm and gives a brief comparison between the generalized basis reduction algorithm and Lenstra's algo-

rithm. In Chapter 5, we present and discuss the computational results obtained by our implementation of Lenstra's algorithm, the other interest of this thesis.

Chapter 6 summarizes the contributions we have made in this thesis: analysis and development of an algorithm for solving the MaxVE problem, and implementation of a version of Lenstra's algorithm using ellipsoidal approximation. We also discuss future directions in this chapter.

Chapter 2

Background

In this chapter, we introduce some notation and terminology in integer programming, including two general algorithms for integer programming (branch-and-bound method and branch-and-cut method), interior point methods for linear programming and convex programming. Linear programming is closely related not only to integer programming but also to interior point methods.

2.1 Linear Programming and Integer Programming Concepts

The linear programming (LP) problem, as a widely used tool in solving practical models, is one of the fundamental problems in mathematical programming. An LP problem can be written as

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \end{aligned} \tag{2.1}$$

where the *polyhedron* $P := \{x \in R^n : Ax \leq b\}$ is its *feasible region*, the linear inequalities in $Ax \leq b$ are called the *constraints* and $c^T x$ is the *objective function*. If \bar{x} satisfies $A\bar{x} \leq b$ or \bar{x} is inside the feasible region P , it is called a *feasible solution*: if x^* is a feasible solution such that $c^T x^* \geq c^T \bar{x}$, for any $\bar{x} \in P$, then x^* is called an *optimal solution*.

A closely related problem to (2.1) is its *dual problem* defined as $\min\{b^T y : A^T y = c, y \geq 0\}$. The original problem is also called the *primal problem*. The duality theorem, proved by D.Gale, H.W. Kuhn and A.W. Tucker [13], illustrates the connection between the primal and the dual problem.

Theorem 2.1 (*Duality Theorem of Linear programming*) Given a matrix $A \in R^{m \times n}$, a vector $b \in R^m$ and another vector $c \in R^n$. If the feasible regions of the primal problem and the dual problem are both nonempty, then

$$\max\{c^T x : Ax \leq b\} = \min\{b^T y : A^T y = c, y \geq 0\}.$$

If one of the primal or dual problems has no feasible solution (i.e. infeasible), then the other is infeasible or unbounded. If one of the primal or dual problem is unbounded, the other must be infeasible.

Duality theory has two-fold importance. On the computational side, it gives the equivalence between solving the primal problem and solving the dual problem. On the theoretical side, it establishes a way to prove optimality of LP solutions. Furthermore, any dual feasible solution provides an upper bound on the optimal value of the primal problem.

A useful optimality condition for LP problems is the complementary slackness theorem.

Theorem 2.2 (*Complementary Slackness Theorem*) Let \bar{x} be a feasible solution to the primal problem $\max\{c^T x : Ax \leq b\}$ and \bar{y} be a feasible solution to the dual problem $\min\{b^T y : A^T y = c, y \geq 0\}$. Necessary and sufficient conditions for the optimality of both \bar{x} and \bar{y} are

$$\bar{y}_i > 0 \text{ implies } \sum_{j=1}^n a_{ij} \bar{x}_j = b_i, \text{ for every } i$$

and

$$\sum_{j=1}^n a_{ij} \bar{x}_j < b_i \text{ implies } \bar{y}_i = 0, \text{ for every } i.$$

To look at an LP problem geometrically, we introduce more terms in polyhedra. For detailed information, readers can refer to Schrijver [42]. The solution set of a system of linear inequalities is called a *polyhedron* $\mathcal{P} = \{x \in R^n : Ax \leq b\}$. A polyhedron \mathcal{P} is bounded if there exists $r > 0$ such that $\mathcal{P} \subseteq S(0, r)$, where $S(0, r)$ represents a ball of radius r centered at the origin. A bounded polyhedron is called a *polytope*. The polyhedron $\{x \in R^n : a^T x \leq \beta, a \in R^n, \beta \in R\}$ is called a *half space* and the special polyhedron $\{x \in R^n : a^T x = \beta\}$ is called a *hyperplane*. It is clear that every polyhedron is the intersection of a finite number of half spaces. An inequality $a^T x \leq \beta$ is called *valid* with respect to a polyhedron \mathcal{P} if $\mathcal{P} \subseteq \{x \in R^n : a^T x \leq \beta\}$. A set $F \subseteq \mathcal{P}$ is called a *face* of \mathcal{P} if there exists a valid inequality $a^T x \leq \beta$ for \mathcal{P} such that $F = \{x \in \mathcal{P} : a^T x = \beta\}$. Thus F is called the *face induced (or defined) by* $a^T x \leq \beta$. A *facet* of \mathcal{P} is the maximum face distinct from \mathcal{P} with respect to inclusion. If a point $v \in \mathcal{P}$ is a face of \mathcal{P} , then v is called the a *vertex* of \mathcal{P} . A polyhedron is called *pointed* if it has a vertex.

The set of optimal solutions of an LP problem over a polyhedron \mathcal{P} comprises a face of \mathcal{P} . If a polyhedron \mathcal{P} is pointed, every face of \mathcal{P} contains at least a vertex. Therefore, under the conditions that \mathcal{P} is pointed and the associated LP problem $\max\{c^T x | x \in \mathcal{P}\}$ is bounded (which by definition means that the optimal value of the LP is finite), the LP problem has at least one optimum solution x^* that is a vertex of \mathcal{P} . In other words, every bounded LP problem over a nonempty polyhedron has an optimum vertex solution. This argument was applied to the simplex method by Dantzig [10] for LP, whose iteration moves from one vertex of the corresponding polytope to another vertex until it finds an optimal vertex point. Based on the duality theorem, one can apply the simplex method to the dual problem without writing down the dual problem explicitly, which is the so-called *dual simplex method*.

To analyze the structure of a polyhedron, it is useful to divide the linear inequalities into two groups. An inequality $a^T x \leq \beta$ from $Ax \leq b$ is called an *implicit equality* if $a^T x = \beta$ for all $x \in \mathcal{P} = \{Ax \leq b\}$. Thus the linear system defining the polytope can be divided into two :

$$A^-x \leq b^- \quad \text{the subsystem of implicit equalities in } Ax \leq b;$$

$$A^+x \leq b^+ \quad \text{the subsystem of all other inequalities in } Ax \leq b.$$

Hence $\mathcal{P} = \{x : Ax \leq b\} = \{x : A^-x = b^-, A^+x < b^+\}$. A constraint is called *redundant* if it is implied by the other constraints in the constraint system. The dimension of a polytope is equal to $n - \text{rank}(A^-)$. Therefore, \mathcal{P} is full-dimensional if and only if the linear system has no implicit equalities.

With additional requirement that $x \in R^n$ be an integral vector to an LP problem $\max\{c^T x : Ax \leq b\}$, the problem becomes an *integer linear programming problem* (in short, IP). The corresponding LP without the integer requirement is called the *LP relaxation* of the IP problem. With the integer restriction, the feasible region becomes the set of discrete integer points inside the polyhedron instead of the whole polyhedron corresponding to the LP relaxation problem. The difficulty of solving IP increases tremendously as the IP problem is NP-complete as proved by Cook [8].

The *mixed integer programming problem* is a common model in practice when some of the variables are continuous and others are discrete, i.e.,

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & l \leq x \leq u \\ & x_j \text{ integral, for } j \in J \end{aligned} \tag{2.2}$$

where $A \in R^{m \times n}$, $b \in R^m$, $x \in R^n$ which is between its lower bound l and its upper bound u , and $J \subseteq \{1, 2, \dots, n\}$.

To solve LP problems, several solution techniques have been developed, including the simplex method [10], the ellipsoid method [23] and recently interior-point methods (see [22], for example). From the complexity point of view, the simplex method is not a polynomial algorithm even though it works quite efficiently in practice. The dual simplex method has remained the algorithm of choice for re-optimizations in the branch-and-bound method for solving IP problems. Khachiyan's ellipsoid algorithm [23] was the first polynomial algorithm with $O(n^2L)$ iterations for LP, where L is a measure for the size of an LP problem. However, the algorithm is much slower than the simplex method. Karmarkar's polynomial algorithm [22] with $O(nL)$ iterations and its improvement by Renegar [38] with $O(\sqrt{n}L)$ iterations have good complexity and perform much better than Khachiyan's ellipsoid algorithm. Moreover, later versions of primal-dual interior point methods are computationally competitive with the simplex method.

In the following sections, we first summarize two general algorithms for solving IP, branch-and-bound method and branch-and-cut method, and then give a review on the basics of interior point methods and convex programming problems.

2.2 Branch-and-Bound Algorithms

Based on the definitions of LP and IP, it is natural to make use of LP solution techniques to attack IP problems. One of the solution techniques is the LP based branch-and-bound method. The branch-and-bound method is also termed as *implicit enumeration* or *divide-and-conquer approach*. The idea is to divide the feasible domain into several subdomains and solve the subproblem on each subdomain and make use of the optimal value of the subproblem to analyze the upper and lower bounds on the IP objective function. Once the upper bound equals the lower bound, the optimality is achieved.

A simple case is the LP based branch-and-bound algorithm on binary IP problems, where $x \in \{0, 1\}^n$. Many decision variables can be modeled as binary integer variables. In the original problem, a variable x_i is chosen to branch on and two subproblems are generated: one with x_i fixed to 1 and the other to 0. One then solves the two subproblems and updates the upper and lower bounds on the original problem. This process continues until the upper bound equals the lower bound or the whole binary tree is searched and no integer feasible solution exists. This whole procedure is an implicit enumeration demonstrated in Figure 2.1. We call different child nodes of the same parent node *siblings* on the search tree.

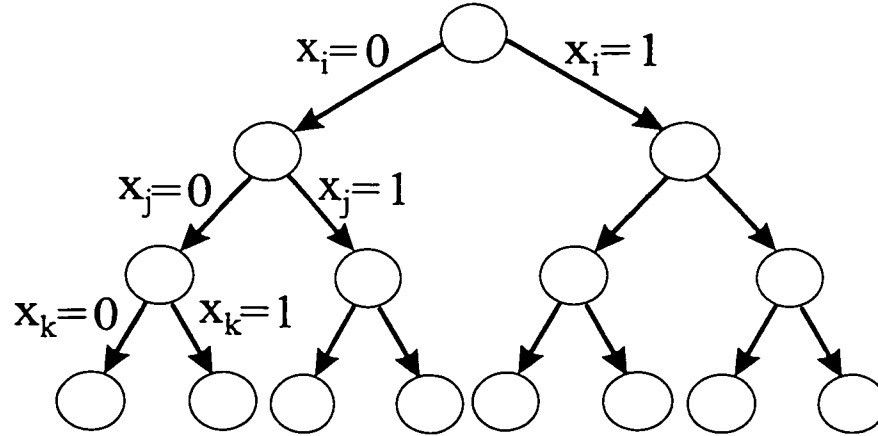


Figure 2.1 Branch-and-Bound Tree on Binary IP Problem

For general integer programming problem, the branch-and-bound tree can grow much larger than for BIP problems. To be more general, we describe an LP based branch-and-bound algorithm given by Land and Doig [26] in terms of the MIP in Figure 2.2. Pure IP problems can be treated as a special case when all variables are required to be integral. In the Step 3 of the node-selection and the relaxation, different strategies can be applied. Readers are directed to a recent study by Linderoth and Savelsbergh [18] for more details.

The terminologies and definitions introduced in an MIP problem are explained as follows. We use P to denote the original MIP problem to minimize the objective function. z^I is the current best objective value and X^I records the integer feasible solution corresponding to the value of z^I . A node associated with an LP relaxation problem P' and a lower bound z' of P' is denoted as (P', z') .

LP Based Branch-and-Bound Method For MIP Problems

- Step 1 (Initialization):** Let $\mathcal{T} = \{(P, -\infty)\}$, $z^I = +\infty$, X^I is undefined.
- Step 2 (Termination):** If $\mathcal{T} = \emptyset$, STOP. In this case, if $z^I < +\infty$, then X^I is an optimal solution; otherwise, P is infeasible.
- Step 3 (Node selection and relaxation):** Select and delete a pair (node) (P', z') from \mathcal{T} . Solve the LP relaxation of P' . Let X' be an optimal solution, if one exists, and set $z' = c^T X'$; otherwise, set $z' = +\infty$.
- Step 4 (Fathoming):** Execute (a), (b), (c), in order:
- (a) If $z' \geq z^I$, go to Step 2.
 - (b) If X' is not integer feasible for P' go to Step 5.
 - (c) Let $z^I = z'$ and $X^I = X'$. Delete from \mathcal{T} all pairs (P'', z'') with $z'' \geq z^I$. Go to Step 2.
- Step 5 (Branching):** Let $j \in J$ be such that X'_j is fractional. Let P'_l be P' with the added restriction that $x_j \leq \lfloor X'_j \rfloor$, and let P'_u be P' with the added restriction that $x_j \geq \lceil X'_j \rceil$. Replace \mathcal{T} by $\mathcal{T} \cup \{(P'_l, z'), (P'_u, z')\}$. Go to Step 2.

Figure 2.2 LP Based Branch-and-Bound Method For MIP Problems

2.3 Cutting Plane Methods

Geometrically, different polytopes may include the same set of integer points, corresponding to the feasible point set of an IP problem. If all vertices of the polytope are integers, the solution to the corresponding relaxed LP problem will be integer by applying the simplex method and it will also be an optimal solution to the IP. The linear system corresponding to such a polytope is an ideal system. However, for general integer programming problems, it is almost impossible to find such an ideal system except for some specific problems.

Some concepts are introduced below to describe the feasible set. Given a nonempty finite set $S \in R^n$, the *linear hull* of the elements in S is defined as $lin(S) = \{x \in R^n : x = \sum \lambda_i s_i, s_i \in S\}$. If $\lambda \geq 0$ is enforced, then it is called the *conic hull*, $cone(S)$. If $\sum_{i=1}^k \lambda_i = 1$ is required, then we have *affine combination* and *affine hull*, $aff(S)$, correspondingly. If both $\lambda \geq 0$ and $\sum_{i=1}^k \lambda_i = 1$ are required, they are called a *convex combination* and *convex hull* $conv(S)$ accordingly. Hence the ideal formulation we discussed above is the convex hull of the integer points inside the polytope of the IP problem, which is also called the *integer hull*.

While it is difficult to find the integer hull of an IP problem, the integer hull can be approximated by adding valid inequalities. If an inequality $a^T x \leq b$ is satisfied for all $x \in X \subseteq R^n$, then $a^T x \leq b$ is called a *valid inequality*. Cutting plane methods are concerned with how to construct valid inequalities (or cutting planes) for IP problems. A recent book on the advances in IP including cutting plane methods is written by Wolsey [51].

Cutting plane methods have already been applied in commercial codes such as CPLEX [36] to work together with the branch-and-bound procedure, which is so called branch-and-cut procedure. On each node of the branch-and-bound tree, first solve the LP relaxation problem, then construct multiple cutting planes if the optimal

solution is not integral. By adding cutting planes, i.e. valid inequalities, the non-integral optimal solution to the relaxed LP problems is cut off and the polytope gets closer to the integer hull step by step. In this way cutting planes can reduce the size of the feasible region and improve the bounds on the objective function. It accelerates the process of the branch-and-bound by supplying better bounds to IP or MIP problems.

2.4 Interior Point Methods for Linear Programming

Interior point methods have been developed for solving linear, semidefinite, and convex programming problems. We are interested in using the interior point methodology to solve a specific convex programming problem, the MaxVE problem. We first review some basic concepts in interior point methods for linear programming in this section.

A linear programming problem can be written in the following form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \end{aligned} \tag{2.3}$$

where $c \in R^n$, $x \in R^n$, $A \in R^{m \times n}$ and $b \in R^m$. Its dual problem is

$$\begin{aligned} \max \quad & b^T y \\ \text{s.t.} \quad & A^T y + s = c, \\ & s \geq 0 \end{aligned} \tag{2.4}$$

where $y \in R^m$ and $s \in R^n$.

For general constrained optimization problem, the Karush-Kuhn-Tucker (or KKT) conditions are the necessary optimization conditions under some constraint qualifica-

tions. Specifically, the vector x^* is a solution of the primal LP problem if and only if the following conditions hold at (x^*, y^*, s^*) for some $y^* \in R^m$ and $s^* \in R^n$:

$$F(x, y, s) = \begin{bmatrix} A^T y + s - c \\ Ax - b \\ XSe \end{bmatrix} = 0. \quad (2.5)$$

$$(x, s) \geq 0$$

where $X = \text{Diag}(x_1, x_2, \dots, x_n)$, $S = \text{Diag}(s_1, s_2, \dots, s_n)$ and $e \in R^n$ is the vector of all ones. The condition $XSe = 0$ is called the *complementary condition*. The complementary condition is the only nonlinear part in the system (2.5) and the other two equations are the dual feasibility and primal feasibility conditions. The vector (x^*, y^*, s^*) is called a *primal-dual solution*. The primal-dual *feasible set* \mathcal{F} and *strictly feasible set* \mathcal{F}^o are defined as:

$$\mathcal{F} = \{(x, y, s) \mid Ax = b, A^T y + s = c, (x, s) \geq 0\}.$$

$$\mathcal{F}^o = \{(x, y, s) \mid Ax = b, A^T y + s = c, (x, s) > 0\}.$$

We introduce a new system with respect to a scalar parameter $\tau > 0$ as follows:

$$F(x_\tau, y_\tau, s_\tau) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix} \quad (2.6)$$

$$(x_\tau, s_\tau) > 0$$

The *central path* is defined as $\mathcal{C} = \{(x_\tau, y_\tau, s_\tau) \mid \tau > 0\}$, where (x_τ, y_τ, s_τ) is the solution to (2.6). It is known that \mathcal{C} converges to a primal-dual solution of the linear program as $\tau \downarrow 0$. In primal-dual interior point methods, the central path plays the role of guidance for the iterations, keeping the products $x_i s_i$ strictly positive and decreasing to zero at the same rate.

A *centering parameter* $\sigma \in [0, 1]$ is introduced and a *duality measure* or *duality gap* μ is defined as $\mu = x^T s / n$. The linear system associated with applying Newton's method to (2.6), with $\tau = \sigma\mu$, is

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe + \sigma\mu e \end{bmatrix}. \quad (2.7)$$

We now describe a general framework for primal-dual interior point methods in Fig. 2.3.

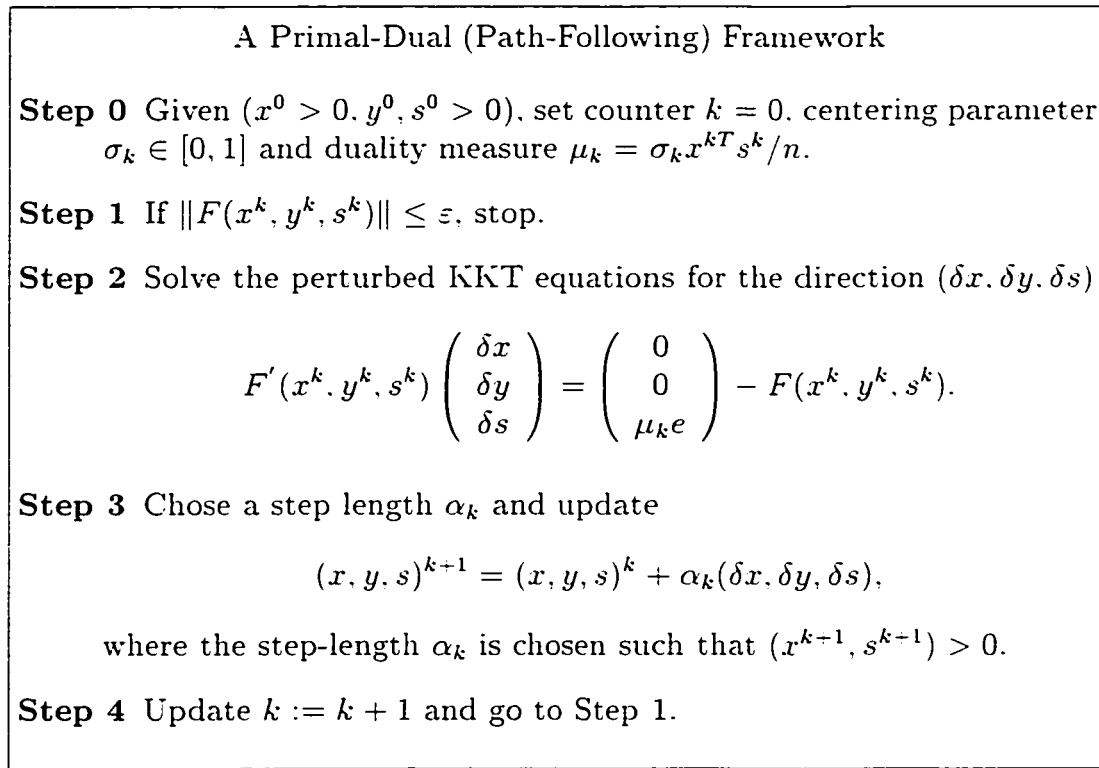


Figure 2.3 A Primal-Dual Interior Point Method Framework

2.5 Convex Programming

One of the most developed areas of study in nonlinear programming is convex programming, which is to find the minimum of a convex function in a convex set. All linear programming problems fall into this category. The MaxVE problem we are interested in is a specific convex programming problem with an unknown matrix variable. In this section, we give basic concepts in convex programming and its optimality condition – KKT conditions.

Definition 2.1 (*convex set*) A set $S \subset R^n$ is a *convex set* if given any $x, y \in S$, then $tx + (1 - t)y \in S$ for all $t \in [0, 1]$.

Definition 2.2 (*convex and concave functions*) Given a function $f : S \rightarrow R$ (where S is a convex set), f is *convex* if $x, y \in S$, then $f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y)$ for all $t \in [0, 1]$. If the inequality is strict when $x \neq y$, the function f is a *strictly convex function*. A function f is *concave* if $-f$ is convex.

If the domain of a function f is an open convex set and f is twice continuously differentiable with a positive semidefinite Hessian on the domain, then f is a convex function. Moreover, f is strictly convex if its Hessian is positive definite on the domain. A nice property of the convex programming problem is that it has a unique optimal value. Therefore any local optimizer of a convex programming problem is also a global optimizer.

Convex programming problems include a special case of the constrained minimization problem which has the following features:

1. the objective function is convex;
2. the equality constraints are linear;

3. the inequality constraint functions are convex in the format $g_i(x) \leq 0$.

The last two conditions are equivalent to the requirement that the feasible region defined by the constraints is a convex set.

The general convex programming problem can be written as

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, 2, \dots, m \\ & x \in S \end{aligned} \tag{2.8}$$

where $f(x)$ and g_i are convex functions and the subset $S \in R^n$ is a convex set. Similar to linear programming, $f(x)$ is called the *objective function* and the function inequalities $g_i(x) \leq 0$ are called *inequalities constraints* for (2.8). If $x \in S$ satisfies all the inequality constraints, x is called a *feasible point*. The set of all feasible points of the problem $\mathcal{F} = \{x \in S : g_i(x) \leq 0, i = 1, \dots, m\}$ consists the *feasible region* for (2.8), which is a convex set. If there exists feasible point $x_0 \in \mathcal{F}$ such that $g_i(x_0) < 0, i = 1, \dots, m$, the convex feasible region \mathcal{F} is said to satisfy *Slater's constraint qualification*. If x^* is a feasible point of (2.8) such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{F}$ for (2.8), then x^* is called the *solution* (or *minimizer*) of (2.8).

The Lagrangian function of a convex program is defined by

$$L(x, \lambda) = f(x) + \sum_{i=1}^m \lambda_i g_i(x)$$

for $x \in S$ and $\lambda \geq 0$. The vector $\lambda \in R^m$ is called the *Lagrangian multiplier*. The gradient of the Lagrangian function with respect to x is as follows:

$$\nabla_x L(x, \lambda) = \nabla f(x) + \sum_{i=1}^m \lambda_i \nabla g_i(x).$$

To solve optimization problem, it is crucial to know how to test whether a point is an optimal solution or not. The corresponding conditions are so called *optimal-*

ity conditions. For convex programming problem, its optimality conditions are the Karush-Kuhn-Tucker (KKT) conditions.

Suppose that a convex programming problem satisfies the Slater's constraint qualification and that its objective function $f(x)$ and inequality constraints $g_i(x)$ have continuous first partial derivatives on the set S for (2.8). If x^* is feasible for (2.8) and is an interior point of the set S , then x^* is a minimizer (or solution) of (2.8) if and only if there exists $\lambda^* \in R^m$ such that the following first-order necessary and sufficient conditions, i.e. KKT conditions, hold:

$$\nabla_x L(x, \lambda) = 0,$$

$$\Lambda^* g(x^*) = 0.$$

$$\lambda^* \geq 0.$$

where Λ^* denotes the diagonal matrix $\text{Diag}(\lambda^*)$ and $g(x) \in R^m$ is a vector whose i -th element is $g_i(x)$. In the next chapter on ellipsoidal approximation, we illustrate how the KKT conditions for convex programming problems can be applied to help locate solutions of the problem.

Chapter 3

Ellipsoidal Approximation of Polytopes

3.1 Introduction

The ellipsoidal approximation of polytopes is an important problem in its own right while it is also a basic subroutine in a number of algorithms for different problems. One example is that Lenstra's algorithm for the integer programming feasibility problem [28, 30] uses the ellipsoidal approximation of polytopes as a subroutine.

Consider a full-dimensional polytope $\mathcal{P} \in \mathbb{R}^n$ defined by m linear inequalities. For brevity, we will call the problem of finding the maximum volume ellipsoid inscribing \mathcal{P} the MaxVE problem. The MaxVE problem has its root in the rounding of convex bodies in \mathbb{R}^n . One of the earliest studies was done by F. John [17]. In particular, John's results imply that once the maximum-volume, inscribing ellipsoid \mathcal{E} is found in \mathcal{P} , then $\mathcal{E} \subset \mathcal{P} \subset n\mathcal{E}$, where $n\mathcal{E}$ is the ellipsoid resulting from dilating \mathcal{E} by a factor n about its center. Such a pair of ellipsoids is also called a *Löwner-John pair* of \mathcal{P} . That is, \mathcal{E} provides an n -bounding for \mathcal{P} . Moreover, if \mathcal{P} is centrally symmetric around the origin, then the rounding factor can be reduced to \sqrt{n} .

Ellipsoids have good geometric and computational properties that make them much easier to handle, both theoretically and computationally, than polytopes. For example, the global minimum of any quadratic in an ellipsoid can be located in polynomial time, while finding such a global minimum in a polytope is generally an NP-hard problem. For many problems a fruitful and effective approach is to use ellipsoids to approximate polytopes in various theoretic and algorithmic settings. A celebrated example is Khachiyan's ellipsoid method [23] – the first polynomial-time

algorithm for linear programming. Other applications include optimal design [44, 46], computational geometry (for example, [50]) and algorithm construction (for example, [14] and [45]).

Recently, several randomized polynomial-time algorithms ([11, 20, 32], for example) have been proposed for approximating the volume of convex bodies (computing the volume itself is NP-hard). In the case of a polytope, these algorithms require approximating the polytope by an ellipsoid.

It is known that the rounding of a polytope can be accomplished by the (shallow-cut) ellipsoid method in polynomial time (see, for example, [42, 14]). It is also known, however, that the ellipsoid method is not a practically efficient algorithm. A number of interior-point algorithms have been proposed in recent years for the maximum volume ellipsoid problems, for example, by Nesterov and Nemirovskii [34], Khachiyan and Todd [25] (also see [24] for a related problem), Nemirovskii [33], and Anstreicher [3].

Nesterov and Nemirovskii [34] constructed a three-stage barrier method for finding an ϵ -optimal ellipsoid \mathcal{E} such that its volume $\text{Vol}(\mathcal{E}) \geq \text{Vol}(\mathcal{E}^*)e^{-\epsilon}$, where \mathcal{E}^* is the maximum volume ellipsoid inscribing \mathcal{P} and $\epsilon \in (0, 1)$, with the complexity estimate $O(m^{2.5}(n^2 + m) \ln(\frac{mR}{\epsilon}))$ where m is the number of constraints and R is a priori known ratio of radius of the two concentric balls, the larger ball containing the given polytope \mathcal{P} and the smaller one being contained in \mathcal{P} . The term n^2 comes from the requirement of solving linear systems involving an $n \times n$ matrix-valued variable.

Khachiyan and Todd [25] proposed an algorithm that attains the complexity estimate of $O(m^{3.5} \ln(\frac{mR}{\epsilon}) \ln(\frac{n \ln R}{\epsilon}))$. The algorithm applies the basic barrier method to a small number of subproblems and only requires solving linear systems of $n + m$ equations to compute involved Newton directions. In their formulation the matrix-valued variable is explicitly treated as dependent on another vector-valued variable during the solution of Newton linear systems.

Nemirovskii [33] showed that the maximum volume ellipsoid problem can be reformulated as a saddle-point problem of $m + n$ variables and be solved by a path-following method for approximating saddle points of a sequence of self-concordant convex-concave functions as defined in [33]. Nemirovskii proved that the complexity of the algorithm is $O(m^{3.5} \ln(\frac{mR}{\epsilon}))$.

Most recently, Anstreicher [3] proposed an algorithm that uses key ideas of Khachiyan and Todd [25] but avoids solving the subproblems required in the Khachiyan and Todd algorithm. This way, Anstreicher's algorithm attains the complexity estimate of $O(m^{3.5} \ln(\frac{mR}{\epsilon}))$, which is the same as in [33]. Anstreicher also showed that computing an approximate analytic center of the polytope can reduce the complexity to $O((mn^2 + m^{1.5}n) \ln(R) + m^{3.5} \ln(\frac{m}{\epsilon}))$.

In addition, Vandenberghe, Boyd and Wu [48] proposed an algorithm for the class of problems called MAXDET problems to which the MaxVE problem belongs. However, their algorithm does not take into account the special structure of the MaxVE problem.

All the aforementioned works on the ellipsoidal approximation are primarily concerned with the complexity issues and the proposed algorithms are theoretical in nature. On the contrary, the objective of our current study is to construct or identify a numerically efficient and stable algorithm for solving general MaxVE problems. Our study is not aimed at solving very large-scale problems, so we will not consider aspects of exploiting sparsity and other special structures that may be present in the polytope-defining inequalities.

Since for many convex programs, primal-dual interior-point algorithms have proven to be superior in practice than either primal or dual algorithms, we will mainly investigate primal-dual type algorithms, though we will also consider particular primal algorithms for the purpose of comparison.

Two features are common in all the known interior-point algorithms for solving the MaxVE problem. First, they are iterative in nature. Second, they require solving a linear system at each iteration to obtain an update. Hence, in judging the practical efficiency of an algorithm, we must consider two key factors: (i) how many iterations the algorithm typically requires in practice for obtaining an approximate solution of a certain quality; and (ii) how expensive it is to solve the relevant linear system at each iteration. Besides efficiency, another important consideration is the robustness of the algorithm. The robustness of an iterative algorithm is often determined by the numerical stability of the solution procedure for linear systems that has to be invoked at every iteration.

In most primal-dual algorithms for linear programming or semidefinite programming, at each iteration one solves a large linear system by reducing it to a smaller Schur complement system through a block Gaussian elimination. Moreover, the coefficient matrix in the Schur complement system is often positive definite. This procedure has proven to be efficient and at the same time adequately stable. Likewise, in this chapter we will try to identify primal-dual algorithms for which the corresponding linear systems can be reduced by block Gauss elimination to a well-behaved Schur complement system.

The chapter is organized as follows. In Section 2 we describe the formulation of the MaxVE problem. We introduce some primal-dual type interior-point algorithm in Section 3 and give related theoretical results in Section 4. We summarize the Khachiyan and Todd algorithm and our modification in Section 5. Numerical comparative results on these three algorithms are presented in Section 6. Finally, we offer some concluding remarks in Section 7.

Notation For This Chapter

We now introduce some notation. For any given vector $v \in \mathbb{R}^p$, we denote the $p \times p$ diagonal matrix with v on its diagonal either by $\text{Diag}(v)$, or by its upper-case letter V whenever no confusion can occur. On the other hand, for a square matrix M , $\text{diag}(M)$ is the vector formed by the diagonal of M . The Hadamard product is represented by the small circle “ \circ ”. Unless specified otherwise, superscripts for vectors and subscripts for scalars, that are not elements of a matrix, are iteration counts. For a vector v , inequalities of the form $v > \alpha$ are interpreted as component-wise where α can be a scalar or a vector of the same size. For symmetric matrices, $A \succ B$, or equivalently $A - B \succ 0$, means that $A - B$ is positive definite. We use \mathbb{R}_+^m and \mathbb{R}_{++}^m to represent the nonnegative and positive orthants in \mathbb{R}^m , respectively. The notation \mathcal{S}_{++}^n represents the subspace of all symmetric positive definite matrices in $\mathbb{R}^{n \times n}$. For a set \mathcal{W} in \mathbb{R}^m , we denote its closure by $\text{col}(\mathcal{W})$. Finally, by default $\|\cdot\|$ represents the Euclidean norm unless otherwise specified.

3.2 The Maximum Volume Ellipsoid Problem

Consider a polytope \mathcal{P} in \mathbb{R}^n given by

$$\mathcal{P} = \{v \in \mathbb{R}^n : Av \leq b\}, \quad (3.1)$$

where $A \in \mathbb{R}^{m \times n}$, $m > n$ and $b \in \mathbb{R}^m$. Recall that by definition, a polytope is a bounded polyhedron. For convenience of discussion, we will make the following two assumptions throughout the chapter:

- A1. The matrix A has full rank n and contains no zero-rows.
- A2. There exists a strictly interior point $\bar{v} \in \mathcal{P}$ satisfying $A\bar{v} < b$.

Given a center $x \in \mathbb{R}^n$ and a nonsingular scaling matrix $E \in \mathbb{R}^{n \times n}$, an ellipsoid in \mathbb{R}^n centered at x can be defined as

$$\mathcal{E}(x, E) = \{v \in \mathbb{R}^n : (v - x)^T (EE^T)^{-1} (v - x) \leq 1\};$$

or equivalently,

$$\mathcal{E}(x, E) = \{v \in \mathbb{R}^n : v = x + Es \text{ and } \|s\| \leq 1\}, \quad (3.2)$$

where $\|\cdot\|$ is the Euclidean norm in \mathbb{R}^n . Clearly, the shape of the ellipsoid is uniquely determined by the symmetric positive definite matrix EE^T , but not uniquely by E since the same ellipsoid can also be generated by EQ for any orthogonal matrix $Q \in \mathbb{R}^{n \times n}$. Without loss of generality, we can assume that E itself is symmetric positive definite. Therefore, the ellipsoid $\mathcal{E}(x, E)$ is uniquely determined by the center x and the scaling matrix $E \in S_{++}^n$.

It is easy to see that the ellipsoid $\mathcal{E}(x, E)$ is contained in \mathcal{P} if and only if

$$\sup_{\|s\|=1} a_i^T (x + Es) \leq b_i, \quad i = 1, \dots, m$$

where a_i^T is the i -th row of A ; or equivalently

$$a_i^T x + \|Ea_i\| \leq b_i, \quad i = 1, \dots, m.$$

Introducing the notation

$$h(E) = (\|Ea_1\|, \dots, \|Ea_m\|)^T \in \mathbb{R}^m, \quad (3.3)$$

we have

$$\mathcal{E}(x, E) \subset \mathcal{P} \iff b - Ax - h(E) \geq 0. \quad (3.4)$$

Let V_n be the volume of the n -dimensional unit ball, then the volume of the ellipsoid $\mathcal{E}(x, E)$ defined by (3.2) is

$$\text{Vol}(\mathcal{E}) \equiv V_n \det E.$$

It is evident that $\mathcal{E}(x^*, E^*)$ is the maximum-volume ellipsoid contained in \mathcal{P} if and only if $(x^*, E^*) \in \mathbb{R}^n \times \mathcal{S}_{++}^n$ solves the following optimization problem:

$$\begin{aligned} \min \quad & -\log \det E \\ \text{s.t.} \quad & b - Ax - h(E) \geq 0 \\ & (E \succ 0) \end{aligned} \tag{3.5}$$

where $E \succ 0$ means that E is symmetric positive definite. It is well known that the optimization problem (3.5) is a convex program with a unique pair of solution $(x^*, E^*) \in \mathbb{R}^n \times \mathcal{S}_{++}^n$; and this solution is uniquely determined by the first-order optimality, or Karush-Kuhn-Tucker (KKT), conditions for the problem which can be derived as follows.

The Lagrangian function of the convex program (3.5) is

$$L(x, E, u) = -\log \det E - u^T(b - Ax - h(E))$$

where $u \in \mathbb{R}^m$ is the vector of Lagrange multipliers. The KKT conditions consist of the equations $\nabla_x L = 0$, $\nabla_E L = 0$, feasibility and complementarity. Using the differentiation formulas

$$\nabla[\log \det E] = E^{-1} \quad \text{and} \quad \nabla h_i(E) = \frac{E a_i a_i^T + a_i a_i^T E}{2h_i(E)},$$

and introducing the notation $U := \text{Diag}(u)$ and

$$Y \equiv Y(E, u) := \text{Diag}(h(E))^{-1} U, \tag{3.6}$$

we can write the KKT conditions as

$$A^T u = 0. \tag{3.7a}$$

$$E^{-1} - [E(A^T Y A) + (A^T Y A)E]/2 = 0, \tag{3.7b}$$

$$z - (b - Ax - h(E)) = 0. \tag{3.7c}$$

$$U z = 0, \tag{3.7d}$$

$$u, z \geq 0, \tag{3.7e}$$

where $E \succ 0$, and z is a slack variable.

3.3 Formulations and Primal-Dual Algorithms

In this section, we propose formulations and algorithms for effectively solving the MaxVE problem in practice. In constructing practically efficient algorithms, we consider the following three guidelines:

1. the algorithms should not carry the matrix-valued variable E as a completely independent variable because it would require too much computation;
2. the algorithms should be primal-dual algorithms because of their proven practical efficiency in numerous cases;
3. the algorithms should have theoretical guarantees to be well defined and well behaved.

The first objective above can be achieved by eliminating the matrix variable E . The elimination may occur either at the beginning of a formulation, or at the time of solving linear systems during iterations. In this chapter, we will take the former approach.

3.3.1 Formulations without Matrix Variable

In this subsection, we describe three new formulations, recently proposed by Zhang in [52], for the MaxVE problem which are free of the matrix variable E . The key idea of these formulations is to eliminate the matrix-valued variable E from the system by solving the equation (3.7b) for E . As can be verified easily, a solution to (3.7b) is

$$E(y) = (A^T Y A)^{-1/2}, \quad (3.8)$$

where $Y = \text{Diag}(y)$ and Y is defined in (3.6). We will later demonstrate that this solution is unique in \mathcal{S}_{++}^n . Upon the substitution of $E(y)$ into the definition of $h(y)$ (recall that $h_i(E) = \|Ea_i\|$), the vector $h(E)$ becomes a function of y that we will denote, with a slight abuse of notation, as $h(y)$; namely.

$$h(y) \equiv h(E(y)). \quad (3.9)$$

After substituting (3.8) and (3.9) into the KKT system, deleting (3.7b) and adding (3.6) written in a different form, i.e.,

$$u = g(y) := Yh(y), \quad (3.10)$$

the author of [52] obtained the following system:

$$F_0(x, y, u, z) = 0, \quad y, u, z \geq 0, \quad (3.11)$$

where $x \in \mathbb{R}^n$, $y, u, z \in \mathbb{R}^m$, and the function $F_0 : \mathbb{R}^{n+3m} \rightarrow \mathbb{R}^{n+3m}$ is

$$F_0(x, y, u, z) = \begin{bmatrix} A^T u \\ Ax + h(y) + z - b \\ u - g(y) \\ Uz \end{bmatrix}. \quad (3.12)$$

Moreover, it was proposed in [52] to eliminate the variable u from the above system using the equation $u = g(y)$ in (3.10). The resulting system is

$$F_1(x, y, z) = 0, \quad y, z \geq 0, \quad (3.13)$$

where the function $F_1 : \mathbb{R}^{n+2m} \rightarrow \mathbb{R}^{n+2m}$ is

$$F_1(x, y, z) = \begin{bmatrix} A^T g(y) \\ Ax + h(y) + z - b \\ Zg(y) \end{bmatrix}. \quad (3.14)$$

In (3.12) and (3.14), we have used the notation $U = \text{Diag}(u)$ and $Z = \text{Diag}(z)$, respectively.

In addition, the complementarity conditions $Uz = 0$ are clearly equivalent to the conditions $Yz = 0$ because $U = Y\text{Diag}(h(y))$ and $h(y) > 0$ at the solution. Based on this observation, the third system was proposed as following:

$$F_2(x, y, z) = 0, \quad y, z \geq 0. \quad (3.15)$$

where the function $F_2 : \mathbb{R}^{n+2m} \rightarrow \mathbb{R}^{n+2m}$ is

$$F_2(x, y, z) = \begin{bmatrix} A^T g(y) \\ Ax + h(y) + z - b \\ Yz \end{bmatrix}. \quad (3.16)$$

The three systems (3.11), (3.13) and (3.15) are all free of the matrix-valued variable E , which will form the bases for our algorithm construction. Other systems, derived in [52] have been found to be less satisfactory. However, in obtaining them we have applied nonlinear transformations whose properties need to be investigated. A most important question is whether or not these transformations preserve the uniqueness of solutions. We will answer this question and others in a subsequent section.

3.3.2 Primal-Dual Algorithmic Framework

The primal-dual algorithms to be proposed can be motivated from the view of the damped Newton method applied to the so-called perturbed complementarity conditions. Another useful perspective is to view them as path-following algorithms. In this construction, one replaces the zero right-hand-side of relevant complementarity conditions by μw^0 , where $\mu > 0$ and $w^0 \in \mathbb{R}_{++}^m$, and applies the Newton method to the

resulting “perturbed” system while decreasing the parameter μ to zero. Specifically, the perturbed systems for (3.13) and (3.15) have the form

$$F(x, y, z) = \mu \begin{bmatrix} 0 \\ 0 \\ w \end{bmatrix}, \quad y, z > 0. \quad (3.17)$$

where F can be either F_1 or F_2 , and for some $w^0 \in \mathbb{R}_{+-}^m$

$$w = \mu w^0, \quad \mu > 0.$$

Normally, one chooses $w^0 = e$ where e is the vector of all ones.

We will prove later that each of the perturbed systems have a unique solution for every $\mu > 0$, and as $\mu \rightarrow 0$ the corresponding solutions will converge to the (same) solution of the unperturbed systems from which the solution to the MaxVE problem can be easily constructed.

We now present our primal-dual interior-point algorithmic framework for the systems (3.13) and (3.15). The framework for the system (3.11) would be the same except that an extra variable $u \in \mathbb{R}^m$ is present. In the rest of this chapter, we will concentrate only on the formulations (3.13) and (3.15) but omit (3.11) because, being so closely related to (3.13), (3.11) shares almost identical theoretical properties with (3.13), while in our tests it seems to produce algorithms with inferior performance to that of their counterparts based on (3.13) and (3.15).

Algorithm 1 (*Primal-Dual Interior-Point Algorithm*) Given $x^0 \in \mathcal{P}$,

$w^0, y^0, z^0 \in \mathbb{R}_{++}^m$, set $k = 0$.

Step 1. Choose $\sigma_k \in (0, 1)$, set μ_k to $\sigma_k \frac{g(y^k)^T z^k}{m}$ for $F = F_1$ or to $\sigma_k \frac{(y^k)^T z^k}{m}$ for $F = F_2$.

Step 2. Solve for (dx, dy, dz) from

$$F'(x^k, y^k, z^k) \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \mu_k \begin{bmatrix} 0 \\ 0 \\ e \end{bmatrix} - F(x^k, y^k, z^k). \quad (3.18)$$

Step 3. Choose a step-length $\alpha_k \in (0, 1]$ and update

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha_k(dx, dy, dz),$$

such that $x^{k+1} \in \mathcal{P}$, $y^{k+1} > 0$ and $z^{k+1} > 0$.

Step 4. If $\|F(x^{k+1}, y^{k+1}, z^{k+1})\| \leq \epsilon$, stop; else increment k and go to Step 1.

In addition to the initial guesses, this algorithmic framework has two essential parameters, σ_k and α_k , that need to be specified at each iteration. The main computation required is to solve the linear system (3.18) at every iteration.

When $F = F_1$, the coefficient matrix in the linear system (3.18), i.e., the Jacobian matrix of $F_1(x, y, z)$, is of the form

$$F'_1(x, y, z) = \begin{bmatrix} 0 & A^T g'(y) & 0 \\ A & h'(y) & I \\ 0 & Zg'(y) & \text{Diag}(g(y)) \end{bmatrix}. \quad (3.19)$$

where $g'(y)$ and $h'(y)$ are the Jacobian matrices of $g(y)$ and $h(y)$, respectively. A direct differentiation shows that

$$g'(y) = \text{Diag}(h(y)) + Yh'(y), \quad (3.20)$$

and (see [52])

$$h'(y) = -\text{Diag}(2h(y))^{-1}[Q(y) \circ Q(y)], \quad (3.21)$$

where

$$Q(y) = A(A^T Y A)^{-1} A^T. \quad (3.22)$$

It is worth noting that $Y^{1/2}Q(y)Y^{1/2}$ is an orthogonal projection matrix.

On the other hand, when $F = F_2$ we have

$$F'_2(x, y, z) = \begin{bmatrix} 0 & A^T g'(y) & 0 \\ A & h'(y) & I \\ 0 & Z & Y \end{bmatrix}. \quad (3.23)$$

An efficient way to solve the linear system (3.18) is the following block Gaussian elimination procedure: first eliminating dz and dy , then solving for dx , finally computing dy and dz by back substitutions. We now formally describe the procedure for $F = F_1$. To simplify notation, we define three $m \times m$ matrices:

$$H \equiv H(y) := \text{Diag}(h(y)), \quad N \equiv N(y) := g'(y), \quad (3.24)$$

and

$$M_1 \equiv M_1(y, z) := -h'(y) + [YH(y)]^{-1}ZN(y). \quad (3.25)$$

For now we will assume that M_1 is nonsingular, and we will prove this fact later.

The aforementioned block Gaussian elimination reduces $F'_1(x, y, z)$ into a lower triangular matrix, which is equivalent to, when $F = F_1$, pre-multiplying the equation (3.18) by the upper triangular elimination matrix

$$T_1(y, z) = \begin{bmatrix} I & A^T N M_1^{-1} & -A^T N M_1^{-1} (YH)^{-1} \\ 0 & I & -(YH)^{-1} \\ 0 & 0 & I \end{bmatrix}.$$

It is straightforward to verify that

$$T_1(y, z)F'_1(x, y, z) = \begin{bmatrix} A^T N M_1^{-1} A & 0 & 0 \\ A & -M_1 & 0 \\ 0 & ZN & YH \end{bmatrix}, \quad (3.26)$$

and

$$T_1(y, z) \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} r_1 + A^T N M_1^{-1} (r_2 - (YH)^{-1} r_3) \\ r_2 - (YH)^{-1} r_3 \\ r_3 \end{bmatrix}, \quad (3.27)$$

for any vectors $r_1 \in \mathbb{R}^n$ and $r_2, r_3 \in \mathbb{R}^m$. Clearly, the linear system

$$F'_1(x, y, z) \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

is equivalent to the linear system where the coefficient matrix is the one in (3.26) and the right-hand side is that of (3.27). The linear system can be formally solved by the procedure:

$$dx = [A^T N M_1^{-1} A]^{-1} (r_1 + A^T N M_1^{-1} (r_2 - (YH)^{-1} r_3)), \quad (3.28a)$$

$$dy = -M_1^{-1} (r_2 - (YH)^{-1} r_3 - A dx), \quad (3.28b)$$

$$dz = (YH)^{-1} (r_3 - Z N dy). \quad (3.28c)$$

This solution procedure requires $\mathcal{O}(m^3)$ operations (recall that $m > n$), with the bulk of the computation involving the $m \times m$ matrix M_1 .

In a similar fashion, the linear system (3.18) corresponding to $F = F_2$ can be formally solved by the following procedure:

$$dx = [A^T N M_2^{-1} A]^{-1} (r_1 + A^T N M_2^{-1} (r_2 - Y^{-1} r_3)), \quad (3.29a)$$

$$dy = -M_2^{-1} (r_2 - Y^{-1} r_3 - A dx), \quad (3.29b)$$

$$dz = Y^{-1} (r_3 - Z dy), \quad (3.29c)$$

where

$$M_2 \equiv M_2(y, z) := -h'(y) + Y^{-1} Z. \quad (3.30)$$

This procedure also requires $\mathcal{O}(m^3)$ operations in terms of the order, but less linear algebra computation than does the procedure (3.28a)-(3.28c).

Of course, we still need to establish in theory that the proposed primal-dual algorithms are well-defined. To this end, we need to show that the matrix $F'_i(x, y, z)$ are nonsingular for any $y, z > 0$, and the matrices M_i and $A^T N M_i^{-1} A$ are also nonsingular for both $i = 1$ and $i = 2$. These results will be presented next.

3.4 Theoretical Results

In this section, we give theoretical results regarding the well-definedness of the proposed algorithms, the uniqueness of solution in our formulations, as well as the existence and convergence of solution paths. We note that the formulations introduced in the last section are obtained by applying some nonlinear transformations. Therefore we need to show that these nonlinear transformations preserve the uniqueness of solution. We also mention that when $F = F_2$, the system in (3.17) is not equivalent to the optimality conditions of a convex program. Hence, it is not evident that solution paths defined by (3.17) should always exist for $F = F_2$.

3.4.1 Well-Definedness of Algorithms

We will show in this subsection that the proposed primal-dual algorithmic framework and the solution procedures (3.28a)-(3.28c) and (3.29a)-(3.29c) are well-defined for both $F = F_1$ and $F = F_2$. (Following the same approach, one can also easily verify similar results for $F = F_0$.)

We recall that throughout the chapter we have assumed that A has full rank with no zero rows. The main results of this subsection is the following theorem.

Theorem 3.1 (*Non-singularity of Jacobian*) For any $y, z > 0$, the Jacobian matrices $F'_i(x, y, z)$ are nonsingular for $i = 1, 2$. Moreover, both the procedures (3.28a)-(3.28c) and (3.29a)-(3.29c) are well defined.

Proof The theorem follows directly from Lemma 3.3 below. \square

Now we prove the three technical results that will lead to the proof of Theorem 3.1.

Lemma 3.1 Let $P \in \mathbb{R}^{n \times n}$ be an orthogonal projection matrix, i.e., P satisfies $P^T = P$ and $P^2 = P$. Then the symmetric matrix

$$G_\gamma = \text{Diag}(\text{diag}(P)) - \gamma P \circ P \quad (3.31)$$

is positive semidefinite for any $\gamma \leq 1$. Moreover, if $\text{diag}(P) > 0$, then G_γ is positive definite for any $\gamma < 1$.

Proof We note that since P is symmetric positive semidefinite, so is $P \circ P$ (see for example [16]). For the first statement, it suffices to only consider $\gamma = 1$.

Let (λ, x) be a pair of the eigenvalue and the associated eigenvector of G_1 such that

$$x_k = \max_i |x_i| \equiv 1,$$

which can always be achieved by a proper scaling. The k -th equation in $G_1 x = \lambda x$ is

$$P_{kk}x_k - \sum_{j=1}^n P_{kj}^2 x_j = \lambda x_k.$$

Since $P^2 = P$ and $P^T = P$, implying that $P_{kk} = \sum_{j=1}^n P_{kj}^2$, and $x_k = 1$, we have

$$\lambda = \sum_{j=1}^n P_{kj}^2 (1 - x_j) \geq 0,$$

where the last inequality follows from the fact that $|x_j| \leq 1$ for all j . Hence we have proved that G_1 is positive semidefinite. Together with the identity

$$G_\gamma = G_1 + (1 - \gamma)P \circ P.$$

it implies that G_γ are positive semidefinite for all $\gamma \leq 1$ since both terms in the sum are positive semidefinite.

To prove the second statement, we assume that $\text{diag}(P) > 0$ and $\gamma < 1$. Then we write

$$G_\gamma = (1 - \gamma)\text{Diag}(\text{diag}(P)) + \gamma G_1.$$

which is clearly positive definite since the first term is positive definite and the second one is semidefinite. \square

Lemma 3.2 For any $y > 0$, the matrix $N(y) \equiv g'(y)$ is similar to a symmetric positive definite matrix, and thus is nonsingular.

Proof We first note $h(y) > 0$ whenever $y > 0$. In view of (3.24), (3.20) and (3.21),

$$\begin{aligned} N &= H - (2H)^{-1}Y[Q(y) \circ Q(y)] \\ &= H^{-1}(HYH - \frac{1}{2}Y[Q(y) \circ Q(y)]Y)Y^{-1} \\ &= [H^{-1/2}Y^{1/2}] \left([HY]^{-1/2}G[YH]^{-1/2} \right) [H^{-1/2}Y^{1/2}]^{-1} \end{aligned}$$

where

$$G := HYH - \frac{1}{2}Y[Q(y) \circ Q(y)]Y. \quad (3.32)$$

Therefore, N is similar to $[HY]^{-1/2}G[YH]^{-1/2}$, which is positive definite if and only if the matrix G is positive definite since both Y and H are positive diagonal matrices.

Recall that by our notation $Q(y) = A(A^T Y A)^{-1} A^T$, $H = \text{Diag}(h(y))$ and

$$h(y) \equiv h(E(y)) = (\text{diag}(Q(y)))^{1/2},$$

where the square root is taken element-wise. We have

$$HYH = \text{Diag}(\text{diag}(Q(y))Y = \text{Diag}(\text{diag}(Y^{1/2}Q(y)Y^{1/2})).$$

In addition, since $y_i Q_{ij}^2 y_j = (\sqrt{y_i} Q_{ij} \sqrt{y_j})^2$, we have

$$Y[Q(y) \circ Q(y)]Y = (Y^{1/2}Q(y)Y^{1/2}) \circ (Y^{1/2}Q(y)Y^{1/2}).$$

Therefore we can write

$$G = \text{Diag}(\text{diag}(P)) - \frac{1}{2}P \circ P,$$

where the matrix

$$P = Y^{1/2}Q(y)Y^{1/2} = Y^{1/2}A(A^T Y A)^{-1}A^T Y^{1/2}$$

is an orthogonal projection matrix. Since the vector $y > 0$ and the matrix A has no zero rows, we have $\text{diag}(P) > 0$. It follows from Lemma 3.1 with $\gamma = 1/2$ that G is indeed positive definite. This completes the proof. \square

The relationships

$$N = H^{-1}GY^{-1} \quad \text{and} \quad N^{-1} = YG^{-1}H \tag{3.33}$$

that were used in the proof of Lemma 3.2 will be useful later.

Lemma 3.3 For $y, z > 0$, there hold the following statements:

1. the matrix M_1 is similar to a symmetric positive definite matrix, and $A^T N M_1^{-1} A$ is symmetric positive definite;
2. the matrix M_2 is similar to a symmetric positive definite matrix, and $A^T N M_2^{-1} A$ is nonsingular.

Proof To prove the first statement, it suffices to prove that the matrix $M_1 N^{-1}$ is symmetric positive definite. Using the definitions of M_1 , N and the formula for g' , see (3.25), (3.24) and (3.20), respectively, and the relationships (3.33), we have $h' = Y^{-1}(N - H)$ and

$$\begin{aligned}
 M_1 N^{-1} &= ((YH)^{-1} Z N - h') N^{-1} \\
 &= (YH)^{-1} Z - Y^{-1}(N - H) N^{-1} \\
 &= (YH)^{-1} Z - Y^{-1} + Y^{-1} H N^{-1} \\
 &= (YH)^{-1} Z - Y^{-1} + Y^{-1} H (Y G^{-1} H) \\
 &= (YH)^{-1} Z - Y^{-1} + H G^{-1} H \\
 &= (YH)^{-1} Z + H (G^{-1} - (HYH)^{-1}) H.
 \end{aligned}$$

Then it suffices to show that $G^{-1} - (HYH)^{-1}$ is symmetric positive definite since H , Y and Z are all positive diagonal matrices. While the symmetry is obvious, the positive definiteness follows from the fact that G equals HYH minus a positive semidefinite matrix (see (3.32)), hence $G \prec YHY$ and $G^{-1} \succ (YHY)^{-1}$.

To prove the second statement, we use the formula for $h'(y)$ in (3.21), to obtain

$$M_2 = Y^{-1} Z - h' = H^{-1} (HY^{-1} Z + \frac{1}{2} Q \circ Q),$$

which is the product of two symmetric positive definite matrices, implying that M_2 is similar to a symmetric positive definite matrix. Since both M_2 and N are nonsingular, so is $ANM_2^{-1}A$. This completes the proof. \square

3.4.2 Uniqueness of Solution

Since we have utilized nonlinear transformations in the elimination of variables $E = E(y)$ and $u = g(y)$ from the KKT system (3.7a)-(3.7d), we need to establish a rigorous

equivalence of our formulations (3.13) and (3.15) to the original KKT system. The main result is the following.

Theorem 3.2 (*Uniqueness of Solution*) The systems (3.13) and (3.15) have the same, unique solution (x^*, y^*, z^*) such that $y^*, z^* \geq 0$. Moreover, let $u^* = g(y^*)$ and $E^* = E(y^*)$. Then (x^*, E^*, u^*, z^*) is the unique solution of the KKT conditions (3.7a)-(3.7e).

Proof The conclusions follow directly from Lemmas 3.4 and 3.5, and the uniqueness of the solution to the MaxVE problem. \square

We now prove the two technical lemmas.

Lemma 3.4 Let $C \in \mathcal{S}_{-+}^n$, then the matrix equation

$$X^{-1} = \frac{1}{2}(CX + XC) \quad (3.34)$$

has a unique solution $X^* = C^{-1,2}$ in \mathcal{S}_{++}^n . Moreover, the mapping: $C \rightarrow X^*$ defined implicitly through (3.34) is homeomorphic between \mathcal{S}_{-+}^n and itself.

Proof One can easily verify that both X^* and $-X^*$ are solutions to (3.34). This implies that the matrix equation (3.34) does not in general have a unique solution in $\mathcal{R}^{n \times n}$.

Suppose that $\hat{X} \in \mathcal{S}_{++}^n$ is a solution to the equation (3.34) and V is an orthogonal matrix that diagonalizes \hat{X} , i.e., $V^T \hat{X} V = \Sigma$ where Σ is a positive diagonal matrix. Pre-multiplying both side of the equation (3.34) by V^T and post-multiplying them by V , we obtain

$$\Sigma^{-1} = \frac{1}{2}(D\Sigma + \Sigma D),$$

where $D = V^T C V$. Comparing the elements on both sides, we have

$$\frac{1}{2} D_{ij} (\Sigma_{ii} + \Sigma_{jj}) = \begin{cases} 0, & i \neq j, \\ 1/\Sigma_{ii}, & i = j. \end{cases}$$

Since $\text{diag}(\Sigma) > 0$, we must have (i) $D_{ij} = 0$ for $i \neq j$ and (ii) $\Sigma_{ii} = D_{ii}^{-1/2}$. The first relationship says that $D = V^T C V$ is also diagonal. The second relationship says that $\Sigma = D^{-1/2}$, that is, $\hat{X} = C^{-1/2} \equiv X^*$. Consequently, X^* is the only solution of the equation (3.34) in \mathcal{S}_{++}^n .

The last statement of the lemma is evident in view of the explicit relationships $X^* = C^{-1/2}$ and $C = (X^*)^{-2}$. \square

Lemma 3.5 Let $g(y) \equiv Yh(y)$. Then the mapping $g : \mathcal{R}_{++}^m \rightarrow \mathcal{R}_{++}^m$ is homeomorphic between \mathcal{R}_{++}^m and the range of g , $g(\mathcal{R}_{++}^m) \subset \mathcal{R}_{++}^m$.

Proof It is straightforward to verify that the function $g(y)$ is continuously differentiable in \mathcal{R}_{++}^m whose derivative is represented by the matrix $g'(y) \equiv N(y)$. By Lemma 3.2, $N(y)$ is nonsingular in \mathcal{R}_{++}^m . With these properties, the lemma is a direct consequence of the inverse function theorem. \square

3.4.3 Existence and Convergence of Paths

To justify our algorithms as the path-following type, we need to show that (i) the perturbed system (3.17) with either $F = F_1$ or $F = F_2$ permits a unique solution for any given $w^0 \in \mathcal{R}_{++}^m$ and each $\mu > 0$, hence the solution set forms a path; and (ii) as $\mu \rightarrow 0$ the path converges to the unique solution of the unperturbed system. Although it is straightforward to establish these results in the case of $F = F_1$, it is much more involved in the case of $F = F_2$ since the perturbed system (3.17) for $F = F_2$ does not correspond to the optimality conditions of a convex program.

Following the conventional terminology in the literature of interior-point methods, we will refer the collection of solution to the system (3.17) for $w^0 = e$ and $\mu > 0$ as the *central path* of the system, where $e \in \mathbb{R}^m$ is the vector of all ones. Our analysis in this subsection applies not only to the central path but also to the so-called weighted paths where $w^0 > 0$ is not equal to e .

The existence of a path for $F = F_1$ follows a standard argument as given below.

Proposition 1 (*Existence and Convergence of Path for $F = F_1$*) For any $w^0 \in \mathbb{R}_{++}^m$ and $\mu > 0$, the system (3.17) with $F = F_1$ have a unique solution $(x(\mu), y(\mu), z(\mu))$ such that $y(\mu), z(\mu) > 0$. Moreover,

$$\lim_{\mu \rightarrow 0} (x(\mu), y(\mu), z(\mu)) = (x^*, y^*, z^*),$$

where (x^*, y^*, z^*) is the solution of (3.13).

Proof The proof follows from a standard argument which we will outline as follows. It is well-known that the system of the “perturbed” KKT (PKKT) conditions:

$$A^T u = 0, \quad (3.35a)$$

$$E^{-1} - [E(A^T Y A) + (A^T Y A)E]/2 = 0, \quad (3.35b)$$

$$z - (b - Ax - h(E)) = 0, \quad (3.35c)$$

$$Uz = w, \quad (3.35d)$$

$$u, z > 0. \quad (3.35e)$$

have a unique solution for any $w > 0$, where Y is defined in (3.6), because they are equivalent to that the gradient of the following barrier function $B_w(x, E)$.

$$B_w(x, E) = -\log \det(E) - \sum_{i=1}^m w_i \log (b_i - a_i^T x - h_i(E)), \quad (3.36)$$

equals zero. This barrier function is strongly convex and has a unique stationary point $(x(\mu), E(\mu), u(\mu), z(\mu))$ corresponding to $w = \mu w^0$ for a fixed $w^0 \in \mathbb{R}_{++}^m$ and any

$\mu > 0$. It is well known that $(x(\mu), E(\mu), u(\mu), z(\mu))$ converges to the unique solution (x^*, E^*, u^*, z^*) of the (unperturbed) KKT system as $\mu \rightarrow 0$. Due to the homeomorphic relationships between the PKKT conditions and the conditions in (3.17) with $F = F_1$, we know that $(x(\mu), y(\mu), z(\mu))$, where $y(\mu) = \text{Diag}(h(E(\mu)))^{-1}u(\mu)$, is also the unique solution of (3.17) with $F = F_1$. Moreover, the path $\{(x(\mu), y(\mu), z(\mu)) : \mu > 0\}$ converges to (x^*, y^*, z^*) where $y^* = \text{Diag}(h(E^*))^{-1}u^*$. \square

We now consider the existence of solution to the system (3.17) when $F = F_2$; that is, the existence of solution to the system

$$A^T g(y) = 0, \quad (3.37a)$$

$$Ax + h(y) + z - b = 0, \quad (3.37b)$$

$$Yz = w, \quad (3.37c)$$

$$y, z > 0, \quad (3.37d)$$

where $g(y)$ is defined as in (3.10). The situation here is more complicated because this system is no longer equivalent to the PKKT conditions (3.35a)-(3.35e) when $w > 0$, even though they are equivalent when $w = 0$. As such, we can no longer use the standard argument used in the proof of Proposition 1, in contrary to the case when $F = F_1$. The question is whether or not the following statement holds:

$$\{0 \in \mathbb{R}^n\} \times \{0 \in \mathbb{R}^m\} \times \mathbb{R}_{++}^m \subset \mathcal{R}(F_2),$$

where

$$\mathcal{R}(F_2) := F_2(\mathbb{R}^n \times \mathbb{R}_{++}^m \times \mathbb{R}_{++}^m)$$

is the range of the function F_2 corresponding to the domain $\mathbb{R}^n \times \mathbb{R}_{++}^m \times \mathbb{R}_{++}^m$. In particular, we want to know if the vectors $(0, 0, \mu e)$ for $\mu > 0$ are in the range of F_2 ; in other words, whether a central path exists for the system (3.17) in the case of $F = F_2$.

The answer to the above question is affirmative and given in Theorem 3.3. To prove the theorem, it is necessary to establish a number of technical results. We start with the following proposition stating some useful facts.

Proposition 2 The following facts hold:

1. Both F_1 and F_2 are locally homeomorphic at any point $(x, y, z) \in \mathbb{R}^n \times \mathbb{R}_{++}^m \times \mathbb{R}_{++}^m$.
2. If $(\hat{x}, \hat{y}, \hat{z})$ is the solutions to the system (3.17) with $F = F_1$ and $w = \hat{w}$, then $(\hat{x}, \hat{y}, \hat{z})$ also satisfies (3.17) with $F = F_2$ (i.e., (3.37a)-(3.37d)) and $w = \text{Diag}(h(\hat{y}))^{-1} \hat{w}$.

If one were able to choose \hat{w} such that $\text{Diag}(h(\hat{y}))^{-1} \hat{w} = \mu e$, then he/she would find the point $(0, 0, \mu e)$ in the range of F_2 . However, since \hat{y} is dependent on \hat{w} , it is not clear whether or not such a vector \hat{w} exists, let alone how to find it. Nevertheless, we do find a form of points $(0, 0, w)$ that are in the range of F_2 .

Lemma 3.6 Let $x \in \mathbb{R}^n$, $E \in \mathcal{S}_{++}^n$ and $z \in \mathbb{R}_{++}^m$ satisfy the equation

$$Ax + h(E) + z = b. \quad (3.38)$$

Then there exists a constant $\gamma > 0$, independent of x , E and z , such that

$$\max(\|x\|, \|E\|, \|z\|) \leq \gamma.$$

Proof The equation (3.38) implies that $x \in \mathcal{P}$ where \mathcal{P} is the given polytope (bounded), hence such x 's must be uniformly bounded above. Consequently, $b - Ax$ for $x \in \mathcal{P}$ is also uniformly bounded above, which in turn implies that both z and $h(E)$ are uniformly bounded above because they are both nonnegative and they sum up to $b - Ax$. Since $h_i(E) = (a_i^T E^2 a_i)^{1/2}$ and, by our assumption, the set $\{a_1, a_2, \dots, a_m\}$

spans \mathbb{R}^n , the uniform boundedness of $h(E)$ implies that of E . This completes the proof. \square

Lemma 3.7 Let the barrier function $B_w(x, E)$ be defined as in (3.36). let \mathcal{W} be a bounded set with its closure $\text{col}(\mathcal{W})$ in $\mathbb{R}_{++}^m \cup \{0\}$. For any $w \in \mathbb{R}_{++}^m$, define

$$(x_w, E_w) := \arg \min B_w(x, E), \quad (3.39)$$

and for $w = 0 \in \mathbb{R}^m$ define $(x_w, E_w) := (x^*, E^*)$ as the solution of the MaxVE problem (3.5). Then

$$\beta_{\mathcal{W}} := \inf_{w \in \text{col}(\mathcal{W})} \{\log \det(E_w)\} > -\infty.$$

Proof Since the pair (x_w, E_w) , $E_w \succ 0$, is the unique minimizer of $B_w(x, E)$, there exists some $(u_w, z_w) \in \mathbb{R}_{++}^m \times \mathbb{R}_{++}^m$ such that together they satisfies (3.35a)-(3.35e). It is well-known that the quadruple (x_w, E_w, u_w, z_w) is a continuous function of w in \mathbb{R}_{++}^m and that (x_w, E_w, u_w, z_w) converges to (x^*, E^*, u^*, z^*) as w converges to 0 from the interior of \mathbb{R}_{++}^m . Hence, the composite function $\log \det(E_w)$ of w is a continuous function of w in $\mathbb{R}_{++}^m \cup \{0\}$ and must attain its maximum on the compact set $\text{col}(\mathcal{W}) \subset \mathbb{R}_{++}^m \cup \{0\}$. This proves the lemma. \square

Lemma 3.8 Let $\mathcal{R}(F_2)$ be the range of the function F_2 corresponding to the domain $\mathbb{R}^n \times \mathbb{R}_{++}^m \times \mathbb{R}_{++}^m$, and \mathcal{W} be a bounded set in \mathbb{R}_{++}^m such that its closure $\text{col}(\mathcal{W}) \subset \mathbb{R}_{++}^m \cup \{0\}$. Let

$$\{0 \in \mathbb{R}^n\} \times \{0 \in \mathbb{R}_{++}^m\} \times \mathcal{W} \subset \mathcal{R}(F_2),$$

and $(x(w), y(w), z(w))$ be the solution to (3.37a)-(3.37c) corresponding to $w \in \mathcal{W}$. Then the set $\{y(w) : w \in \mathcal{W}\}$ is bounded.

Proof The triple $(x(w), y(w), z(w))$ being the solution to (3.37a)-(3.37c) implies that the quadruple

$$(x_{w'}, E_{w'}, u_{w'}, z_{w'}) := (x(w), E(y(w)), g(y(w)), z(w))$$

is the solution to (3.35a)-(3.35e) with the right-hand side of (3.35d) being replaced by $w' = \text{Diag}(h(y(w)))w$. It is worth noting that $(x_{w'}, E_{w'})$ also satisfies (3.39) with $w = w'$. Evidently, we have

$$E_{w'} \equiv E(y(w)).$$

Define the set

$$\mathcal{W}' := \{w' : w' = \text{Diag}(h(y_w))w, w \in \mathcal{W}\},$$

which is bounded because both \mathcal{W} and the set of $\{h(y(w)) : w \in \mathcal{W}\}$ are bounded.

It follows from Lemma 3.6 that the set

$$\{E_{w'} : w' \in \mathcal{W}'\} \equiv \{E(y(w)) : w \in \mathcal{W}\}$$

is bounded. Hence, the eigenvalues of $E(y(w))$ are uniformly bounded above. On the other hand, Lemma 3.7 implies that

$$\det(E(y(w))) \geq \exp(\beta_{\mathcal{W}}) > 0.$$

As a result, the eigenvalues of $E(y(w))$ are also uniformly bounded away from zero in the set \mathcal{W} . Consequently, the components of $h(y(w))$ are uniformly bounded above and away from zero in the set \mathcal{W} because $h_i(y(w)) = (a_i^T E(y(w)) a_i)^{1/2}$ and the rows a_i^T of A are all nonzero for $i = 1, \dots, m$.

We note that the vector $\text{Diag}[h(y(w))]^2 y(w)$ is the diagonal of the orthogonal projection matrix $Y(w)^{1/2} A [A^T Y(w) A]^{-1} A^T Y(w)^{1/2}$ and therefore is component-wise bounded above by the unity; namely,

$$y_i(w) \leq \frac{1}{h_i(y(w))^2}, \quad i = 1, 2, \dots, m.$$

Since $h(y(w))$ is uniformly bounded away from zero for $w \in \mathcal{W}$, we conclude that $y(w)$ is uniformly bounded above for $w \in \mathcal{W}$. This completes the proof. \square

Lemma 3.9 Let $\mathcal{R}(F_2)$ be defined as in Lemma 3.8, then

$$\{0 \in \mathbb{R}^n\} \times \{0 \in \mathbb{R}^m\} \times \mathbb{R}_{++}^m \subset \mathcal{R}(F_2).$$

Proof From the second statement of Proposition 2. we have known that there exists a triple $(0, 0, w^\alpha) \in \mathcal{R}(F_2)$ for some $w^\alpha \in \mathbb{R}_{++}^m$. Now for any given $w^\beta \in \mathbb{R}_{++}^m$, we are to show that $(0, 0, w^\beta) \in \mathcal{R}(F_2)$.

Let us define the line segment between w^α and w^β

$$w(t) = (1 - t)w^\alpha + t w^\beta,$$

and the number

$$\hat{t} = \sup\{t \in [0, 1] : \{(0, 0, w(t')) : t' \in [0, t]\} \subset \mathcal{R}(F_2)\}.$$

Since $(0, 0, w(0)) \in \mathcal{R}(F_2)$ and F_2 is homeomorphic between $\mathbb{R}^n \times \mathbb{R}_{++}^m \times \mathbb{R}_{++}^m$ and $\mathcal{R}(F_2)$, we must have $\hat{t} > 0$. If $\hat{t} = 1$, we already have $w^\beta \in \mathcal{R}(F_2)$ and we are done.

Now suppose $\hat{t} < 1$. This implies that $(0, 0, w(\hat{t})) \notin \mathcal{R}(F_2)$; otherwise by the local homeomorphism of F_2 the number \hat{t} would not have been a supremum. Consider the set

$$\mathcal{W} := \{w(t) : t \in [0, \hat{t})\} \subset \mathcal{R}(F_2),$$

which is clearly bounded. It follows from Lemmas 3.6 and 3.8, the set

$$\{(x(w), y(w), z(w)) : w \in \mathcal{W}\}$$

is also bounded. Let us denote $x(w(t))$ by $x(t)$, and so on. Then there must exist a sequence $\{t_k\}_{k=1}^\infty$ such that $t_k \rightarrow \hat{t}$ and $(x(t_k), y(t_k), z(t_k)) \rightarrow (\hat{x}, \hat{y}, \hat{z})$ for some $(\hat{x}, \hat{y}, \hat{z}) \in \mathbb{R}^n \times \mathbb{R}_+^m \times \mathbb{R}_+^m$ (otherwise, a convergent subsequence can be selected).

Since the function F_2 is continuous, we have $F_2(\hat{x}, \hat{y}, \hat{z}) = (0, 0, w(\hat{t}))^T$, which means that $(0, 0, w(\hat{t})) \in \mathcal{R}(F_2)$. This is a contradiction. Thus the assumption $\hat{t} < 1$ is false, and we have proved the lemma. \square

Finally we prove the existence and convergence of solution paths, including the central path, leading to the solution of the original MaxVE problem in the sense specified in the following theorem.

Theorem 3.3 (*Existence and Convergence of Path for $F = F_2$*)

For any $w^0 \in \mathbb{R}_{++}^m$ and $\mu > 0$, the system (3.17) with $F = F_2$ and $w = \mu w^0$ has a unique solution $(x(\mu), y(\mu), z(\mu))$. Moreover,

$$\lim_{\mu \rightarrow 0} (x(\mu), y(\mu), z(\mu), u(\mu), E(\mu)) = (x^*, y^*, z^*, u^*, E^*).$$

where (x^*, y^*, z^*) satisfies the system (3.15), and (x^*, E^*, u^*, z^*) satisfies the KKT system (3.7a)-(3.7e). Consequently, (x^*, E^*) solves the MaxVE problem (3.5).

Proof The first statement follows directly from Lemma 3.9 and the fact that F_2 is homeomorphic in $\mathbb{R}^n \times \mathbb{R}_{++}^m \times \mathbb{R}_{++}^m$.

By Lemmas 3.6 and 3.8, the quantities $x(\mu), y(\mu), z(\mu), u(\mu)$ and $E(\mu)$ are all bounded as $\mu \rightarrow 0$. Hence, they must have accumulation points as $\mu \rightarrow 0$, say, x^*, y^*, z^*, u^* and E^* . Clearly, these accumulation points satisfy the two systems in the theorem. Since these systems only permit unique solutions, we conclude that all accumulation points of $x(\mu)$ as $\mu \rightarrow 0$ must coincide, and the same is true for other quantities; namely, accumulation points are actually the limit point. Obviously, (x^*, E^*) solves the optimization problem (3.5) because they, together with u^* and z^* , satisfy the optimality conditions (3.7a)-(3.7e). This proves the theorem. \square

3.4.4 Issues of Algorithmic Convergence

So far polynomial convergence theory for primal-dual interior point algorithms has been established only for convex conic programming in symmetric cones (see [35], for example). Given the highly nonlinear formulations upon which we build our primal-dual interior-point algorithms, it seems unlikely that polynomial convergence could be proven for our primal-dual interior-point algorithms unless some new paradigm is discovered.

On the other hand, performing some non-polynomial, global convergence analysis for the proposed algorithmic framework appears to be a worthy task. Given the good properties we have already established for our formulations, we do not see fundamental difficulties in proving global and fast local convergence for some parameter choices in the proposed algorithmic framework. Such an analysis, however, would be rather lengthy and technical. This issue of algorithmic convergence can be a good topic of further research.

3.5 Khachiyan-Todd Algorithm and Modification

We will introduce two other algorithms, the Khachiyan and Todd algorithm [25] and a modification of it, and will later compare them with algorithms proposed in Section 3.

Given a set of inequalities $Ax \leq b$ and a strictly interior point x^0 , using the change of variable $x = v + x^0$, we can rewrite the inequalities as $Av \leq b - Ax^0$. By multiplying both sides by the positive diagonal matrix $\text{Diag}(b - Ax^0)^{-1}$, we obtain the following polytope

$$\mathcal{P} = \{v \in \mathbb{R}^n : Cv \leq e\}, \quad (3.40)$$

where $C \equiv \text{Diag}(b - Ax^0)^{-1}A \in \mathbb{R}^{m \times n}$ and e is the vector of all ones in \mathbb{R}^m . We will use this form of polytopes in this section as it was used by Khachiyan and Todd in [25].

In the formulation (3.5), the matrix-valued variable E appears in the constraints in a nonlinear manner. In an alternative formulation given below, through the change of variables $B = E^2$ one can have the unknown matrix B to appear linearly in the constraints. Indeed, after substituting E^2 by B and using the form (3.40), we can rewrite the problem (3.5) into

$$\begin{aligned} \min \quad & -\log \det B \\ \text{s.t.} \quad & c_i^T B c_i \leq (1 - c_i^T x)^2, \quad i = 1, \dots, m \\ & (Cx < e, \quad B \succ 0) \end{aligned} \tag{3.41}$$

While the constraints of (3.41) are linear with respect to the matrix variable B , they are no longer linear or convex with respect to the vector variable x .

3.5.1 Khachiyan and Todd's Algorithm

Khachiyan and Todd's algorithm [25] for the MaxVE problem has a good complexity bound and also takes the advantage of the special structure of the MaxVE problem. It is a suitable candidate for the purpose of performance comparison.

To make use of the simplicity of linear constraints, Khachiyan and Todd introduced the following subproblem, or auxiliary problem $AP(a)$, from (3.41):

$$\begin{aligned} \min \quad & -\log \det B \\ \text{s.t.} \quad & c_i^T B c_i \leq (1 - c_i^T x)(1 - c_i^T a), \quad i = 1, \dots, m \\ & (B \succ 0) \end{aligned} \tag{3.42}$$

for a fixed $a \in \mathbb{R}^n$ where $Ca < e$. Note that the constraints are linear in both B and x . The key idea here is to solve subproblems $AP(a)$ iteratively until x and a

become sufficiently close to each other so the subproblem (3.42) becomes a good approximation of (3.41). Khachiyan and Todd use a primal barrier method to solve the subproblem $AP(a)$ where the barrier function has the form

$$F_t(x, B | a) = -\log \det B - t \sum_{i=1}^m \log ((1 - c_i^T x)(1 - c_i^T a^k) - c_i^T B c_i).$$

where a is fixed and t is the barrier parameter. The Khachiyan and Todd (KT) Algorithm can be summarized as follows.

Algorithm 2 (*Khachiyan and Todd's Algorithm*)

Step 1. Let a^0 be a strictly interior point of \mathcal{P} . $B^0 \succ 0$, $\epsilon > 0$, and $k = 0$.

Step 2. Solve the subproblem $AP(a^k)$ by using Newton method to minimize the barrier function $F_t(x, B | a^k)$ for a sequence of $t \downarrow 0$. The solution of $AP(a^k)$ is (x^k, B^k) .

Step 3. If $\|x^k - a^k\| \leq \epsilon$, then stop; else let $a^{k+1} = (a^k + x^k)/2$, increment k and go to Step 2.

Khachiyan and Todd prove that to attain a sufficient accuracy only a small number of subproblems need to be solved, and they derive a linear system of size $n + m$ for calculating the Newton direction. The updates to the matrix-valued variable B are not directly calculated as an independent variable, thus reducing the complexity of the algorithm. However, the drawback of their algorithm is that the barrier method used to solve the subproblem is not particular efficient in practice. Moreover, as we can see from the algorithmic framework, three layers of loops are involved in the KT algorithm: the loop for the subproblem parameter a , the loop for barrier parameter t , and the iterations for a fixed a and a fixed t .

3.5.2 A Modification of the KT Algorithm

Since primal barrier methods are generally less efficient than primal-dual, interior-point methods, in order to speed up the KT algorithm we modify it by applying a primal-dual interior-point method to the subproblems in the Step 2 of the KT algorithm, while keeping the outer iterations intact.

Following Khachiyan and Todd's approach, we transform the subproblem $AP(a)$ into the standard form $AP(0)$:

$$\begin{aligned} \min \quad & -\log \det B \\ \text{s.t.} \quad & c_i^T B c_i + c_i^T x \leq 1, \quad i = 1, \dots, m \\ & (B \succ 0) \end{aligned} \tag{3.43}$$

by the change of variable $x \Leftarrow x - a$ and the change of data $c_i \Leftarrow c_i / (1 - c_i^T a)$ for $i = 1, \dots, m$.

The optimality conditions, or the Karush-Kuhn-Tucker (KKT) conditions, of problem $AP(0)$ are as follows:

$$C^T y = 0, \tag{3.44a}$$

$$B^{-1} - C^T Y C = 0, \tag{3.44b}$$

$$Cx + \text{diag}(CBC^T) + z - e = 0, \tag{3.44c}$$

$$Yz = 0, \tag{3.44d}$$

$$y, z \geq 0. \tag{3.44e}$$

where $y \in \Re^m$ is the vector of Lagrangian multipliers, $z \in \Re^m$ consists of slack variables, and $C \in \Re^{m \times n}$ with c_i^T as its i -th row.

Following the same strategy used earlier, we eliminate the matrix variable B from the system using the substitution $B(y) = (C^T Y C)^{-1}$ that is the solution to (3.44b). We also replace the zero right-hand side of (3.44d) by μe . The resulting system that defines the central path is

$$F_3(x, y, z) := \begin{pmatrix} C^T y \\ Cx + \text{diag}(Q(y)) + z - e \\ Yz \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \mu e \end{pmatrix} \quad (3.45)$$

and $y, z > 0$ where $Q(y) = C(C^T Y C)^{-1} C^T$. Clearly, the equation (3.45) is a square, nonlinear system of $n + 2m$ variables. The Jacobian matrix of $F_3(x, y, z)$ is

$$F'_3(x, y, z) = \begin{bmatrix} 0 & C^T & 0 \\ C & -Q \circ Q & I \\ 0 & Z & Y \end{bmatrix}.$$

To solve the Newton linear system

$$F'_3(x, y, z) \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} := \begin{pmatrix} 0 \\ 0 \\ \mu e \end{pmatrix} - F_3(x, y, z).$$

we use the following block Gaussian elimination procedure:

$$\begin{aligned} dx &= (C^T M^{-1} C)^{-1} (r_1 + C^T M^{-1} (r_2 - Y^{-1} r_3)), \\ dy &= M^{-1} (C dx - r_2 + Y^{-1} r_3), \\ dz &= Y^{-1} (r_3 - Z dy), \end{aligned}$$

where the matrix $M := Q \circ Q + Y^{-1} Z$ is symmetric positive definite.

The primal-dual algorithm for solving the subproblem $AP(0)$ falls into the same framework of Algorithm 1.

3.6 Numerical Results

In this section, we report our numerical results on the four algorithms: the KT algorithm, the modified KT, or MKT, algorithm, and the two direct primal-dual interior-point algorithms based on the systems (3.13) and (3.15) which we name F1PD and

F2PD, respectively. The numerical tests were performed on three sets of test problems with a total of 200 problems. Our implementation of the four algorithms is in MATLAB. All the experiments were run on an SGI Origin2000 computer with multiple 300-MHZ R12000 processors. However, our programs use only a single processor at a time.

3.6.1 Implementation Details

In describing the implementation details, we first give some features common to all the algorithms and then other features specific to individual algorithms.

For all the algorithms, the input data for a polytope include the matrix A , the vector b and a strictly interior point x^0 such that $Ax^0 < b$ which will serve as the starting point for the center of the initial ellipsoid. In our implementation, the point x_0 is selected to be the solution to an auxiliary linear program $\max\{\tau : Ax + \tau e \leq b\}$. Other choices are certainly possible such as the analytic center of the polytope. However, it was our intention not to use the best possible starting point.

Scaling is an important issue in numerical computation. In our implementations, we always first transform the inequality $Ax \leq b$ into the form $Cv \leq e$ using the change of variables and the row scaling as is described at the beginning of Section 3.5. After the transformation, the starting point x^0 is transformed into the origin, and the transformed polytope turns out to be better scaled.

In all the algorithms, the stopping tolerance is set to $\epsilon = 10^{-4}$. In the case of the KT and MKT algorithms, we stop the outer iterations whenever the relative change between the current and previous centers is less than or equal to ϵ . In the case of the F1PD and F2PD algorithms, we stop whenever the residual norm of F_i , $i = 1$ or 2 , becomes less than or equal to ϵ .

We now describe some algorithm-specific features.

- The KT and MKT algorithms: Both algorithms have the same outer loop with the center varying. The initial center is the origin and the initial value for the matrix variable B is $B^0 = \rho I$ where I is the identity matrix and ρ is chosen such that the corresponding ball, centered at the origin with a radius ρ , lies entirely inside the polytope. During the outer iterations, we use a warm-start strategy in which a later iteration always starts from the solution of the previous iteration.
- The KT algorithm: In the subproblems, the barrier parameter t is set to 0.5 initially and then decreased by a factor of 10 whenever the subproblem stopping criterion is met. For a fixed t value, the subproblem stopping criterion is that the gradient norm of the corresponding barrier function becomes less than or equal to t . This way, the stopping criterion becomes progressively more stringent as t approaches zero. We found that this adaptive strategy made the algorithm run significantly faster. To prevent the loss of symmetry during the computation, we set $B = (B + B^T)/2$ after B is updated at every iteration. We update an iterate for (x, B) by a damped Newton step to ensure that the updated ellipsoid remains inside the polytope. Specifically, the step length is 0.75 times the largest allowable step that keeps the updated ellipsoid inside the polytope.
- The Primal-Dual algorithms: The primal-dual algorithmic framework (i.e., Algorithm 1) encompasses the F1PD and F2PD algorithms, and the subproblem solver of the MKT algorithm. The initial values for the primal-dual algorithms are set as follows: the initial center is $x = 0$; the initial multiplier value is $y = e$; and the initial slack variable z , say in the equation $z - g = 0$, is set as $z_i = \max(0.1, g_i)$. In addition to the initial values, there are two critical parameters in these algorithms: the so-called centering parameter σ^k and the step length α^k . In our

implementations, we choose $\sigma^k = \min\{0.5, (y^k)^T z^k / m\}$ and $\alpha^k = \min(1, \tau \hat{\alpha})$ where $\tau \in (0, 1)$ and $\hat{\alpha}$ is the maximum length such that updated iterate for (x, y, z) reaches the boundary of the set $\mathcal{P} \times \mathbb{R}_{++}^m \times \mathbb{R}_{++}^m$. We use $\tau = 0.75$ for the F1PD and F2PD algorithms, and a more aggressive $\tau = 0.9$ for the subproblem solver of the MKT algorithm because the subproblem (3.42) is not as nonlinear as its counterparts are in the F1PD and F2PD algorithms.

The parameter settings given above are rather generic and unsophisticated. For example, a line search scheme for determining step length could be a more effective and theoretically sound strategy. However, given our purpose of identifying the most robust and efficient algorithm, we consider our current settings to be appropriate and sufficient.

3.6.2 Test Problems

Three sets of test problems were used in our numerical experiments, consisting of 47, 143 and 10 problems, respectively. The total number of test problems is 200 and all the 200 test problems are made available at the web site:

<http://www.caam.rice.edu/~zhang/maxvep/>.

Test sets 1 and 2 are obtained from two integer programming feasibility problems through the search trees of an integer programming algorithm – the Lenstra algorithm for integer programming feasibility problem [28, 30]. This algorithm searches on a tree of subproblems and applies ellipsoidal approximation on each one of them. The polytopes in Sets 1 and 2 are taken from some branches of the search trees for two integer programming feasibility problems, respectively. The problem sizes in Sets 1 and 2 are relatively small with $m \leq 288$ and $n \leq 80$. Nevertheless, our numerical experience has indicated that some of the problems are non-trivial to solve.

In order to test the ability of our algorithms for solving larger problems, we generated a set of ten random problems that is called Set 3. The largest problem in this set has $m = 1200$ and $n = 500$. For each problem, we first use the MATLAB function `sprandn` to generate a sparse random matrix B , and also use the `rand` function to generate a right-hand side vector $c > 0$, a upper-bound vector $ub > 0$ and a lower-bound vector $lb < 0$. Together, they form a polytope

$$\{x \in \mathbb{R}^n : Bx \leq c, \quad lb \leq x \leq ub\}$$

where $B \in \mathbb{R}^{k \times n}$ and $c \in \mathbb{R}^k$ and $lb, ub \in \mathbb{R}^n$. By construction, the origin $x = 0$ is strictly interior to the polytope. Then we rewrite the polytope into the standard form

$$\{x \in \mathbb{R}^n : Ax \leq b\},$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ with $m = k + 2n$. The matrix A is constructed, in an obvious manner, from the matrix B and the identity matrix in \mathbb{R}^n , and the vector b from the vectors $c \in \mathbb{R}^k$ and $lb, ub \in \mathbb{R}^n$. The problems in Set 3 are sparse.

3.6.3 Test Results

Test results on Sets 1 and 2, totaling 190 problems are summarized in Table 3.1, while detailed results are given in Tables 3.3-3.6. Four rows of numbers are presented in Table 3.1. In the first two rows, we list the test set number, the number of test problems in each set, the total number of iterations and the total amount of CPU time taken by each algorithm for solving the entire set of test problems. In the last two rows, we gives the algebraic and geometric means for each category over the 190 test problems.

We note that the iteration numbers for the KT and the MKT algorithms are the numbers of inner-most, Newton iterations that involve solving linear systems of

equations. These inner-most iterations are comparable with the iterations of the direct primal-dual algorithms because for a given problem they all require to solve linear systems of essentially the same size. The CPU time is given in seconds. In addition, detailed results on the test sets 1 and 2 are attached as Tables 3.3-3.6.

Table 3.1 Summary of Results for Tests 1 and 2

Prob Set	# of Prob	KT		MKT		F1PD		F2PD	
		Iter	Time	Iter	Time	Iter	Time	Iter	Time
1	47	19416	3340	2655	240	692	124	694	77
2	143	56783	3567	9720	429	2448	168	2058	104
Total	190	76199	6907	12375	669	3140	292	2752	181
alge.	mean	401.0	36.4	65.1	3.5	16.5	1.5	14.5	1.0
geom.	mean	397.7	30.5	64.3	3.2	16.2	1.3	14.3	0.8

From Table 3.1, we observe that on average, the MKT algorithm is about 10 times faster than the original KT algorithm; F1PD algorithm is about 2 times faster than the MKT algorithm, and the F2PD algorithm is about 1.5 times faster than the F1PD algorithm.

We mention that out of the 190 test problems in test sets 1 and 2 the KT algorithm failed to converge on two: problems 22 and 120 in the set 2. More conservative choices of parameters would make the KT algorithm converge on these two problems, but would also adversely affect the overall performance of the algorithm. We kept the current choices of parameters for the benefit of the KT algorithm.

The test results on the randomly generated test set 3 are presented in Table 3.2. Only the F1PD and F2PD algorithms were tested on this set of larger problems because the other two algorithms, non-competitive in time, would require an excessively long time to run. Since these test problems are sparse, in addition to the matrix sizes

m and n we also include the number of nonzero entries, denoted as nnz , in the matrix A .

Table 3.2 Results of Test Set 3: Problems 1-10

Prob No.	Size			F1PD		F2PD	
	m	n	nnz	Iter	Time (sec)	Iter	Time (sec)
1	600	100	7426	31	97	22	30
2	600	150	8408	30	107	23	39
3	600	200	7669	53	203	29	58
4	600	250	5022	60	249	31	73
5	800	100	5914	34	235	22	63
6	800	200	8029	34	271	24	91
7	800	300	8933	58	549	32	165
8	1000	300	11993	40	675	28	245
9	1000	400	8433	60	1134	31	330
10	1200	500	10518	73	2917	37	703
Total	—	—	—	473	6433	279	1796
alge. mean	—	—	—	47.3	643.3	27.9	179.6
geom. mean	—	—	—	45.1	364.1	27.5	110.3

The results in Table 3.2 indicate that given the current choices of parameters, the F2PD algorithm clearly outperforms the F1PD algorithm by a considerable margin on the test set 3. Although the performance of the F1PD algorithm may be somewhat improved by selecting different parameters, we do not believe that it can in general outperform the F2PD algorithm because it requires more linear algebra calculation in each iteration for solving its version of the Newton linear system.

Detailed Data for Sets 1 and 2

The first three columns of the tables 3.3-3.6 give problem number and size where m is the number of polytope-defining inequalities and n the number of variables. For the KT and MKT algorithms, the tables give both the number of outer-iterations (i.e., the number of subproblems solved) and the number of inner-iterations (i.e., the

number of Newton iterations), separated by a slash. The time is measured by CPU seconds. On Problems 22 and 120 of the test set 2, the KT algorithm failed to solve the first subproblem (i.e., at the zero-th outer iteration). For these two problems, we set the outer iteration number to 0 and give the number of inner iterations the algorithm took before it stopped.

3.7 Concluding Remarks

The goal of this study is to find a practically efficient algorithmic framework for solving general MaxVE problems. Our extensive numerical results show that among the four tested algorithms, the method of choice is clearly the F2PD algorithm built on the formulation (3.15), which has been shown to have a sound theoretical foundation. We have established, among other things, the existence of a central path for this formulation even though, unlike in the conventional cases, this central path is not known to be directly connected to the optimality conditions of a barrier function.

The main advantage of the F2PD algorithm over the KT and the MKT algorithms is that, without the need for solving a number of subproblems either for fixed centers or fixed barrier parameter values, it requires less iterations (or linear system solutions) than the other two algorithms. We expect that the same advantage would still hold against some other untested algorithms like the one given in [3]. On the other hand, compared to the F1PD algorithm, the F2PD algorithm requires less linear algebra computation per iteration and seems to be more robust. These features make the F2PD algorithm particularly attractive.

The algorithms considered in this chapter are all of the general-purpose type. For really large-scale problems with special structures, one will likely need special-purpose algorithms that can take full advantage of the problem-specific structures, in

particular sparsity, in order to solve the problems efficiently. This should be a topic of further research.

Table 3.3 Results of Test Set 1: Problems 1-47

Prob No.	Size		KT		MKT		FIPD		F2PD	
	m	n	Iter	Time (sec)	Iter	Time (sec)	Iter	Time (sec)	Iter	Time (sec)
1	288	80	8/369	155.83	8/43	10.14	21	8.82	18	4.40
2	286	79	9/416	194.64	9/79	19.18	20	9.00	21	5.80
3	272	72	9/375	137.47	9/48	9.30	15	5.33	15	3.22
4	260	66	10/431	145.55	10/58	9.81	17	5.54	17	3.38
5	256	64	10/410	132.96	10/49	8.04	14	4.39	14	2.69
6	246	59	10/414	117.01	10/53	7.44	15	4.09	15	2.50
7	244	58	10/411	116.80	10/52	7.11	14	3.79	15	2.52
8	236	54	10/405	98.98	10/57	7.05	15	3.55	16	2.33
9	234	53	10/414	101.27	10/54	6.51	15	3.54	15	2.21
10	232	52	10/425	101.98	10/53	6.17	15	3.54	16	2.37
11	230	51	9/386	87.61	9/52	5.98	15	3.31	15	2.11
12	228	50	9/384	85.17	9/54	6.08	15	3.26	15	2.01
13	226	49	10/441	96.21	10/56	6.06	15	3.20	16	2.19
14	224	48	10/433	88.96	10/58	6.19	16	3.33	16	2.07
15	222	47	10/439	88.64	10/55	5.57	16	3.18	16	1.95
16	220	46	10/432	83.21	10/55	5.41	16	3.08	16	1.96
17	218	45	10/433	81.07	10/54	5.16	16	2.94	16	1.81
18	216	44	10/439	80.37	10/55	5.10	16	2.95	16	1.76
19	212	42	9/405	69.66	9/52	4.58	15	2.54	15	1.55
20	210	41	9/391	65.11	9/53	4.57	13	2.14	13	1.36
21	208	40	10/423	67.76	10/63	5.26	16	2.57	15	1.47
22	206	39	10/434	66.50	10/61	4.91	16	2.53	15	1.49
23	206	39	10/434	66.50	10/63	5.10	15	2.30	15	1.46
24	204	38	10/431	64.24	10/60	4.67	16	2.43	15	1.35
25	202	37	9/393	55.93	9/63	4.84	15	2.11	15	1.31
26	200	36	9/393	53.98	10/59	4.29	15	2.04	15	1.26
27	198	35	10/429	57.06	10/60	4.24	15	1.99	15	1.22
28	196	34	10/428	55.18	10/67	4.64	15	1.92	15	1.20
29	194	33	10/430	53.54	10/66	4.45	16	1.99	15	1.16
30	192	32	10/424	50.88	10/58	3.83	15	1.83	15	1.15
31	192	32	10/434	52.16	10/61	4.05	15	1.85	14	1.05
32	190	31	10/411	46.93	10/56	3.46	15	1.75	15	1.10
33	188	30	10/410	44.56	10/57	3.39	15	1.69	15	1.04
34	186	29	10/406	42.55	10/56	3.21	14	1.51	14	0.97
35	184	28	10/410	41.42	10/60	3.34	13	1.34	13	0.83
36	184	28	10/406	40.98	10/60	3.37	14	1.45	14	0.91
37	182	27	10/417	40.33	10/56	3.03	13	1.29	14	0.86
38	180	26	10/413	37.95	10/58	3.03	13	1.25	14	0.81
39	180	26	10/413	37.95	10/57	2.93	14	1.35	14	0.84
40	178	25	10/403	35.61	10/56	2.76	13	1.19	13	0.72
41	176	24	10/406	34.18	10/51	2.40	13	1.21	13	0.69
42	174	23	10/418	33.77	10/54	2.46	12	0.99	13	0.68
43	172	22	10/404	31.10	10/50	2.17	12	0.95	13	0.66
44	172	22	10/405	31.17	10/52	2.28	12	0.97	13	0.66
45	168	20	10/404	28.50	10/57	2.31	13	0.96	14	0.63
46	164	18	10/384	24.35	10/56	2.10	13	0.89	12	0.51
47	162	17	10/400	15.91	10/48	1.61	10	0.42	10	0.39

Table 3.4 Results of Test Set 2: Problems 1-48

Prob No.	Size		KT		MKT		F1PD		F2PD	
	m	n	Iter	Time (sec)	Iter	Time (sec)	Iter	Time (sec)	Iter	Time (sec)
1	147	50	11/463	34.60	11/73	3.78	21	1.71	16	0.98
2	147	49	9/426	32.07	12/88	4.44	20	1.59	16	0.96
3	147	48	7/376	27.92	9/67	3.43	18	1.41	14	0.81
4	147	49	9/390	29.09	12/83	4.20	25	2.09	18	1.11
5	147	48	7/370	27.43	9/65	3.31	18	1.47	14	0.80
6	147	47	7/407	29.87	9/64	3.22	17	1.39	14	0.85
7	147	48	9/403	29.54	10/77	3.92	26	2.05	19	1.19
8	147	47	8/391	28.61	10/74	3.64	19	1.51	16	0.95
9	147	49	9/395	29.52	11/72	3.60	20	1.70	16	0.99
10	147	48	7/385	29.59	11/73	3.61	16	1.32	14	0.86
11	147	48	9/414	31.41	12/83	4.13	24	2.00	19	1.19
12	147	47	8/380	27.81	10/74	3.75	18	1.44	16	0.92
13	147	48	9/398	30.07	12/79	3.91	21	1.79	17	1.05
14	147	47	8/383	28.59	10/73	3.66	19	1.58	16	0.94
15	147	47	9/389	28.83	10/74	3.66	23	1.84	18	1.06
16	137	35	7/513	26.29	7/55	2.11	20	1.11	15	0.61
17	137	35	7/491	25.08	7/49	1.84	14	0.79	13	0.54
18	137	35	7/383	19.23	7/50	1.89	16	0.91	13	0.57
19	137	34	6/383	19.22	7/47	1.74	13	0.70	13	0.51
20	137	34	7/415	20.74	7/49	1.79	15	0.85	13	0.53
21	132	30	6/336	14.67	6/48	1.62	15	0.74	13	0.46
22	134	33	0/166	8.36	7/55	1.94	21	1.13	16	0.61
23	137	36	7/354	17.94	7/64	2.53	21	1.20	17	0.69
24	137	35	7/358	18.02	7/63	2.45	18	1.00	16	0.72
25	137	34	7/396	19.95	7/57	2.22	20	1.09	18	0.73
26	137	34	8/409	20.60	9/73	2.70	20	1.15	17	0.70
27	136	33	6/320	15.61	6/59	2.20	23	1.29	18	0.71
28	136	32	7/506	25.15	8/71	2.61	20	1.13	18	0.75
29	137	36	7/484	25.32	7/58	2.28	16	0.89	13	0.53
30	137	35	8/588	30.33	9/58	2.14	14	0.80	13	0.53
31	137	35	6/432	22.19	7/57	2.17	21	1.15	15	0.61
32	137	34	7/437	21.98	7/53	1.97	16	0.87	13	0.52
33	137	34	7/422	21.24	7/45	1.62	13	0.73	13	0.53
34	137	36	7/361	18.32	7/63	2.43	21	1.16	17	0.70
35	137	35	7/359	18.19	7/61	2.36	19	1.08	17	0.71
36	137	35	6/299	15.09	6/66	2.67	23	1.34	19	0.84
37	137	34	7/366	18.47	8/68	2.56	19	1.05	17	0.69
38	137	35	6/299	15.08	6/57	2.24	20	1.14	17	0.69
39	137	34	6/296	14.70	6/57	2.19	20	1.14	18	0.76
40	136	33	6/323	15.77	6/59	2.22	24	1.33	19	0.75
41	147	48	9/390	29.49	11/74	3.67	21	1.68	17	1.02
42	147	47	7/357	26.46	10/85	4.29	15	1.18	13	0.76
43	147	46	7/462	34.29	7/59	3.04	19	1.47	15	0.85
44	147	47	8/363	26.70	13/75	3.55	17	1.34	14	0.82
45	147	46	7/419	30.84	14/88	4.19	15	1.16	13	0.74
46	147	46	8/372	26.95	12/80	3.84	15	1.15	13	0.75
47	147	45	7/354	24.79	12/79	3.74	14	1.06	13	0.71
48	147	44	7/442	30.97	12/82	3.88	17	1.26	14	0.76

Table 3.5 Results of Test Set 2: Problems 49-96

Prob No.	Size		KT		MKT		F1PD		F2PD	
	m	n	Iter	Time (sec)	Iter	Time (sec)	Iter	Time (sec)	Iter	Time (sec)
49	147	46	8/368	26.21	10/65	3.14	17	1.30	14	0.79
50	147	45	7/371	26.21	13/82	3.84	15	1.13	13	0.72
51	147	45	8/370	25.95	11/71	3.40	16	1.21	14	0.78
52	147	43	7/370	24.92	12/75	3.44	15	1.08	13	0.68
53	147	46	7/359	25.71	11/72	3.46	15	1.14	12	0.68
54	147	47	11/479	35.21	11/84	4.22	19	1.49	17	0.99
55	147	46	8/401	27.98	12/80	3.83	15	1.12	14	0.76
56	147	46	10/459	31.92	12/80	3.81	17	1.28	16	0.87
57	147	45	9/430	29.70	13/74	3.40	15	1.12	14	0.75
58	147	45	8/396	26.77	9/67	3.23	13	0.94	12	0.65
59	135	39	7/389	20.67	7/55	2.16	14	0.81	13	0.56
60	135	38	7/403	21.29	9/61	2.31	14	0.80	12	0.52
61	135	38	7/461	24.80	8/59	2.30	15	0.87	12	0.51
62	135	38	7/390	20.77	9/61	2.30	14	0.81	12	0.51
63	135	37	8/383	19.70	11/67	2.42	15	0.84	14	0.58
64	135	39	8/381	20.03	11/64	2.32	12	0.70	11	0.47
65	135	38	7/340	17.95	10/58	2.12	11	0.63	11	0.49
66	135	38	7/350	18.56	10/60	2.20	11	0.63	11	0.48
67	135	38	6/345	18.53	10/70	2.60	14	0.81	12	0.52
68	135	37	7/455	23.12	8/53	1.98	13	0.71	12	0.49
69	135	38	7/381	19.89	9/62	2.34	14	0.80	12	0.50
70	135	36	7/389	19.86	10/65	2.31	14	0.79	13	0.53
71	135	35	7/472	23.91	7/55	2.07	15	0.82	13	0.52
72	135	35	7/357	17.73	9/55	1.93	11	0.61	11	0.44
73	135	34	7/393	19.29	9/62	2.20	15	0.81	13	0.50
74	135	35	7/399	20.11	11/61	2.12	13	0.71	12	0.49
75	135	35	7/472	23.92	7/55	2.07	15	0.82	13	0.52
76	135	35	7/356	17.67	10/59	2.06	13	0.71	12	0.48
77	135	36	7/459	23.39	9/63	2.30	14	0.76	12	0.49
78	135	37	6/379	19.85	7/55	2.10	13	0.73	12	0.51
79	137	39	7/425	23.02	13/83	3.19	15	0.87	13	0.56
80	137	40	7/415	22.99	10/65	2.63	14	0.84	12	0.53
81	147	45	8/398	26.92	9/66	3.17	13	0.95	12	0.63
82	135	38	7/381	19.86	10/66	2.43	14	0.80	12	0.50
83	134	37	7/365	18.55	10/58	2.07	13	0.72	12	0.50
84	134	37	7/376	19.15	7/55	2.07	14	0.77	12	0.51
85	134	37	7/418	21.52	8/58	2.15	14	0.77	12	0.50
86	135	39	7/410	21.69	8/59	2.30	14	0.80	12	0.52
87	137	37	7/373	19.62	9/65	2.45	16	0.92	13	0.54
88	128	36	7/385	17.27	7/57	1.99	16	0.79	13	0.51
89	128	36	7/395	17.23	10/60	1.98	13	0.63	12	0.46
90	137	39	7/338	18.27	10/61	2.34	11	0.65	11	0.48
91	128	37	6/316	14.37	11/71	2.40	14	0.72	13	0.50
92	137	38	6/399	21.73	13/79	2.96	13	0.76	12	0.51
93	137	38	8/370	19.65	12/65	2.35	11	0.64	11	0.48
94	137	37	7/372	19.61	11/74	2.80	15	0.85	13	0.55
95	137	37	6/384	20.61	11/67	2.48	14	0.81	12	0.51
96	137	39	7/389	21.29	10/68	2.62	13	0.76	12	0.53

Table 3.6 Results on Test Set 2: Problems 97-143

Prob No.	Size		KT		MKT		F1PD		F2PD	
	m	n	Iter	Time (sec)	Iter	Time (sec)	Iter	Time (sec)	Iter	Time (sec)
97	137	38	5/346	18.75	8/59	2.35	15	0.86	12	0.51
98	137	39	6/364	20.08	10/63	2.47	13	0.78	12	0.52
99	137	39	7/425	23.03	13/83	3.15	15	0.87	13	0.56
100	137	38	7/394	21.05	11/67	2.52	15	0.86	13	0.56
101	128	37	7/350	15.76	11/68	2.30	15	0.76	13	0.50
102	128	36	6/306	13.53	11/69	2.29	16	0.79	14	0.54
103	137	40	7/416	23.02	10/65	2.58	14	0.84	12	0.53
104	147	47	7/353	26.40	10/70	3.44	15	1.18	13	0.76
105	147	48	9/401	31.14	12/80	3.96	24	1.94	18	1.07
106	147	47	9/428	32.29	12/83	4.07	20	1.58	16	0.93
107	147	46	7/419	30.25	10/73	3.59	14	1.06	12	0.68
108	147	46	9/446	31.81	12/78	3.73	19	1.43	15	0.83
109	147	45	7/359	25.30	9/70	3.43	14	1.05	13	0.75
110	147	44	7/552	39.44	7/59	2.98	16	1.18	14	0.76
111	147	47	7/396	29.70	8/68	3.46	15	1.17	13	0.77
112	147	48	9/438	33.47	12/82	4.06	16	1.30	14	0.84
113	147	48	7/432	33.53	10/70	3.54	14	1.13	12	0.73
114	147	49	9/396	29.62	11/71	3.56	20	1.60	16	0.94
115	147	48	8/425	31.56	12/80	3.94	17	1.33	14	0.80
116	147	47	7/410	30.40	9/66	3.32	18	1.39	14	0.82
117	147	48	9/420	30.96	12/83	4.10	24	1.87	19	1.10
118	147	47	7/407	30.10	7/54	2.77	17	1.32	13	0.76
119	147	47	9/431	31.60	11/81	3.99	21	1.64	17	0.97
120	147	46	0/164	12.65	8/69	3.55	25	1.91	18	1.01
121	147	46	8/411	29.85	8/70	3.53	20	1.53	16	0.91
122	147	48	9/391	28.62	12/79	3.89	21	1.66	17	0.98
123	147	47	9/449	32.80	9/74	3.73	21	1.63	17	0.97
124	147	46	8/423	30.63	8/68	3.40	18	1.37	15	0.84
125	147	47	9/403	29.17	10/75	3.71	22	1.69	17	0.98
126	147	46	8/395	28.53	13/84	4.01	19	1.45	16	0.90
127	147	43	7/521	36.15	7/67	3.29	19	1.37	15	0.79
128	147	46	9/419	30.14	11/77	3.73	24	1.84	18	1.01
129	147	45	9/402	28.43	11/74	3.52	23	1.73	17	0.95
130	147	45	7/416	30.03	7/59	3.00	16	1.21	13	0.73
131	147	49	9/389	29.05	12/82	4.11	23	1.87	17	1.02
132	147	48	7/373	27.86	12/80	3.96	18	1.41	15	0.87
133	147	47	7/410	30.43	9/66	3.32	17	1.32	14	0.80
134	147	48	9/419	30.81	12/83	4.11	24	1.88	19	1.10
135	147	47	7/400	29.58	7/54	2.81	17	1.32	14	0.81
136	147	47	9/432	31.51	11/81	3.98	20	1.56	17	0.97
137	147	46	8/419	30.38	8/67	3.36	18	1.38	15	0.85
138	147	49	9/410	30.74	12/86	4.34	21	1.67	17	1.01
139	147	48	8/402	29.83	10/74	3.71	18	1.41	15	0.87
140	147	47	8/420	30.66	10/74	3.66	18	1.42	15	0.85
141	147	48	6/381	28.65	8/62	3.24	17	1.34	14	0.81
142	147	50	12/514	38.50	12/86	4.40	27	2.17	20	1.18
143	147	49	8/415	31.70	8/77	4.11	21	1.67	18	1.06

Chapter 4

Lenstra's Algorithm for Integer Programming

4.1 Introduction

Many practical decision problems can be formulated as integer programming or mixed integer programming problems, such as transportation scheduling, production planning and telecommunications. Because integer programming problems have wide applicability, algorithms for IP are of great interest to researchers. The most common approach to solving integer programming problems is the branch-and-bound method. However, there is no guarantee that the branch-and-bound method will terminate in time polynomial in the size of input. While specially structured 0-1 integer programming problems have attracted a lot of attention in this field, very few results are reported on general integer programming. Many practical general integer programming solvers rely on heuristic algorithms or random sampling to get a "good" feasible solution. Still, some hard integer programming problems remain unsolved. A breakthrough came in 1983 when H. W. Lenstra, Jr. showed that integer programming problems in a fixed dimension can be solved in polynomial time. However, the computational properties of Lenstra's algorithm have not been fully explored [9]. Therefore, this dissertation studies issues involved in building a practical code based on Lenstra's algorithm using ellipsoidal approximation to solve general integer programming problems.

The integer programming problem (IP) can be stated as follows. Let m and n be positive integers.

Given a rational $m \times n$ matrix A and a rational m -vector b , does $Ax \leq b$ have an integral solution x ?

The integer linear programming problem is NP-complete, essentially due to S. A. Cook [8]. Cook's result implies that IP belongs to the most difficult problems in the class of NP. A question under this circumstance is: can the integer programming problem with a fixed number of variables be solved in polynomial time?

In special cases, the answer to the above question is yes. Consider the integer linear programming problem with a fixed number of variables n . In the case $n = 1$, the solution of the problem is trivial. For $n = 2$, Scarf [39, 40] gave a complete treatment after Hirschberg and Wong [15] and Kannan [19] had given polynomial algorithms in special cases.

In general cases, a breakthrough came when H. W. Lenstra, Jr. [28] proved that for a fixed number of variables n , there is a polynomial algorithm for integer linear programming problems. The proof consists of two main techniques: the rounding of polytopes and basis reduction on lattices. Lenstra's algorithm introduced the geometry of numbers concept into integer programming and proposed the strategy of branching on hyperplanes instead of on variables as in traditional branch and bound methods. Later, Grötschel, Lovász, and Schrijver [14] further proved that the rounding problem can be solved in polynomial time even for varying n .

Lovász and Scarf [31] applied the same idea, branching on hyperplanes with respect to a flat direction, to the body of the polytope and developed the generalized basis reduction algorithm to solve the problem without ellipsoidal approximations. W. Cook *et al* [9] implemented the generalized basis reduction algorithm and solved some difficult integer network design instances. Wang [49] solved a set of linear and nonlinear integer programming test problems using the generalized basis reduction algorithm as a subroutine. Recently Aardal, Hurkens, and Lenstra [1] developed an algorithm for solving a system of Diophantine equations with lower and upper bounds on the variables, based on the lattice basis reduction algorithm.

Since some small but hard problems still remain unsolved, it is worthwhile to build an efficient implementation of Lenstra's algorithm and compare its performance with the generalized basis reduction algorithm. The generalized basis reduction algorithm requires the solution of many linear programming problems, and there are tradeoffs between using an ellipsoidal approximation to the polytope, or working directly with the polytope itself. In this dissertation, we concentrate on building a version of Lenstra's algorithm using ellipsoidal approximations.

To this end, we have developed a practical primal-dual interior point method to solve this problem, which is discussed in Chapter 3. The other ingredient is the lattice basis reduction method, which has been implemented in LiDIA library [29]. We use CPLEX [36] to solve the linear programming problems involved. We give insights on the computational properties of Lenstra's algorithm based on our implementation and test experiments.

An efficient implementation of Lenstra's algorithm will supply a new tool to attack some small but hard general integer programming problems. To get a feasible integer solution is a crucial step in solving integer programming problems. Once a good feasible solution is found, the branch-and-bound method may terminate more quickly. Therefore, Lenstra's algorithm can be used when the traditional branch-and-bound method tends to go very deep into its search tree and is deemed inefficient on those cases.

The structure of this chapter is as follows. Section 2 describes the framework of Lenstra's algorithm and its improvement. In Section 3, we summarize the main results in lattices and basis reduction applied in integer programming. We briefly compare the properties of Lenstra's algorithm and the generalized basis reduction algorithm in Section 4.

4.2 Lenstra's Algorithm and Improvement

Assume that A is a real $m \times n$ matrix and $b \in \mathcal{R}^m$, then $Ax \leq b$ is called a *system of linear inequalities*. The solution set $\{x \in \mathcal{R}^n | Ax \leq b\}$ of a system of linear inequalities is called a *polyhedron*. A bounded polyhedron \mathcal{P} is called a *polytope*. An inequality $a_i^T x \leq b_i$ from $Ax \leq b$ is called an *implicit equality* if $a_i^T x = b_i$ for all x satisfying $Ax \leq b$. Let $A^=x \leq b^=$ denote the system of implicit equalities in $Ax \leq b$. The dimension of a polyhedron \mathcal{P} is equal to n minus the rank of the matrix $A^=$. \mathcal{P} is full-dimensional if its dimension is n . Therefore, \mathcal{P} is full-dimensional if and only if there are no implicit equalities [42]. From a geometric point of view, the integer feasibility problem is to find an integer point inside the polytope given by the system of linear inequalities.

We give two useful definitions for Lenstra's algorithm first. The *width* of a polytope $\mathcal{P} \in \mathcal{R}^n$ in a nonzero direction c is defined as

$$F(c) = \max\{c^T x : x \in \mathcal{P}\} - \min\{c^T x : x \in \mathcal{P}\}.$$

If $F(c)$ is bounded, we call the direction c a *thin direction* of the polytope. Note that the width of a polytope $\mathcal{P} = \{Ax \leq b\}$ in a given direction c can be computed by solving two associated LP problems.

The intuition of Lenstra's algorithm is the observation that when a polytope is "thin" and extends arbitrarily far in some directions as in Figure 4.1, the traditional branch-and-bound search tree will grow exceedingly deep in order to find out whether there is a feasible solution or not. The problem can be fixed if a thin direction can be identified.

Assume that $\mathcal{P} = \{x \in \mathcal{R}^n : Ax \leq b\}$, with $A \in \mathcal{Q}^{m \times n}$ and $b \in \mathcal{Q}^m$, is bounded and full-dimensional. The key idea of Lenstra's algorithm is to find either an integral

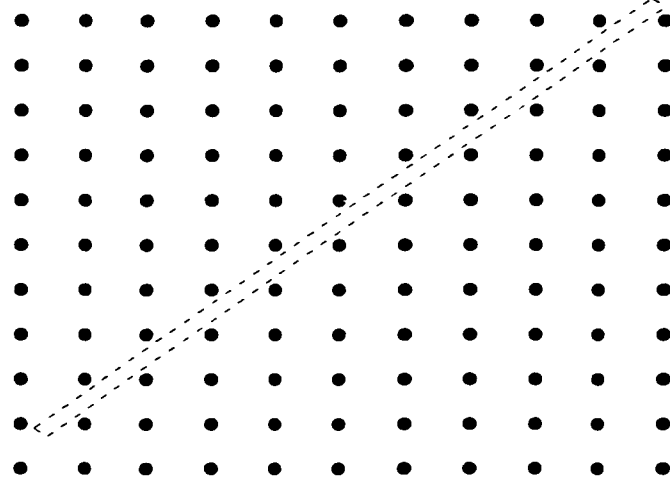


Figure 4.1 A Thin Polytope in 2-D

vector $a \in \mathcal{P} \cap \mathbb{Z}^n$ or a nonzero integral vector (or a flat direction) c such that

$$\max\{c^T x : x \in \mathcal{P}\} - \min\{c^T x : x \in \mathcal{P}\} \leq c(n)$$

in polynomial time where $c(n)$ depends only on n . Babai [4] and Lovász [30] improved Lenstra's algorithm and obtained an improved bound $c(n) = 2(n+1)\sqrt{n}(3/\sqrt{2})^n$ on the right-hand side. The structure of Lenstra's algorithm is a search tree branching on hyperplanes derived from lattice basis reduction.

H. W. Lenstra, Jr. constructively proved that for any fixed dimension n , the integer linear programming problem can be solved in time polynomial in the size of A and b . The procedure goes as follows. Start the procedure by finding an ellipsoidal approximation \mathcal{E} of the full-dimensional polyhedron \mathcal{P} and apply a nonsingular linear transformation τ to \mathcal{E} to transform \mathcal{E} into a unit ball. Next use the basis reduction method to find either an integral feasible vector or a specific nonzero direction $c \in \mathbb{Z}^n$. In the later case, for each integer k such that $\lceil \min\{c^T x : x \in \mathcal{P}\} \rceil \leq k \leq \lfloor \max\{c^T x : x \in \mathcal{P}\} \rfloor$, search if $\mathcal{P} \cap \{x : c^T x = k\}$ contains an integral vector. Work on subproblems

with dimension at least one less than the dimension of its predecessor by branching on these hyperplanes, and the number of hyperplanes is bounded. In this way a search tree, at most n deep, is built. For fixed n , this is a polynomial-time algorithm.

A framework of the Lovász version [30] of the Lenstra algorithm is described as follows.

Step 1. Find a maximum volume ellipsoid $\mathcal{E}(s^0, E)$ inscribed in the full-dimensional polytope \mathcal{P} , where s^0 is the center of \mathcal{E} and E is a positive definite matrix.

Construct a nonsingular linear transformation τ of the vector space \mathcal{R}^n , such that $\tau\mathcal{E}$ is a unit ball, i.e. $\tau = E^{-1}$. Let $L = \tau(\mathcal{Z}^n)$ be the image of the standard lattice \mathcal{Z}^n under the nonsingular transformation τ , and let $s = \tau(s^0)$.

Step 2. Use an algorithm of lattice basis reduction to construct a lattice point $p \in L$ “near” s and a dual-basis direction d such that the distance of s from the two lattice hyperplanes $d^T x = \lfloor d^T s \rfloor$ and $d^T x = \lceil d^T s \rceil$ is at least $(\sqrt{2}/3)^n \|p - s\|$. Let $a = \tau^{-1}p$ and $c = \tau^T d$.

Step 3. consider two cases:

case 1. $a \in \mathcal{P}$, then $a \in \mathcal{P} \cap \mathcal{Z}^n$ and exit. We have found the integral vector a in \mathcal{P} .

case 2. $a \notin \mathcal{P}$, i.e. $\|p - s\| > 1$, then consider the polytopes $\mathcal{P} \cap \{x : c^T x = k\}$ in \mathcal{R}^{n-1} for all $k \in \mathcal{Z}$ such that $\lceil \min\{c^T x : x \in \mathcal{P}\} \rceil \leq k \leq \lfloor \max\{c^T x : x \in \mathcal{P}\} \rfloor$. Reset n accordingly and repeat the whole procedure on the new polytopes in lower dimension.

Three problems need to be solved in this algorithm. On each node of the search tree, ellipsoid approximation and lattice basis reduction are the two main steps. Another issue in Step 3 case 2 is how to derive a full-dimensional polyhedron after hyperplanes are added.

The framework gives only a big picture of the algorithm. How to obtain an ellipsoid approximation in Step 1 is discussed in Chapter 3. In case 2 of Step 3, the question how to derive a full-dimensional polytope from a non-full-dimensional one will be addressed in Chapter 5. Assume that the maximum volume ellipsoid inscribed in a polytope can be found, we first explain how the lattice basis reduction algorithm finds a flat direction of the ellipsoid in the next section.

4.3 Lattices and Basis Reduction

In this section, we summarize the main results on lattice basis reduction applied in integer programming including the concepts of lattice and dual lattice, the reduced basis and its properties, and the relation between the lattice basis reduction and Lenstra's algorithm for integer programming.

4.3.1 Lattices and Dual Lattices

H. W. Lenstra, Jr. [28] introduced the concept of the geometry of numbers into the theory of integer programming and made a connection between these two independently developed fields. The concept of lattice comes from the geometry of numbers [7] developed by Minkowski, which investigates the intersections of lattice points with convex sets. The standard lattice is \mathbb{Z}^n , i.e. all of the integer points in \mathbb{R}^n . Two branches of mathematics dealing with lattice points in convex bodies are integer linear programming and the geometry of numbers. The lattice representation

adds geometric insights into solving *ILP* problems. The integer feasibility problem becomes the problem of deciding whether or not a polytope contains a lattice point.

Definition 4.1 (*lattice basis*) A subset $\mathcal{L} \subset \mathcal{R}^n$ is called a *lattice* if there exist linearly independent vectors b_1, b_2, \dots, b_l in \mathcal{R}^n such that

$$\mathcal{L} = \left\{ \sum_{i=1}^l m_i b_i : m_i \in \mathcal{Z}, 1 \leq i \leq l \right\}$$

The set of vectors b_1, b_2, \dots, b_l is called a *lattice basis*. The lattice \mathcal{L} is also denoted by $\mathcal{L}(b_1, \dots, b_l)$.

Definition 4.2 (*unimodular matrix*) A nonsingular matrix U is called *unimodular* if U is integral and has determinant ± 1 .

Three operations on a matrix are called *elementary column operations*:

- exchanging two columns;
- multiplying a column by -1;
- adding an integral multiple of one column to another column

Theorem 4.1 ([42]) The following are equivalent for a nonsingular rational matrix U of order n :

1. U is unimodular;
2. U^{-1} is unimodular;
3. the lattice generated by the columns of U is \mathcal{Z}^n ;
4. U comes from the identity matrix by elementary column operations.

Corollary 4.1 ([42]) Let M and M' be nonsingular matrices. The following are equivalent:

1. the columns of M and those of M' generate the same lattice;
2. M' comes from M by elementary column operations;
3. $M' = MU$ for some unimodular matrix U (i.e. $M^{-1}M'$ is unimodular).

Let B be a nonsingular matrix with column vectors b_1, \dots, b_l . The number l is called the *rank* of the lattice $\mathcal{L}(b_1, \dots, b_l)$. If b'_1, b'_2, \dots, b'_l is another basis for \mathcal{L} , then $b'_i = \sum_{j=1}^l u_{ij} b_j$ for some $l \times l$ -unimodular matrix $U = (u_{ij})_{1 \leq i, j \leq l}$ with integral coefficients and $|\det(U)| = 1$. It follows that the positive real number $|\det(b_1, b_2, \dots, b_l)|$ depends only on \mathcal{L} instead of on the choice of its basis. The *determinant of \mathcal{L}* is defined as

$$\det(\mathcal{L}) = |\det(b_1, b_2, \dots, b_l)|,$$

where the b_i is written as column vectors. Geometrically, the determinant of a lattice is the common volume of those parallelohedra spanned by basis vectors. This leads to *Hadamard's inequality*

$$\det(\mathcal{L}) \leq \|b_1\| \cdots \|b_l\|,$$

where $\|b_i\|$ denotes the Euclidean norm of the vector b_i and the equality holds if and only if the basis b_1, b_2, \dots, b_l is orthogonal. It is obvious that not every lattice has an orthogonal basis.

A closely related concept is dual lattice corresponding to a lattice. The dual lattice basis vector d_i gives the normal direction of the subspace $\{\sum_{j \neq i} \alpha_j b_j, \alpha_j \in \mathcal{R}\}$.

Definition 4.3 (*dual lattice*) For every full-rank n lattice \mathcal{L} , its *dual lattice*, denoted by \mathcal{L}^* , is defined by

$$\mathcal{L}^* = \{x \in \mathcal{R}^n : y^T x \in \mathcal{Z}, \text{ for every } y \in \mathcal{L}\}.$$

Lemma 4.1 (*dual basis [7]*) If (b_1, \dots, b_n) is a basis of \mathcal{L} , then the vectors c_1, \dots, c_n defined by

$$c_i^T b_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j \end{cases}$$

form a basis for the dual lattice $\mathcal{L}^* = \mathcal{L}(c_1, \dots, c_n)$, called the *dual basis* of (b_1, \dots, b_n) .

If a lattice \mathcal{L} is generated by the columns of a nonsingular matrix B , then \mathcal{L}^* is the lattice generated by the rows of B^{-1} . The dual of the standard lattice \mathbb{Z}^n is itself, and the dual of a rational lattice is rational. The determinant of the dual lattice is $\det(\mathcal{L}^*) = 1/\det(\mathcal{L})$.

4.3.2 Lattice Basis Reduction

While a lattice has different bases, some are nicer or simpler than others. The basis whose elements are relatively short (for the corresponding norm) is so-called reduced. A reduced basis is not too far from being orthogonal because the bases of a lattice have the same determinant. The notion of a reduced basis is quite old, but no really satisfactory algorithm was known to find such a basis in a reasonable time. Lenstra, Lenstra, and Lovász [27] proposed a polynomial-time lattice basis reduction algorithm (LLL-algorithm) and made it possible to find a reduced basis algorithmically. Thereafter lattice basis reduction has played an important role in the theory of integer programming. Lattice basis reduction has also been used in other fields such as computational algebra and number theory.

Given a basis B , it is well known how to obtain orthogonal vectors by applying Gram-Schmidt orthogonalization. However, the Gram-Schmidt vectors do not necessarily belong to the lattice generated by B while they do span the same real vector

space as B . Therefore, Gram-Schmidt orthogonalization is used as a “reference” for the LLL basis reduction algorithm. The Gram-Schmidt orthogonalization process gives an orthonormal basis in a Euclidean vector space. For an orthogonal basis, the same procedure works, except there is no need to normalize the vectors.

Proposition 3 (*Gram-Schmidt*) Let b_1, b_2, \dots, b_n be linearly independent vectors. Define by induction:

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \quad (1 \leq i \leq n),$$

where

$$\mu_{i,j} = b_i^T b_j^* / b_j^{*T} b_j^*. \quad (1 \leq j < i \leq n).$$

then the b_i^* form an orthogonal basis of the space $\text{Span}\{b_1, b_2, \dots, b_n\} = \{\sum_{j=1}^n \alpha_j b_j : \alpha_j \in \mathcal{R}\}$; b_i^* is the projection of b_i on the orthogonal complement of $\text{Span}\{b_1, b_2, \dots, b_{i-1}\} = \text{Span}\{b_1^*, b_2^*, \dots, b_{i-1}^*\}$; the matrix $(\mu_{i,j})$ whose columns gives the coordinates of the b_i^* in terms of the b_i is an upper triangular matrix with diagonal terms equal to 1. In particular, if $\det(\mathcal{L})$ is the determinant of the lattice $\mathcal{L}(b_1^*, b_2^*, \dots, b_n^*)$, $\det(\mathcal{L}) = \prod_{1 \leq i \leq n} \|b_i^*\|$.

Hermite proved that every lattice \mathcal{L} has a basis b_1, b_2, \dots, b_l such that

$$\prod_{1 \leq i \leq l} \|b_i\| \leq \text{cons}(l) \det(\mathcal{L}),$$

while $\text{cons}(l)$ is a constant depending only on the rank l of the lattice \mathcal{L} . Such a basis is called reduced in terms of the geometry of numbers and may be viewed as an approximation of an orthogonal basis.

The idea of the LLL basis reduction algorithm is to find a basis which consists of relatively short vectors and whose Gram-Schmidt orthogonalization consists of reasonably long vectors.

Definition 4.4 (*reduced basis*) Let b_1, b_2, \dots, b_n be a basis of a lattice L . Find an orthogonal basis $b_1^*, b_2^*, \dots, b_n^*$ using the Gram-Schmidt orthogonalization process. The basis b_1, b_2, \dots, b_n is called a *LLL reduced basis* if

$$|\mu_{i,j}| \leq \frac{1}{2} \quad \text{for } 1 \leq j < i \leq n$$

and

$$\|b_i^* + \mu_{i,i-1}b_{i-1}^*\|^2 \geq \frac{3}{4}\|b_{i-1}^*\|^2 \quad \text{for } 1 \leq i \leq n.$$

Notice that the vectors $b_i^* + \mu_{i,i-1}b_{i-1}^*$ and b_{i-1}^* are the projections of b_i and b_{i-1} on the orthogonal complement of $\text{Span}\{b_1, \dots, b_{i-2}\}$. The constant $\frac{3}{4}$ in the definition is arbitrarily chosen and may be replaced by any fixed real number in the interval $[1/4, 1]$. The following theorem [27] describes the properties of such a reduced basis.

Theorem 4.2 (*LLL reduced basis properties*) Let b_1, b_2, \dots, b_n be an LLL reduced basis of a lattice \mathcal{L} . then

1. $\det(\mathcal{L}) \leq \prod_{i=1}^n \|b_i\| \leq 2^{n(n-1)/4} \det(\mathcal{L})$,
2. $\|b_j\| \leq 2^{(i-1)/2} \|b_i^*\|$, for $1 \leq j \leq i \leq n$.
3. $\|b_1\| \leq 2^{(n-1)/4} \det(\mathcal{L})^{1/n}$,
4. for every $x \in \mathcal{L}$ with $x \neq 0$,

$$\|b_1\| \leq 2^{(n-1)/2} \|x\|,$$

5. generally, for any linearly independent vectors $x_1, \dots, x_t \in \mathcal{L}$,

$$\|b_j\| \leq 2^{(n-1)/2} \max(\|x_1\|, \dots, \|x_t\|) \quad \text{for } 1 \leq j \leq t.$$

The vector b_1 in a reduced basis is an approximation of the shortest nonzero vector (i.e. the nonzero lattice point with the shortest norm) in \mathcal{L} . Recently, Ajtai [2] proved

that the shortest lattice vector problem with L_2 norm is NP-hard from randomized reductions. It was already known to be NP-hard if the Euclidean norm is replaced by the maximum norm [47].

A brief outline of the LLL basis reduction algorithms is as follows. For precise details, please refer to [27]. First compute the Gram-Schmidt vectors b_j^* , $1 \leq j \leq n$ and the coefficient of μ_{jk} , $1 \leq k \leq j \leq m$. Initialize the counter $i = 2$. Check the first condition of a reduced basis. If the first condition is violated, replace b_i by $b_i - \lceil \mu_{i,i-1} \rceil b_{i-1}$, where $\lceil \mu_{i,i-1} \rceil = \lceil \mu_{i,i-1} - 1/2 \rceil$. Such a step is also called *size reduction* to obtain $\mu_{i,i-1} \leq 1/2$. Update $\mu_{i,i-1}$. Then check the second condition of a reduced basis for $j = i$. If the second condition is not satisfied, exchange b_i and b_{i-1} , and update the corresponding Gram-Schmidt vectors and coefficients μ_{jk} . If $i > 2$, let $i := i - 1$ and do necessary size reductions to satisfy $|\mu_{im}| \leq 1/2$ for $m = i - 2, i - 3, \dots, 1$. If $i = n$, stop. Finally, set $i := i + 1$. The running time of LLL basis reduction algorithm is $O(n^6(\log \beta)^3)$, where $\beta \in R$, $\beta \geq 2$ such that $\|b_j\|^2 \leq \beta$ for $1 \leq j \leq n$.

LLL basis reduction algorithm performs a series of elementary column operations on an initial basis B for a given lattice and produces a so-called *reduced basis* \tilde{B} such that the reduced basis vectors $\tilde{b}_1, \dots, \tilde{b}_l$ are short and nearly orthogonal, and such that \tilde{b}_1 is an approximation of the shortest vector in the lattice. From Corollary 4.1, $\tilde{B} = BU$ for some unimodular matrix U .

Making use of LLL reduced basis, Babai [4] proved that a rounding-off procedure can find a lattice point p “close” to a given vector s , and find a vector $d \in \mathcal{L}^*$ such that $d(\mathcal{Z}, d^T s) / \|d\|$ is “big” as a lower bound of the minimum distance of s from the vectors in \mathcal{L} , where $d(\mathcal{Z}, d^T s) = \min\{d^T s - \lfloor d^T s \rfloor, \lceil d^T s \rceil - d^T s\}$. An application of Babai’s following theorem improves Lenstra’s algorithm.

Theorem 4.3 (*Babai, 1985*) Given a lattice $\mathcal{L}(b_1, \dots, b_n) \in \mathcal{Q}^{n \times n}$ and a vector $s \in \mathcal{Q}^n$, we can find in polynomial time a vector $p \in \mathcal{L}(b_1, \dots, b_n)$ and another vector $d \in \mathcal{L}^*(b_1, \dots, b_n)$ such that

$$\|p - s\| \leq (3/\sqrt{2})^n d(Z, d^T s) / \|d\|.$$

Hence also

$$\|p - s\| \leq (3/\sqrt{2})^n d(\mathcal{L}, s),$$

with $d(\mathcal{L}, s)$ denoting the minimum distance of s from the vectors in \mathcal{L} .

Remark: The vector $d \in \mathcal{L}^* = \mathcal{L}(d_1, \dots, d_n)$ can be chosen to be the d_j in the dual basis such that $d_j = \arg \max_{d_i} \{d(\mathcal{L}, d_i^T s) / \|d_i\|\}$. From the theorem, $\|d\| \leq (3/\sqrt{2})^n \|p - s\|^{-1}$ holds, which is useful to prove a flat direction.

The lattice concept, LLL reduced basis reduction and Babai's algorithm together serve for the key idea of Lenstra's algorithm: either an integral vector a is found or a nonzero integral vector c is found such that

$$\max\{c^T x : x \in \mathcal{P}\} - \min\{c^T x : x \in \mathcal{P}\} \leq 2(n+1)\sqrt{n}(3/\sqrt{2})^n$$

in polynomial time. It is enough to show that a flat direction is found in the second case $\|p - s\| \geq 1$ to see the connection between lattice basis reduction and the flat direction of the polytope.

Löwner and John [17] proved that for any convex body $K \in \mathcal{R}^n$ there exist a pair of ellipsoids $(\mathcal{E}, \mathcal{E}')$ such that $\mathcal{E} \subseteq K \subseteq \mathcal{E}'$, \mathcal{E} and \mathcal{E}' are concentric, and \mathcal{E} arises from \mathcal{E}' by shrinking a factor $1/n$. If \mathcal{E} is the maximum volume ellipsoid inscribing K , or \mathcal{E}' is the minimum volume ellipsoid circumscribing K , such a pair, which is called a *Löwner-John pair* for K , can be obtained. When the shrinking factor is $1/((n+1)\sqrt{n})$, the pair is called a *weak Löwner-John pair* for K . Lovász [30] proved that a weak

Löwner-John pair for a convex body K can be computed in polynomial time. These results have become the classic on the ellipsoidal approximation problem.

Let $\mathcal{E}(s_0, E)$ be the maximum volume ellipsoid inside the polytope \mathcal{P} and $c = \tau^T d$ where d is chosen as in Babai's theorem. Denote $S(y, r)$ as the n -dimensional ball with center y and radius r . An upper bound on the width of the polytope \mathcal{P} in the direction c is proved as follows:

$$\begin{aligned}
& \max\{c^T x : x \in \mathcal{P}\} - \min\{c^T x : x \in \mathcal{P}\} \\
& \leq \max\{c^T x : x \in \mathcal{E}'\} - \min\{c^T x : x \in \mathcal{E}'\} \\
& = 2 \max\{c^T (x - s_0) : x \in \mathcal{E}', s_0 \in \mathcal{E}'\} \\
& = 2 \max\{d^T (y - s) : y \in \tau(\mathcal{E}') = S(s, (n+1)\sqrt{n})\} \\
& \leq 2\|d\|(n+1)\sqrt{n} \\
& < 2(n+1)\sqrt{n}(3/\sqrt{2})^n.
\end{aligned}$$

where the last inequality follows from Theorem 4.3.

The third equation holds because the linear transformation transform the ellipsoid into a ball. Thus the direction constructed in Lenstra's algorithm is a flat direction.

Recently several new variants of the LLL basis reduction algorithm have been developed and a number of implementation variants have been suggested. The paper by Schnorr and Euchner [41] gives a detailed overview. So far the available implementations of LLL lattice basis reduction and improved practical algorithms can solve problems up to dimension 125. A bottleneck for the LLL-reduction algorithm is the required exact arithmetic on large integers. This limitation has been transferred to our implementation since we have used the LLL-reduction algorithm as a subroutine on each node of the search tree.

Some useful libraries of LLL basis reduction algorithm are available on the internet. Two of them are LiDIA – a C++ library for computational number theory [29],

developed at TH Darmstadt, and NTL – a library for doing number theory [43], developed by V. Shoup, University of Wisconsin-Madison. The LiDIA library is used to do the lattice basis reduction in my implementation of Lenstra’s algorithm.

4.4 Generalized Basis Reduction Algorithm

As we mentioned in the introduction, based on Lenstra’s idea of branching on hyperplanes. Lovász and Scarf [31] proposed the so-called generalized basis reduction algorithm for solving the general integer programming feasibility problem. W. Cook *et al* [9] implemented the generalized basis reduction algorithm and solved some integer network design instances. In this section we describe the concepts behind the generalized basis reduction algorithm and offer a brief comparison between Lenstra’s algorithm and the generalized basis reduction algorithm.

As in Lenstra’s algorithm, the focus of the generalized basis reduction algorithm is on constructing a thin direction of the polytope to branch on. We already explained the construction of a thin direction for Lenstra’s algorithm in previous sections. We now review how a thin direction is constructed in the generalized basis reduction algorithm.

By definition, the width of a polytope \mathcal{P} in a given direction c is

$$\begin{aligned} F(c) &= \max\{c^T x : x \in \mathcal{P}\} - \min\{c^T x : x \in \mathcal{P}\} \\ &= \max\{c^T(x - y) : x \in \mathcal{P}, y \in \mathcal{P}\}. \end{aligned}$$

To construct a nonzero integral direction with a small $F(\cdot)$ value, Lovász and Scarf defined a family of distance functions $F_1(\cdot), \dots, F_n(\cdot)$ with respect to the polytope $\mathcal{P} = \{x : Ax \leq b\}$ and a given basis b_1, \dots, b_n for the integral lattice \mathcal{Z}^n as

$$F_i(c) = \min_{\alpha_1, \dots, \alpha_{i-1}} F(c + \alpha_1 b_1 + \dots + \alpha_{i-1} b_{i-1}).$$

Applying the duality theorem for linear programming to the definition of distance functions $F_i(\cdot)$, we have

$$F_i(c) = \max\{c^T(x - y) : Ax \leq b, Ay \leq b, b_j^T(x - y) = 0, j = 1, \dots, i - 1\}.$$

Therefore the function $F_i(\cdot)$ can be evaluated by solving a linear program.

A basis b_1, \dots, b_n is called a *generalized reduced basis* with respect to distance functions $F_i(\cdot)$ if the following two conditions are satisfied for $i = 1, \dots, n - 1$

$$F_i(b_{i+1} + \mu b_i) \geq F_i(b_{i+1}) \quad \text{for all integers } \mu, \quad (4.1)$$

$$F_i(b_{i+1}) \geq (1 - \epsilon)F_i(b_i). \quad (4.2)$$

where $\epsilon \in (0, 1/2)$ is a fixed constant. For the detailed procedure of the generalized basis reduction, please refer to Lovász and Scarf [31] and W. Cook *et al* [9]. The nice property of a generalized reduced basis is that the first vector b_1 in such a basis is an approximation to the shortest nonzero lattice point. This is the way that the generalized basis reduction algorithm constructs a thin direction to approximate the minimum width of the polytope. The term “generalized” indicates that this algorithm generalizes the basis reduction algorithm by Lenstra. Lenstra and Lovász [27].

Instead of using ellipsoidal approximations, the generalized basis reduction algorithm constructs a thin direction directly from the polytope. Because of the distance function evaluations involved in the generalized basis reduction procedure, a large number of linear programming problems need to be solved. Solving so many LP problems causes considerably more computation than the basis reduction algorithm working directly on the basis vectors.

The difference between Lenstra’s algorithm and the generalized basis reduction algorithm lies in the different methods to capture a thin direction to branch on. Lenstra’s algorithm applies ellipsoidal approximations to the polytope and uses the

information of the ellipsoids to approximate the shortest lattice vector (i.e. a thin direction) through a lattice basis reduction procedure. While the ellipsoidal approximation of a polytope is generally considered to be good, the quality of the approximation may not always be as high as we wish (after all, some information about the original polytope is lost). Therefore, the tradeoff between Lenstra's algorithm and the generalized basis reduction procedure is the uncertainty in the quality of ellipsoidal approximation on one hand and the high cost of a large number of distance function evaluations (i.e. linear programs) on the other hand. A computational comparison of the two algorithms appears to be of value but is beyond the scope of the thesis.

The common feature of Lenstra's algorithm and the generalized basis reduction is that both are polynomial algorithms for integer programming feasibility problems with a fixed number of variables. In contrast, the LP-based branch-and-bound algorithm for integer programming is not a polynomial algorithm. The idea of Lenstra's algorithm is to construct a thin direction of the polytope and to branch on all possible hyperplanes intersecting the polytope. The generalized basis reduction algorithm shares the same idea. A thin direction is crucial here because it guarantees that only a finite number of lattice hyperplanes exist in the thin direction that intersect the polytopes. Thus there is only a finite number of nodes are generated on the search tree. (Since we are searching for integer points, only lattice hyperplanes need to be branched on.)

Chapter 5

Computational Study of Lenstra's Algorithm

Lenstra's algorithm [28] is a polynomial-time algorithm for integer programming feasibility problems when the number of variables is fixed. Utilizing our results on maximum volume ellipsoidal algorithms described in Chapter 3 and the availability of lattice basis reduction software, we have implemented a Lovász version [30] of Lenstra's algorithm. We used our code in solving some medium-sized integer programming feasibility problems. In this chapter, we will focus on implementation issues and computational properties of Lenstra's algorithm.

5.1 Non-full-dimensional Polytopes

The two assumptions of Lenstra's algorithm are

1. The polyhedra given by linear inequalities $Ax \leq b$ is bounded;
2. The polytope is full-dimensional (i.e. has a positive volume).

The first requirement is easy to meet by adding proper bounds. When the second condition is not satisfied, how to derive a full-dimensional polytope from a non-full-dimensional case becomes an issue to address in our implementation.

In Lenstra's paper [28], the assumption that the polytope is full-dimensional is justified by constructing vertices of the polytope and employing the Hermite normal form algorithm. We would like to avoid constructing vertices in our implementation since it is computationally complicated and unnecessary for our purpose.

In a more general setting, the problem becomes much harder when the polytope is given by a strong oracle or a strong violation oracle. Grötschel, Lovász and

Schrijver [14] discussed this question of full-dimensionality. However, they considered polytopes given by a strong oracle and assumed that all of the vertices of the polytope are 0,1-vectors. Their method consists of a combination of the ellipsoid method with Diophantine approximation to determine the affine hull of the polytope. Edmonds, Lovász and Pulleyblank [12] proposed an algorithm to get around non-full-dimensionality by maintaining one list of affinely independent vectors inside the polytope and another list of linearly independent hyperplanes containing the polytope under the assumption that the polytope is given by a strong violation oracle.

While we consider a polytope given by a system of linear inequalities, the above methods are not efficient for our purpose. Our solution to the non-full-dimension problem consists of the following three steps while more details will be given in the next three subsections.

- Step 1. Identify a non-full-dimensional case and implicit equalities;
- Step 2. Reduce implicit equalities to its Hermite normal form;
- Step 3. Project the non-full-dimensional polytope into a lower full-dimensional polytope.

5.1.1 Identify Non-full-dimensional Cases and Implicit Equalities

The necessary and sufficient condition that a polytope given by $Ax \leq b$ is not full-dimensional is that the system of linear inequalities $Ax \leq b$ must have implicit equalities. Therefore, we need only to identify implicit equalities in the linear system to determine whether the given polytope is full-dimensional or not.

We are using a property of barrier algorithms in our method to determine the implicit equalities. For the purpose of identifying implicit equalities, we construct an auxiliary problem $\max\{t : Ax + te \leq b, t \geq 0\}$ with m -vector e of all ones and

solve it by the CPLEX barrier method for LP problems. If the maximum value $t^* > 0$, then the corresponding polytope is full-dimensional and the optimal solution x^* is an interior point such that $Ax^* < b$. If $t^* = 0$, then the polytope is non-full-dimensional and we can identify the implicit equalities. In our implementation, we consider the i -th inequality in the linear system $Ax \leq b$ to be an implicit equality if $b_i - (Ax^*)_i \leq 1.0e - 12$.

A full-dimensional polytope in lower dimensional space can be obtained by a projection onto the orthogonal complement of the subspace defined by the implicit equalities. However, a direct projection doesn't work here because we need to keep the property that an integer point corresponds to an integer point after the projection. This is why Hermite normal form plays a role in the projection.

5.1.2 Hermite Normal Form and Linear Diophantine Equations

Once the implicit equalities are identified, we can focus on the subsystem of linear equalities, which are also called linear Diophantine equations when the variables are required to be integers. We give the definition and related results on the Hermite normal form [42] in this section.

Definition 5.1 (*Hermite normal form*) A matrix of full row rank is said to be in *Hermite normal form* if it has the form $[0 \ N]$, where N is a nonsingular, lower triangular, nonnegative matrix, in which each row has a unique maximum entry, which is located on the main diagonal of N .

A rational matrix of full row rank can be reduced into Hermite normal form by a series of elementary column operations.

The uniqueness of the Hermite normal form of a rational full-rank matrix is given in the following theorem, which is followed by a corollary.

Theorem 5.1 ([42]) The Hermite normal form $[0 \ N]$ of a rational matrix A of full row rank has size polynomially bounded by the size of A . Moreover, there exists a unimodular matrix U with $AU = [0 \ N]$, such that the size of U is polynomially bounded by the size of A .

In the theory, the size of a matrix is measured by the number of bits necessary for storing the whole matrix.

Corollary 5.1 ([42]) Given a system of rational linear equations, we can determine if it has an integral solution, and if so, find one, in polynomial time.

The LiDIA library [29] has six different implementations of the Hermite normal form reduction. We called one of its functions to convert a full row rank matrix into its Hermite normal form and to return the unimodular transformation matrix. One drawback of Hermite normal form is that the elements in the unimodular matrix may become excessively large, though have not encountered this difficulty on our test problems.

Recall that our question is whether the linear system $Ax \leq b$ has an integer feasible solution or not. The question remains the same for the subsystem of all implicit equalities, denoted by $A^-x = b^-$. Based on the corollary, if the subsystem or the linear Diophantine equations have no integer solution, the whole linear system has no integer feasible solution. If the subsystem has an integer solution, we can use the unimodular transformation matrix U to do a projection with integral requirements. The reason that the Hermite normal form plays a role in keeping the integrality is because of the nice properties of the unimodular transformation matrix. We will explain how to do the projection in the next section.

5.1.3 Projection Into Lower Full-Dimensional Case

Consider the linear system of inequalities $Ax \leq b$. With the help of the solution to the auxiliary problem mentioned earlier, the system can be rewritten as $\{A^+x \leq b^+, A^-x = b^-\}$.

Let us denote the linearly independent rows of A^- as A_e and the corresponding right hand side as b_e . Apply the Hermite normal form transformation on the subsystem $A_e x = b_e$, and we obtain $A_e U = [0 \ B]$ and $A_e x = A_e U U^{-1} x = b_e$. We also check whether the left linear dependent rows and the associated right hand sides are consistent with $A_e x = b_e$.

Let $y = U^{-1} x$, the system $A_e x = b_e$ becomes $[0 \ B] y = b_e$. Partition the variable y into

$$y = \begin{bmatrix} y_0 \\ y_B \end{bmatrix}$$

corresponding to the partition $[0 \ B]$. Then solve y_B from $B y_B = b_e$ to obtain y_B .

We do the same transformation on the subsystem $A^+x \leq b^+$ to obtain $A^+U y \leq b^+$. Denote the block of A^+U corresponding to y_0 as A_0^+ and the block of A^+U corresponding to y_B as A_B^+ . The subsystem becomes $A_0^+ y_0 + A_B^+ y_B \leq b^+$. Substitute y_B from the above step and obtain a full dimensional system

$$A_0^+ y_0 \leq b^+ - A_B^+ y_B$$

with a new unknown variable y_0 .

To recover the solution, we compute $x = U y$ for a given y_0 .

With the three steps above, we can solve the problem of non-full-dimensional polytopes. Based on our work on the ellipsoidal approximation and the basis reduction functions from the LiDIA library, we have built a version of Lenstra's algorithm. The implementation details are discussed in the next section.

5.2 Implementation

Our implementation of Lenstra's algorithm aims at solving the integer programming feasibility problem using the ellipsoidal approximation of polytopes and the lattice basis reduction. The input is a linear inequality system $Ax \leq b$, which defines a bounded polyhedron (or a polytope). The output is either that an integer feasible solution is found or that there is no integer feasible solution for the given system. The input data are read through a CPLEX function from an LP or MPS file.

A Different Search Tree

The overall picture of the search procedure is a depth-first search (DFS) tree shown as in Figure 5.1. We use a *stack* to store the nodes on the tree. The root of the search tree is the original polytope. This search tree is different from the branch-and-bound tree for feasibility problems in two ways: one is that this search tree grows only as deep as n while the branch-and-bound tree for feasibility can grow much deeper depending on the input data; the other is that it branches on hyperplanes instead of on variables as in the branch-and-bound tree. This novel way of searching makes Lenstra's algorithm polynomial for a fixed number of variables n whereas the branch-and-bound method is not a polynomial algorithm even for a fixed number of variables.

On Each Node of the Search Tree

We construct a structure node to store its related information for each node on the search tree. The structure node includes components for the matrix A and the vector b , branching bounds klo and kup , a transformation matrix T to bring the solution from the transformed space into the original space. The path from the root node to

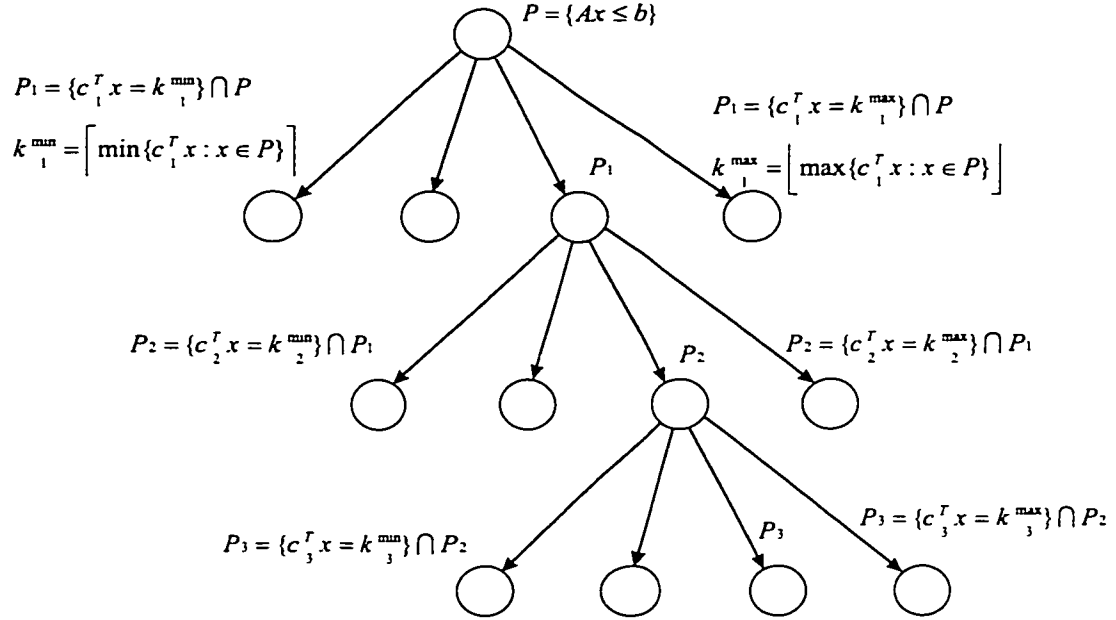


Figure 5.1 Search Tree of Branching on Hyperplanes

the current node is recorded in a vector variable *y_{path}* and another vector *y_{swap}* is used to remember the associated swap operations.

For each node of the tree, the first step is to determine whether the polytope is full-dimensional by solving an auxiliary LP problem $\max\{t : Ax + te \leq b\}$ through the CPLEX barrier method. If the polytope is not full-dimensional, we use the Hermite normal form to project it onto a lower but full-dimensional polytope without affecting the integrality (the correspondence from integers to integers). If the polytope is full-dimensional, the optimal solution to the auxiliary problem gives a starting strictly interior point for the primal-dual interior point algorithm. Once a polytope is found or made to be full-dimensional, a procedure to delete redundant constraints from the linear system defining the polytope, which is called a preprocessing, will reduce

the cost of solving the subproblem. Currently we have built a simple version of preprocessing and we expect it to be further strengthened in the future.

The algorithm is a recursive search through the nodes. Either an integer feasible solution is found on some node and the algorithm stops, or the entire tree is searched to ensure that there exists no feasible solution for the problem. The main subroutine

Lennode(curnode, stack, topi, ypath, yswap, found, nodes, branopti)

is a subroutine to do operations on the current node and to record its path. The subroutine *Lennode* classifies all nodes as four cases. In the first case, if the current node is a leaf node (i.e. 1-dimensional), solve all of its leaf siblings and pop up the top of the stack as the current node. In the second case, if all of the children of the current node are searched, check whether the current node has siblings. In the third case, if it has other siblings, move to its immediate sibling and update the current node; otherwise pop up its parent node from the stack as the updated current node.

In the fourth case, if the current node hasn't been explored, major steps are taken on it. First, do preprocessing to eliminate redundant inequalities in the linear system and solve an auxiliary LP problem $\max\{t : Ax + te \leq b, t \geq 0\}$ to determine whether the polytope is full dimensional. If the polytope is not full-dimensional, apply a Hermite normal form to project the polytope onto a lower but full-dimensional polytope, and remember the transformation matrix U as explained in section 1. If the polytope $\mathcal{P} = \{x : Ax \leq b\}$ associated with the current node is already full dimensional, proceed with the following four steps.

- Step 1. Ellipsoidal Approximation and Linear Transformation.** Find an approximation to the maximum-volume, inscribing ellipsoid $\mathcal{E}(x, E) = \{y \in \mathcal{R}^n : y = x + Es \text{ and } \|s\| \leq 1\}$, construct a nonsingular linear transformation $\tau = E^{-1}$ based on the ellipsoid matrix E , and the center of the ellipsoid is transformed into a vector $s = \tau(x)$. The standard lattice Z^n is transformed into the lattice $L = \tau(Z^n)$.
- Step 2. Lattice Basis Reduction and Close Lattice Point Procedure.** Find a reduced basis B of lattice L and round the transformed center s to a close lattice point $p \in L$ using the reduced basis B .
- Step 3. If an Integer Point is Found, Stop.** If the lattice point p is inside the transformed polytope $\tau(\mathcal{P})$, restore the the original point corresponding to p , which is an integer feasible point in the original polytope, the procedure stops.
- Step 4. If the Lattice Point $p \notin \tau(\mathcal{P})$, Branch on Hyperplanes.** If the point p is not inside, choose some normal direction d from the dual basis of L , denoted by L^* , which corresponds to fewer number of hyperplanes intersecting the polytope. Then it is easy to construct an integral branching direction $c = \tau^T d$ for the polytope $P = \{Ax \leq b\}$. After the branching direction c is chosen, solve two associated LP problems $\max\{c^T x : x \in P\}$ and $\min\{c^T x : x \in P\}$ to get the bounds k^{min} and k^{max} on all of the possible hyper-planes intersecting the polytope. The last step is to add the hyperplane $c^T x = k, k \in Z \cap [k^{min}, k^{max}]$, derive a polytope in at least one dimension less and generate a new current node for the new polytope. Only if the current node has children (or needs to branch), push it into the stack and update the current node with its first children and mark in the structure of its parent node. Therefore the top of the stack is always the parent node of the current one.

The flowchart of our implementation of Lenstra's algorithm in Figure 5.2 gives a concise picture of the algorithm. We then explain how to generate child nodes and how to keep the integrality of the whole algorithm in this section.

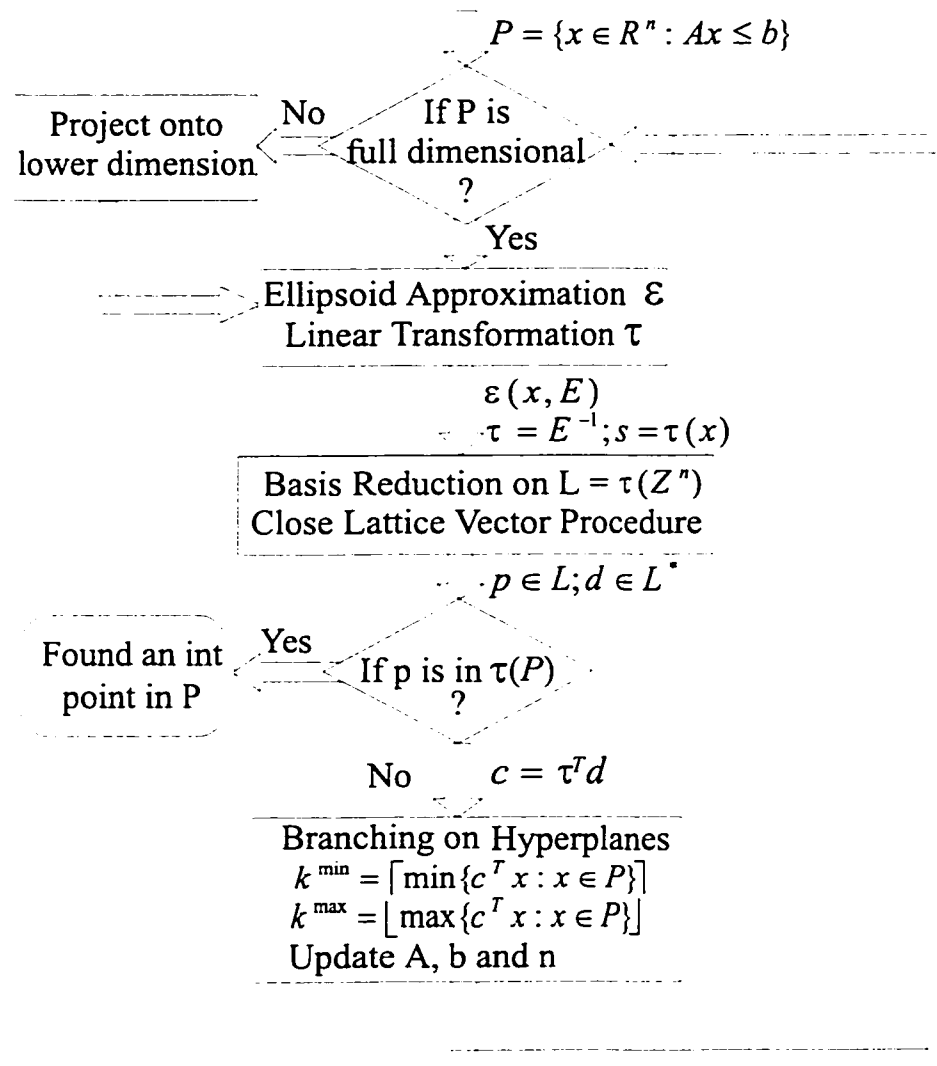


Figure 5.2 Flowchart of Lenstra's Algorithm

Generate child nodes at least one dimension less than the parent node

We generate a child node from the polytope

$$\{x : Ax \leq b, c^T x = k\}$$

as follows. Notice that the branching direction $c = \tau^T(d)$ and the hyperplane $c^T x = d^T \tau(x) = k$ while $y := \tau(x)$ is a vector on the transformed space. Making use of the relation between a lattice basis and its dual basis, we have that the component of y corresponding to dual vector d has a coordinate k on the lattice basis B since $d^T y = k$. Thus we can apply the same transformation to $Ax \leq b$ and fix the relevant component of y to k , which corresponding to a hyperplane in the original space. For computational convenience, we do a swap operation to exchange the fixed component and the last component and record the information in the variable *yswap* accordingly. After fixing one component of y and eliminating the equality $c^T x = k$, the resulting polytope is one dimension less. In the non-full-dimensional case, the dimension of its child node decreases more than 1.

In this algorithm, there is a strong similarity between sibling nodes. From one node to its sibling, only the right hand k in the hyperplane $c^T x = k$ needs to be updated. Therefore we only need to store one child node for each parent node since it is easy to modify the child node to its siblings once it has been searched. Only the path and the transformation matrix need to be remembered. Compared with the traditional branch-and-bound method for feasibility problems, Lenstra's algorithm requires more computation on branching.

Integrality

Even though the floating point computation is applied in the computation of ellipsoidal approximation and lattice basis reduction, the construction of Lenstra's algorithm has the nice property of preserving the integrality.

It is well-known that a rational unimodular matrix, as a linear transformation, maps integer vectors to integer vectors. When the polytope is non-full-dimensional, the projection matrix U in Hermite normal form is unimodular. The transformation matrix $T = E * B$ is unimodular since the transformation matrix T from E^{-1} to B should be a unimodular matrix where E^{-1} is a basis of the transformed lattice and B is a reduced basis of the same lattice. Also the branch direction c is an integral vector. One observation is that the accumulation of floating point computation error can be avoided by rounding off all the elements in the matrix $T = E * B$ at each step because of the above integrality arguments.

So far we have discussed several implementation issues of Lenstra's algorithm except the branching policy. How to choose a good direction to branch on is a key issue in the algorithm and it is also the reason why ellipsoidal approximation and lattice basis reduction can work together here. We will leave the details of different choices of branching directions in the next section.

5.3 Choices of the Branching Direction

The key idea of Lenstra's algorithm is to choose a direction such that there are only a bounded number of hyperplanes intersecting the polytope at each level of the search tree. Such a direction, also called a flat direction, is chosen among the candidates in the dual basis of a reduced basis of the lattice. A direct computation on the number of hyperplanes intersecting the polytope corresponding to all the candidates in the dual basis involves solving $2n$ linear programming problems on each node of dimension n , which is too expensive. Therefore we want to avoid a direct computation and use an easy-to-compute approximation to serve this purpose. In this section we discuss three different choices on branching directions: Babai's direction, the choice of the smallest number of intersecting hyperplanes, and the choice of the direction closest

to the smallest eigenvector of matrix E of the maximum volume ellipsoid inside the polytope.

5.3.1 Babai's Direction Choice in Close Vector Algorithm

Notice that a measure $\frac{d(Z, w^T s)}{\|w\|}$ is the shorter distance from a given point s to two adjacent hyperplanes $w^T x = \lceil w^T x \rceil$ and $w^T x = \lfloor w^T x \rfloor$, which also provides a lower bound on the distance from s to the closest lattice point in a lattice \mathcal{L} . Another observation is that the vectors in the dual basis of a lattice give the normal directions of the lattice hyperplanes. In Babai's close lattice point procedure, the direction in the dual basis $D = (d_1, d_2, \dots, d_n)$ with the largest shorter distance from the given point s to two adjacent hyperplanes $d_i^T x = \lceil d_i^T x \rceil$ and $d_i^T x = \lfloor d_i^T x \rfloor$, i.e. $d_j = \arg \max_i \left\{ \frac{d(Z, d_i^T s)}{\|d_i\|} \right\}$, is chosen. The idea behind this choice is to choose the direction in which the distance between two adjacent lattice hyperplanes from the point s is the largest among the candidates in the dual basis. Since the ellipsoidal approximation of the original polytope is transformed into a unit ball centered at the point s , this choice picks an approximation of the direction that has fewer lattice hyperplanes intersecting the transformed unit ball. Figure 5.3 gives an example on 2-dimensional case. This choice of branching direction outperforms the other on our test problems.

5.3.2 Direction of the Smallest Number of Intersecting Hyperplanes

With the maximum ellipsoid approximation of the polytope and the reduced basis of the lattice generated by the matrix of the ellipsoid at hand, why do we not approximate the number of intersecting hyperplanes on each candidate direction directly? This idea motivates our following choice of branching direction, which is to choose the direction with the smallest number of intersecting hyperplanes.

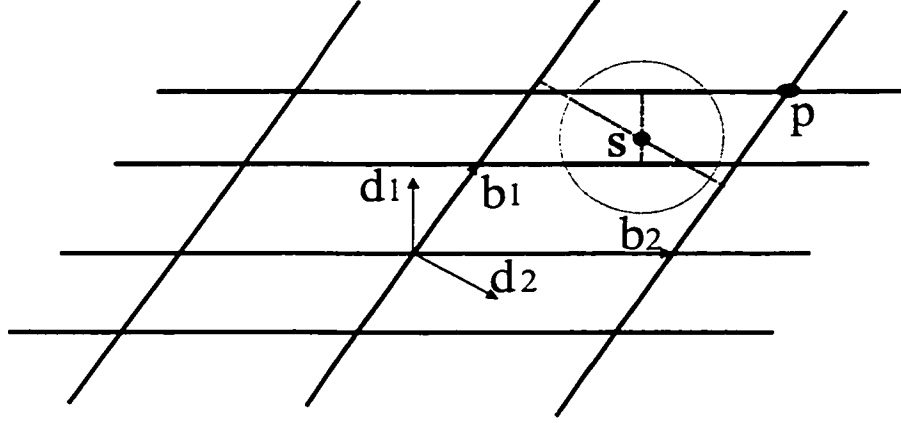


Figure 5.3 Babai's Choice of Branching Direction in 2-D

Consider the situation: the maximum inscribed ellipsoid $\mathcal{E}(x, E)$ is transformed into a unit ball. The ellipsoid which contains the polytope and is concentric to \mathcal{E} is transformed into a ball of radius R . The reduced basis of the lattice generated by the matrix of the ellipsoid is B and its dual basis is D .

Let R be the transformed ball including the transformed polytope and s be the center of the transformed center of the maximum ellipsoid inscribing the polytope. The number of hyperplanes intersecting the transformed big ball on the direction w is

$$\begin{aligned} & \left\lfloor \frac{R - \frac{d(\mathcal{Z}, w^T s)}{\|w\|}}{\frac{1}{\|w\|}} \right\rfloor + \left\lfloor \frac{R - \frac{1 - d(\mathcal{Z}, w^T s)}{\|w\|}}{\frac{1}{\|w\|}} \right\rfloor \\ &= \lfloor \|w\| R - d(\mathcal{Z}, w^T s) \rfloor + \lfloor \|w\| R - 1 + d(\mathcal{Z}, w^T s) \rfloor \end{aligned}$$

Choose the direction d in the dual basis D such that the above number is the smallest among all of the dual basis vector. Note that with this choice, we obtain a direction has the smallest number of hyperplanes intersecting with the ball R , which is concentric to the unit ball transformed from the maximum volume ellipsoid inscribing the polytope.

5.3.3 Direction Closest to the Smallest Eigenvector of E

Ellipsoids have good geometric and computational properties, which is the reason why they are chosen to approximate general convex bodies, specifically to approximate a polytope here. Recall that the ellipsoid can be formulated as

$$\mathcal{E}(x, E) = \{y \in \mathcal{R}^n : y = x + Es \text{ and } \|s\| \leq 1\}.$$

The width of the ellipsoid is the length of the shortest axis, denoted by 2λ , while λ is the smallest eigenvalue of matrix E . The direction of the shortest axis of \mathcal{E} is the eigenvector v corresponding to the smallest eigenvalue λ .

In our implementation, we have computed the maximum volume ellipsoid inside the polytope to approximate the shape of the polytope. We use the the width of the ellipsoid, which is twice of the smallest eigenvalue of matrix E , to approximate the width of the polytope. And the corresponding eigenvector is the thin or flat direction of the approximation ellipsoid. We choose the direction closest to the flat direction of the ellipsoid among the candidates(i.e. the vectors in the dual basis of the reduced basis (d_1, d_2, \dots, d_n)) by comparing the angle between them, i.e. $d_j = \arg \max_i \{ \frac{v^T d_i}{\|v\| \|d_i\|} \}$.

5.4 Results and Discussion

In this section we present some numerical results on our implementation. We describe the set of test problems and point out our comments on the algorithm and our implementation.

The code of Lenstra's algorithm is written in the C and C++ computational language. The ellipsoidal approximation subroutine makes use of CLAPACK/CBLAS [6] linear algebra library. For lattice basis reduction and Hermite normal form, we choose

the LiDIA [29] library for computational number theory. We use the linear programming optimizer of CPLEX [36] for solving LP problems involved by calling the CPLEX callable library. All the experiments were run on Sun ULTRA60 workstation with two 450-MHz UltraSPARC-II modules. We use only a single processor at a time.

Because of the computational complexity, this code is not supposed to solve integer programming feasibility problems with more than a couple of hundred variables. The advantage of this algorithm is to deal with extremely skew polytopes. Therefore it is recommended to be used when the traditional branch-and-bound method has difficulty i.e. when the branch-and-bound tree grows too large.

5.4.1 Test Problems

Three of my test problems come from the MIPLIB [37] library and the other three are from Dr. Robert Bixby's collection of difficult problems for CPLEX. The original problems are IP optimization problems. For each optimization problem, we created two feasibility problems, one feasible and another infeasible, by moving the objective function into the constraints and making use of the known bounds on the original problem.

The original IP problem is given in the format:

$$\begin{aligned}
 \min \quad & c^T x \\
 \text{s.t.} \quad & Ax \leq b \\
 & l \leq x \leq u \\
 & x \in \mathcal{Z}^n
 \end{aligned} \tag{5.1}$$

with the minimal solution x^* and the minimum $c^T x^*$.

The feasible and infeasible IP problems created from (5.1) can be written as an optimization problem of the form:

$$\begin{aligned}
 \min \quad & 0^T x \\
 \text{s.t.} \quad & c^T x \leq c^T x^* + \delta \\
 & Ax \leq b \\
 & l \leq x \leq u \\
 & x \in \mathbb{Z}^n.
 \end{aligned} \tag{5.2}$$

with some scalar δ . The problem 5.2 becomes a feasible IP problem when the $\delta > 0$, and it corresponds to an infeasible IP problem when $\delta < 0$.

The problem statistics are given in Table 5.1. The problem in this table are all pure IP minimization problems. The first three problem are binary integer programming problems. The fourth and the sixth problem are pure general integer programming problems. The fifth problem has 64 binary variables and 17 general integer variables. The last column of the table gives the values we used to create the two versions of the feasibility problems, where a slash separates the value for the infeasible problem and that for the feasible problem. These values corresponds to $c^T x^* + \delta$ in the format 5.2.

Table 5.1 IP Test Problems

Problem	Rows	Variables	Min LP	Max LP	Min IP	$c^T x^* + \delta$
stein15	36	15	7.00	15.00	9	8/9
bm23	20	27	20.57	97.03	34	21/40
p0033	16	33	2520.57	5299.70	3089	2996/3090
afix	44	51	1295.00	470672.00	2136	2135/2140
dp8rint	128	81	8929.00	43239.00	12360	12350/34023
small	24	96	189.65	217.63	191	190/195

5.4.2 Numerical Results

We tested on the three choices of branching direction among the n candidates in the (dual basis of) reduced lattice basis. Table 5.2 shows the time with different choices, and the number of nodes using different choices is shown in Table 5.3. The first six instances are IP infeasible and the last six instances are IP feasible. We use double horizontal line to divide the two groups: the first is infeasible and the second is feasible. Another Table 5.4 gives 40 nodes on two search trees. We compare the time-consumption of the ellipsoidal approximation and the lattice basis reduction on each node. Based on those data, we give our comments and discussion. We use “+” to denote the situation where by the given time or the number of nodes the problem is unsolved and we stopped the procedure.

In the last column of the tables, we also include computational results, i.e., CPU time and number of nodes visited, obtained by running the CPLEX MIP solver version 6.5.2 on the test problems with the default setting. CPLEX is a recognized leading commercial software for solving MIP problems. While as most MIP solvers it uses the branch-and-bound method as the basic tool, the CPLEX MIP solver utilizes a number of cutting plane techniques.

We stress that our inclusion of CPLEX results is for the purpose of providing a reference point. There is little comparability between the performance of our still rather rudimentary implementation of Lenstra’s algorithm and that of the CPLEX MIP solver which, as a leading commercial code, has been under continued research and development for many years. The purpose of our experiments is to test the concepts and viability of Lenstra’s algorithm, not to compare its practical performance with a well-established and comprehensive commercial solver.

We also stress that (i) the computational results for CPLEX were obtained under the CPLEX version 6.5.2 default setting which is not necessarily the optimal for

any given individual problem; and (ii) the newer version of CPLEX performs better on some of the test problems. In particular, we mention that CPLEX version 7.0 can solve the problem *smallm* in a very short amount of time under a well-chosen parameter setting [5].

Table 5.2 Computational Time

Problem	Babai	Hyperplane	Ellipsoid	CPLEX v.6.5.2
stein15m	11	31	37	0.05
bm23m	25	52	125	0.13
p0033m	384.41	483.35	676.09	0.06
afixm	51213	41927	50000+	500.81
dp8rintm	50000+	50000+	50000+	189793+
smallm	2246.77	10488.54	10473.53	286867+
stein15m2	13	14.84	29.41	0.01
bm23m2	49.13	84.52	351.99	0.74
p0033m2	112	340	140	0.02
afixmb2	7741.57	50000+	50000+	131.09
dp8rintm2	40196	44473	41438	0.09
smallm2	60000+	59205	60000+	0.12

5.4.3 Discussion

Based on our computational experience with Lenstra's algorithm, we have the following observations for this algorithm.

i. Lenstra's algorithm takes fewer nodes. Our numerical results indicate that the search tree for Lenstra's algorithm may contain much fewer nodes than a branch-and-bound search tree does. One good example that shows the benefits of hyperplane branching directions is the problem *smallm2.lp*. For this problem, one version of Lenstra's algorithm took only one node to detect the infeasibility while CPLEX v 6.5.2 took tens of thousand nodes to draw the same conclusion. This

Table 5.3 Node Counts

Problem	Babai	Hyperplane	Ellipsoid	CPLEX 6.5.2
stein15m	56	138	181	112
bm23m	4	23	55	58
p0033m	96	234	165	6 (12 cuts)
afixmb	665	336	700+	979.236
dp8rintm	500+	500+	500+	306.148.430+
smallm	1	3	3	734.100.288+
stein15m2	84	97	193	0
bm23m2	35	137	398	49 (22 cuts)
p0033m2	33	82	32	8 (3 cuts)
afixmb2	56	100+	100+	248.296
dp8rintm2	73	67	68	43 (59 cuts)
smallm2	178+	174	200+	132

example suggests that Lenstra's algorithm can be extremely effective in some cases where the branch-and-bound approach encounters great difficulties.

ii. Lattice basis reduction is the bottle neck. In terms of timing, the current implementation of Lenstra's algorithm is still in general non-competitive with good commercial codes like CPLEX. As is mentioned earlier, the limitation lies in the lattice basis reduction part which is too slow when the number of inequality constraints exceeds 120. On our test problem set, the ratio between the time spent on ellipsoidal approximation and that on lattice basis reduction is approximately 1 to 10 on average. In the extreme instance of *smallm2.lp*, where $m = 121$ and $n = 96$, the ratio between the ellipsoidal approximation time and basis reduction time is 1 to 147, and the lattice basis reduction time took more than 99% of the total computation time. This clearly indicates that lattice basis reduction subroutine dominates the computational cost in the current implementation of Lenstra's algorithm. Therefore, a speedup on the lattice basis reduction procedure is crucial to the applicability of Lenstra's algorithm in practice.

iii. Babai's is the best among the three branching policies. Branching policies have a significant impact on the number of nodes of the search trees. In the 11 test problems listed in table 5.3 (problem *dp8rintm2* does not count because it is incomplete), Babai's choice led to fewer nodes in 9 problems. Our choice of the least number of hyperplanes intersecting the circumscribing ball performs better than the other two in the problems *afixmb* and *dp8rintm2*. Even though for this set of problems, the closest direction to the smallest eigenvector of the maximum-volume ellipsoid has provided no advantage, we include it in our study for there may exist problem instances where that branching direction may be favored.

iv. Lenstra's algorithm is promising in combination with other techniques. In advanced codes for integer programming, the traditional branch-and-bound method is often combined with other techniques. To be more effective, Lenstra's algorithm also need to be combined with other techniques. Those techniques include, for example, the warm-start techniques, cutting-plane methods and preprocessing techniques.

In algorithms for integer programming, there is a tradeoff between the number of nodes searched and the cost of searching a node. As mentioned earlier, in the current implementation of Lenstra's algorithm, lattice basis reduction generally takes 90% of the total computational time. Once a breakthrough is made in reducing the cost of lattice basis reduction, Lenstra's algorithm should be very promising for solving certain classes of general IP problems.

Table 5.4 A Sample on Time of the Ellipsoidal Approximation and the Lattice Basis Reduction

Problem Index	Constraints	Variables	Ellipsoidal Appr. Time	Basis Reduction Time
1	282	77	56.27	2084.40
2	280	76	54.76	1890.06
3	278	75	53.61	1710.66
4	276	74	52.49	1623.96
5	274	73	47.29	1507.25
6	272	72	46.43	1377.78
7	270	71	48.97	1313.96
8	268	70	43.82	1165.54
9	266	69	46.35	1110.90
10	264	68	49.97	1031.70
11	262	67	45.54	943.33
12	260	66	43.77	884.03
13	258	65	42.46	742.97
14	256	64	41.86	660.12
15	254	63	43.95	622.03
16	252	62	69.78	569.17
17	250	61	71.25	522.26
18	248	60	72.50	470.91
19	250	61	71.12	535.74
20	248	60	67.31	451.50
21	147	51	10.90	198.66
22	147	50	10.02	192.50
23	147	49	9.79	177.02
24	147	50	11.37	186.18
25	147	49	9.80	174.89
26	147	49	11.16	167.58
27	147	48	9.60	148.03
28	147	50	11.42	182.11
29	147	49	11.16	196.64
30	147	49	9.78	169.53
31	147	48	9.65	164.16
32	147	49	11.16	165.99
33	147	48	9.64	154.77
34	147	48	10.99	143.78
35	147	47	11.48	156.83
36	147	47	9.52	138.16
37	147	46	9.30	128.55
38	147	50	11.37	181.25
39	147	49	11.13	196.29
40	147	49	9.81	172.30

Chapter 6

Conclusions

I have conducted two studies in my thesis research: the development of a practically efficient, primal-dual algorithm for ellipsoidal approximation of polytopes, and the implementation of Lenstra's algorithm along with computational experiments. Main contributions made in this thesis are listed as follows.

A. Ellipsoidal approximation to polytopes.

1. We have developed two primal-dual interior point algorithms for the MaxVE problem. To our knowledge, they are the first algorithms using primal-dual methodology for the MaxVE problem. These two algorithms outperform both the original Khachiyan and Todd algorithm and a modification of it on 200 test problems. One of the two primal-dual interior point algorithms, i.e. the F2PD algorithm, is more robust and efficient than the other.
2. We have proved the well-definedness of the two primal-dual interior-point algorithms for the MaxVE problem, the uniqueness of solution for our formulations, and the existence of paths leading to the solution of the MaxVE problem.

B. Computational study of Lenstra's algorithm.

1. We have implemented a Lovász version of Lenstra's algorithm using our primal-dual algorithm for the MaxVE problem as a subroutine. We have resolved a number of difficult issues in our implementation.
2. We have studied the computational properties of Lenstra's algorithm. We have identified the lattice basis reduction procedure as the main limiting factor af-

fecting the practical performance of Lenstra's algorithm. Our data indicate that Lenstra's algorithm does hold promises for solving certain general integer programming infeasibility problems.

At present, our implementation of Lenstra's algorithm should not be considered as a competitor to the more mature methods such as branch-and-bound and branch-and-cut for solving commonplace integer programming problems. However, we believe that Lenstra's algorithm can be a useful alternative, especially when combined with other available techniques, for solving some small but hard general integer programming problems for which the traditional branch-and-bound algorithms have difficulties.

We suggest the following three points for further study:

1. *Warm-start.* One advantage of the LP-based branch-and-bound method is that one can solve the LP problem corresponding to the child node by making use of the information of the solution of its parent nodes. The same idea, called the warm-start, can be in principle applied to the ellipsoidal approximation and lattice basis reduction procedures.
2. *Combination with other techniques.* Lenstra's algorithm should be combined with other integer programming techniques to become computationally powerful. For example, a stronger preprocessing may have the potential to reduce the computational costs on each node in both ellipsoidal approximation and lattice basis reduction procedures.
3. *Feasibility and Optimization.* Lenstra's algorithm aims at solving the integer programming feasibility problem. The feasibility problem is equivalent to the optimization problem in theory; however, the optimization problem is usually the more common model in practice. How to use the algorithm for feasibility

problem to solve optimization problems efficiently (other than a direct binary search) is another interesting topic.

Bibliography

- [1] K. Aardal, C. Hurkens, and A. K. Lenstra. Solving a linear diophantine equation with lower and upper bounds on the variables. In R.E. Bixby, E.A. Boyd, and R.Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference. Lecture Notes in Computer Science 1412*, pages 229–242. Berlin, Heidelberg, 1998. Springer-Verlag.
- [2] M. Ajtai. The shortest vector problem in l_2 is np-hard for randomized reductions. *Proceedings of 30th ACM Symposium on the Theory of Computing*, pages 10–19, 1998.
- [3] K. M. Anstreicher. *Improved Complexity for Maximum Volume Inscribed Ellipsoids*, June 2001.
- [4] L. Babai. On Lovász’ lattice reduction and nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.
- [5] R. E. Bixby. Personal communication. Rice University, 2001.
- [6] *CBLAS*. <http://www.netlib.org/clapack/cblas>. lapack 2.0 edition.
- [7] J. W. S. Cassels. *An Introduction to the Geometry of Numbers*. Springer-Verlag, Berlin, 1959.
- [8] S.A. Cook. The complexity of theorem-proving procedures. *Proceedings of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

- [9] W. Cook, T. Rutherford, H. E. Scarf, and D. Schallcross. An implementation of the generalized basis reduction algorithm for integer programming. *Operations Research Society of America Journal on Computing*, 5:206–212, 1993.
- [10] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.
- [11] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for estimating volumes of convex bodies. *Journal of the Association for Computing Machinery*, 38:1–17, 1991.
- [12] J. Edmonds, L. Lovász, and W. R. Pulleyblank. Brick decompositions and the matching rank of graphs. *Combinatorica*, 2:247–274, 1982.
- [13] D. Gale, H.W. Kuhn, and A.W. Tucker. *Activity Analysis of Production and Allocation* (T.C. Koopmans, ed.), chapter Linear Programming and the Theory of Games, pages 317–329. John Wiley and Sons, New York, 1951.
- [14] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin, 1988.
- [15] D. S. Hirschberg and C. K Wong. A polynomial-time algorithm for the knapsack problem with two variables. *Journal of the Association for Computing Machinery*, 23:147–154, 1976.
- [16] R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.
- [17] F. John. Extreme problems with inequalities as subsidiary conditions. *Studies and Essays, presented to R. Courant on his 60th Birthday*, pages 187–204, 1948.

- [18] Linderoth J.T. and M.W. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.
- [19] R. Kannan. A polynomial algorithm for the two-variable integer programming problems. *Journal of the Association of Computing Machinery*, 27:118–122, 1980.
- [20] R. Kannan and L. Lovász. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. Technical Report No. 1092, Dept. of Computer Science, Yale University, New Haven, CT, 1996.
- [21] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.
- [22] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.
- [23] L. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.
- [24] L. Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21:307–320, 1996.
- [25] L. Khachiyan and M. Todd. On the complexity of approximating the maximal inscribed ellipsoid for a polytope. *Mathematical Programming*, 61:137–159, 1993.
- [26] A.H. Land and A.G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28:497–520, 1960.
- [27] A. K. Lenstra, H. W. Lenstra, Jr, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

- [28] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.
- [29] LiDIA. *LiDIA – A Library for Computational Number Theory*. TH Darmstadt Universität des Saarlandes, Fachbereich Informatik, Institute für Theoretische Informatik, <http://www.informatik.th-darmstadt.de/pub/TI/LiDIA>, 1.3.4 edition, 1999.
- [30] L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*. The Society for Industrial and Applied Mathematics, Philadelphia, 1986.
- [31] L. Lovász and H.E. Scarf. The generalized basis reduction algorithm. *Mathematics of Operations Research*, 17:751–764, 1992.
- [32] L. Lovász and M. Simonovits. On the randomized complexity of volumes and diameters. In *Proceedings of the 33rd Annual Symposium on Foundation of Computer Science*, pages 482–491, 1992.
- [33] A. Nemirovski. On self-concordant convex-concave functions. Technical report, Technion – Israel Institute of Technology, Technion, Israel, 1997.
- [34] Y. Nesterov and A. Nemirovskii. *Interior-point Polynomial Algorithms in Convex Programming*. The Society for Industrial and Applied Mathematics, Philadelphia, 1994.
- [35] Y. Nesterov and M. Todd. Primal-dual interior-point methods for selfscaled cones. *SIAM Journal on Optimization*, 8:324–364, 1998.
- [36] CPLEX Optimization. *Using the CPLEX Callable Library*. Incline Village, NV, <http://www.cplex.com>, 6.0.1 edition, 1998.

- [37] *MIPLIB*. <http://www.caam.rice.edu/~bixby/miplib/miplib.html>. 3.0 edition. 1996.
- [38] J. Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40:59–93, 1988.
- [39] H.E. Scarf. Production sets with indivisibilities– part I: Generalities. *Econometrica*, 49:1–32, 1981.
- [40] H.E. Scarf. Production sets with indivisibilities– part II: the case of two activities. *Econometrica*, 49:395–423, 1981.
- [41] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–199, 1994.
- [42] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons Ltd, Chichester, 1986.
- [43] V. Shoup. *NTL: A Library for Doing Number Theory*. Department of Computer Science, University of Wisconsin-Madison, <http://www.shoup.net>.
- [44] S. Silvey and D. Titterington. A geometric approach to optimal design theory. *Biometrika*, 60:21–32, 1973.
- [45] S. Tarasov, L. Khachiyan, and I. Erlich. The method of inscribed ellipsoid. *Soviet Mathematics Doklady*, 37:226–230, 1988.
- [46] D. Titteringto. Optimal design: Some geometric aspects of d-optimality. *Biometrika*, 62:313–320, 1975.

- [47] P. van Emde Boas. Another np-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report No. 81-04, Mathematical Institute, University of Amsterdam, Amsterdam, 1981.
- [48] L. Vandenberghe, S. Boyd, and S. Wu. Determinant maximization with matrix inequality constraints. Technical report, EE Dept. Stanford University, 1996.
- [49] X. Wang. *A New Implementation of the Generalized Basis Reduction Algorithm for Convex Integer Programming*. PhD thesis, Yale University, 1997.
- [50] E. Welzl. Smallest enclosing disks, balls and ellipsoids. In H. Maurer, editor, *New Results and New Trends in Computer Sciences*, volume 555, pages 359–370. Springer Lecture Notes in Computer Science, New York, 1991.
- [51] L.A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc. 1998.
- [52] Y. Zhang. An interior-point algorithm for the maximum-volume ellipsoid problem. Technical report, CAAM Dept. Rice University, 1998.