(4)

Systems
Optimization
Laboratory

Semi-Infinite Programming

by
Hui Hu

TECHNICAL REPORT SOL 89-2

March 1989

Department of Operations Research
Stanford University
Stanford, CA 94305

4

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305-4022

Semi-Infinite Programming

by
Hui Hu

TECHNICAL REPORT SOL 89-2

March 1989

Research and reproduction of this report were partially supported by the National Science Foundation grants DMS-8800589 and ECS8617905; U.S. Department of Energy grant DE-FG03-87ER25028 and Office of Naval Research grant N00014-89-J-1659. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do NOT necessarily reflect the views of the above sponsors.

SEMI-INFINITE PROGRAMMING

Hui Hu, Ph.D.
Stanford University, 1989

Abstract

Semi-Infinite programming, that allows for either infinitely many constraints or infinitely many variables but not both, is a natural extension of ordinary mathematical programming. There are many practical as well as theoretical problems in which the constraints depend on time or space and thus can be formulated as semi-infinite programs. The focus of this dissertation is on formulating and solving semi-infinite programming problems. The main results include:

(1) An algorithm for solving a matrix rescaling problem formulated as a semi-infinite linear program; Sufficient conditions that guarantee finite termination are discussed and computational results are reported.

(2) An algorithm for solving a matrix estimation problem equivalent to a semi-infinite quadratic program; For a specified constant, this algorithm will find an approximate solution after finitely many iterations, or will tend to an optimal solution in the limit. An upper bound on the total number of iterations needed for finding an approximate solution is given. Computational results are reported.

(3) A one-phase algorithm for solving a large class of semi-infinite linear programming problems; This algorithm has several advantages: it handles feasibility and optimality together and can detect infeasibility after a finite number of iterations; it has very weak restrictions on the constraints; it allows cuts that are not near the most violated cut; and it solves the primal and the dual problems simultaneously. Upper bounds for finding an ϵ-optimal solution and for the distance between an ϵ-optimal solution and an optimal solution are given.

(4) Applications of the above algorithm to convex programming. First, a certain semi-infinite linear program is solved by this algorithm so as to obtain a feasible solution of a convex program. Then, another semi-infinite linear program is solved by this algorithm so as to obtain an optimal solution of the convex program. In particular, it is shown that for a strongly consistent convex program this algorithm can find a feasible solution after a finite number of iterations.

Acknowledgments

First and foremost, I would like to express my sincere appreciation to my advisor, Professor G. B. Dantzig, for his constant encouragement, guidance, and support. I feel privileged and honored to have him as my dissertation advisor. Without him, this dissertation would not have been written.

I am very grateful to Professor A. J. Hoffman for his encouragement and many stimulating discussions and letters that greatly improved some of my results. He is like a second advisor to me.

I am very grateful to Professor R. W. Cottle and Professor B. C. Eaves for their careful reading of the drafts of this dissertation, their valuable comments, and their friendship.

I wish to thank all my friends at Stanford for their friendship. They made my stay at Stanford very enjoyable. In particular, I wish to thank G. L. Stein for her constant assistance, J. C. Stone for helping me implement my algorithm, and I. Lustig for helping me learn to use TEX, which certainly improved the appearance of this dissertation.

I wish to dedicate this dissertation to Professor G. B. Dantzig and Professor A. J. Hoffman.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Overview

Semi-Infinite programming, that allows for either infinitely many constraints or infinitely many variables but not both, is a natural extension of ordinary mathematical programming. There are many practical problems as well as theoretical problems in which the constraints depend on time or space and thus can be formulated as semi-infinite programs. For example:

(1) A large class of engineering design problems such as electronic circuit design, seismic-resistant structure design, and single-input single-output control systems design can be formulated as nonlinear semi-infinite programs (see Polak (1981)).

(2) The problem of determining a cumulative distribution function that corresponds to a stochastic variable can be formulated as a nonlinear semi-infinite program (Gustafson and Kortanek (1973)).

(3) The educational testing problem, which concerns a reliability coefficient (that measures how reliable the students' total scores are in an examination consisting of a number of subtests), can be formulated as a semi-infinite linear program (Fletcher (1981)).

(4) The standard convex program can be formulated as a semi-infinite linear program (Dantzig (1963), Chapter 24).

The focus of this dissertation is on formulating and solving semi-infinite programming problems.

In Chapter 2, we study a matrix rescaling problem that is interesting from a purely mathematical point of view as well as having important applications. We first formulate this

problem as a semi-infinite linear program and then develop an algorithm that completely solves the problem by taking advantages of its special structure. Sufficient conditions that guarantee finite termination are discussed, and computational results are reported.

In Chapter 3, we solve a matrix estimation problem that differs from the ordinary least-square estimation in that the estimated matrix is required to be positive definite. We transform this problem into an equivalent semi-infinite quadratic program and develop an algorithm to solve it. A constant that measures the amount of positive definiteness required of the estimated matrix must be specified. Given this constant, the algorithm will find an approximate optimal solution after finitely many iterations, or will tend to an optimal solution in the limit. We also give an upper bound on the total number of iterations needed for finding an approximate optimal solution and estimate how good an approximate optimal solution is compared to an optimal solution.

In Chapter 4, we present a one-phase algorithm for solving a large class of semi-infinite linear programming problems. This algorithm is based on the algorithms developed in Chapter 2 and Chapter 3. This algorithm has several advantages: it handles feasibility and optimality together and can detect infeasibility after a finite number of iterations; it has very weak restrictions on the constraints; it allows cuts that are not near the most violated cut; and it solves the primal problem and the dual problem simultaneously. We prove the convergence of this algorithm in two steps. First, we show that this algorithm can find an ϵ-optimal solution after finitely many iterations. Then we use this result to show that it can find an optimal solution in the limit. We also estimate how good an ϵ-optimal solution is compared to an optimal solution and give an upper bound on the total number of iterations needed for finding an ϵ-optimal solution under some assumptions.

In Chapter 5, we show how to solve a convex program by semi-infinite linear programming. We apply the general algorithm presented in Chapter 4 to solve a certain semi-infinite program so as to obtain a feasible solution of the convex program, and then use this solution as a starting point of another semi-infinite program to obtain an optimal solution of the convex program. In particular, for a strongly consistent convex program the algorithm can

find a feasible solution after finitely many iterations.

1.2 Preliminaries

The primal problem of semi-infinite programming is defined as:

(SIP) minimize $f(x)$

subject to

$$g(x,u) \geq 0 \text{ for all } u \in U,$$

where $x \in R^n$, $f : R^n \to R^1$, $U \subseteq R^m$ is an infinite parameter set, and $g : R^n \times U \to R^1$.

An important class of semi-infinite programming problems are semi-infinite linear programs in which the objective function is linear and the constraints are linear in x for any fixed $u \in U$. Explicitly, the primal problem of semi-infinite linear programming is:

(SIL) minimize $c^T x$

subject to

$$a(u)^T x - b(u) \geq 0 \text{ for all } u \in U,$$

where c, $x \in R^n$, U is an infinite parameter set, $a : U \to R^n$, and $b : U \to R^1$. The dual program of (SIL) is defined as an infinite collection of finite dual linear programs: *for all* finite subsets $\{u^i : i \in \Delta\}$ of U

(SID) maximize $\sum_{i \in \Delta} b(u^i) y_i$

subject to

$$\sum_{i \in \Delta} a(u^i) y_i = c$$
$$\Delta \text{ is finite and } \{u^i : i \in \Delta\} \subseteq U$$
$$y_i \geq 0 \text{ for all } i \in \Delta \ .$$

Semi-infinite programming has been studied since the early 1960s. So far there are three types of methods for solving semi-infinite programs: discretization-type methods, semi-continuous-type methods, and continuous-type methods (see, e.g., Hettich (1979)).

A discretization-type method generates $\bar{U}$, a *finite* subset of the infinite parameter set U, and finds an $\bar{x}$ that minimizes $f(x)$ over finitely many constraints $g(x,u) \geq 0$ for

all $u \in \bar{U}$. If the grid size $\bar{d} = \max_{u \in U} \min_{\bar{u} \in \bar{U}} \|u - \bar{u}\|$ is sufficiently small, then $\bar{x}$ is an approximate optimal solution to (SIP). The discretization-type methods are limited to those U having nice properties, e.g., $U = \{u \in R^1 : 0 \leq u \leq 1\}$, and generally fail when U is unbounded. The disadvantage of this type of method is that sometimes the approximate solution $\bar{x}$ may not be satisfactory.

A semi-continuous-type method generates a sequence of finite subsets of U, $\{U^k : k = 1, 2, \dots\}$ with $U^k \subseteq U^{k+1}$ for all k, and finds $x^k = \text{argmin}\{f(x) : g(x, u) \geq 0 \text{ for all } u \in U^k\}$, where $\{U^k : k = 1, 2, \dots\}$ are generated in such a way that any cluster point of the sequence $\{x^k : k = 1, 2, \dots\}$ is an optimal solution of (SIP). Semi-continuous-type methods differ in the ways they generate $\{U^k : k = 1, 2, \dots\}$, some being more efficient than others for the problem at hand.

A continuous-type method typically finds a finite set of differentiable convex functions $h_i(x)$, $i = 1, \dots, t$ such that $\{x : h_i(x) \leq 0, i = 1, \dots, t\}$ is the same as $\{x : g(x, u) \geq 0 \text{ for all } u \in U\}$. When this can be done, the semi-infinite program can be reduced to a standard convex program: $\min_x \{f(x) : h_i(x) \leq 0,\ i = 1, \dots,\ t\}$. The drawback of continuous-type methods is that the assumptions are often restrictive and convergence is often only local.

Chapter 2

Rescaling a Matrix Positive Definite

2.1 Introduction

We solve the following matrix rescaling problem:

Given a square real matrix M, does there exist a positive diagonal matrix D such that DM is positive definite? If such a D exists, how can it be constructed?

Such questions arise in mathematical economics and in the study of certain engineering systems (see, e.g., Arrow and McManus (1958), and Araki (1975)). A necessary and sufficient condition for the existence of such a D for a 3×3 matrix was given by Cross (1978). A necessary and sufficient condition for the existence of D for the general $n \times n$ case was given by Barker, Berman and Plemmons (1978). However, their condition is difficult to verify in practice. Methods for constructing such D's for special classes of matrices were discussed in Barker, Berman and Plemmons (1978), and Berman and Hershkowitz (1983). The existence of D for a square Leontief (Minkowski) matrix was proved by Tartar (1971) and Dantzig (1983). The first algorithm for solving the general problem was given by Khalil (1982). He defines a function $g(x)$ to be the smallest eigenvalue of $D(x)M + M^T D(x)$, where $D(x)$ is a diagonal matrix with diagonal elements $x_1, \ldots, x_n$. His idea is that $D(x)M$ is positive definite if and only if $g(x)$ is positive. His algorithm finds a positive x such that $g(x) > 0$ as follows: At iteration k, if $g(x^k) > 0$, then stop because $D(x^k)M$ is positive definite; otherwise, find a direction d^k by solving a linear program and then construct x^{k+1}

as a convex combination of x^k and d^k such that x^{k+1} is in the interior of the unit box of R^n. If such D's exist, then his algorithm will find one after a finite number of steps. If his algorithm goes on infinitely, then there is no positive diagonal matrix D such that DM is positive definite.

Herein we present an alternative way of solving this problem. Our approach is: first formulate the matrix rescaling problem as a semi-infinite linear program and then solve the semi-infinite linear program by generating and solving a sequence of standard linear programs. Our algorithm is "almost" finite. To see this, let us divide square matrices M into three classes.

Class 1. M that can be rescaled positive definite (i.e., DM is positive definite for some positive diagonal matrix D).

Class 2. M that cannot be rescaled positive definite and moreover cannot be rescaled positive semidefinite (i.e., DM is not positive semidefinite for any nonnega- tive nonzero diagonal matrix D).

Class 3. M that cannot be rescaled positive definite but can be rescaled positive semidefinite (i.e., DM is positive semidefinite for some nonnegative nonzero diagonal matrix D).

Given any Class 1 matrix M, our algorithm can find a positive diagonal matrix D such that DM is positive definite after a finite number of steps. Given any Class 2 matrix M, our algorithm can detect that M cannot be rescaled positive definite after a finite number of steps. Given any Class 3 matrix M, our algorithm can detect that M cannot be rescaled positive definite in the limit. But, a matrix M is of class 3 if and only if the origin is on the boundary of a convex set (see Lemmas 2.2 and 2.3), or equivalently, if and only if a particular function of the elements of M is zero (see Remark 2.3), which can only happen for a set of very special M of measure zero. Thus our algorithm is "almost" finite. Another advantage of our algorithm is that it is not compromised by small errors in the calculation of eigenvalues and eigenvectors.

The content of this chapter is as follows: in Section 2.2 we explain notation and prelim-

inaries. In Section 2.3 we formulate the problem as a semi-infinite linear program, specify the algorithm, prove its correctness and convergence, give conditions that guarantee finite termination, and finally present necessary and sufficient conditions for M to be rescaled positive definite. In Section 2.4 we report on computational results. Finally, we discuss possible ways to accelerate the convergence in Section 2.5.

2.2 Notation and Preliminaries

We define an $n \times n$ real matrix M, not necessarily symmetric, to be *positive definite* if $x^T M x > 0$ for all $0 \neq x \in R^n$, and to be *positive semidefinite* if $x^T M x \geq 0$ for all $x \in R^n$.

If there exists a positive diagonal matrix D such that DM is positive definite, we say that M can be rescaled positive definite. Such matrices are called "diagonally stable" in Barker, Berman and Plemmons (1978), "Lyapunov diagonally stable" in Hershkowitz and Schneider (1985), and "Volterra-Lyapunov stable" in Cross (1978).

Superscripts on vectors are used to denote different vectors, while subscripts are used to denote different components of a vector.

Let $S^{n-1} = \{x \in R^n : x^T x = 1\}$ denote the unit sphere in R^n and $S_+^{n-1} = \{x \in S^{n-1} : x \geq 0\}$ denote the set of nonnegative vectors in S^{n-1}.

$D(x)$ is a diagonal matrix with diagonal elements x_i for $i = 1, \ldots, n$.

For a real symmetric matrix B, let $\lambda[B]$ stand for the smallest eigenvalue of B and $V[B]$ a corresponding eigenvector of unit length.

Given a mathematical programming problem (P), v(P) denotes the optimal objective function value of (P).

Let e^i be the i-th unit vector of R^n and $e = e^1 + \cdots + e^n$.

$\|x\|$ denotes the Euclidean norm of x.

Fact 2.1. *M is positive definite if and only if $M + M^T$ is positive definite.*

Fact 2.2. *M is positive definite if and only if $x^T M x > 0$ for all $x \in S^{n-1}$.*

Fact 2.3. *All principal minors of M remain sign invariant under a positive rescaling DM.*

Fact 2.4. *If M is positive definite, then all its principal minors are positive* (see, e.g., Cottle (1964)).

Fact 2.5. *If M is positive definite, then B^TMB is positive definite for any real nonsingular matrix B.*

Fact 2.6. *For any real symmetric matrix B, $\lambda[B] = \min\{u^TBu : u \in S^{n-1}\}$* (see, e.g., Wilkinson (1965), pp.98-99).

Fact 2.7. *For any real symmetric matrix B, $\lambda[B]$ is a continuous function of the elements of B* (see, e.g., Isaacson and Keller (1966), p.136).

Remark 2.1.

(1) Let DM (MD) be a positive rescaling of the rows (columns) of the matrix M. Then, M can be column-rescaled positive definite if and only if M can be row-rescaled positive definite. Indeed, if DM is positive definite, where D is a positive diagonal matrix, then $(D^{-1})^TDMD^{-1} = MD^{-1}$ is also positive definite (Fact 2.5) and vice versa.

(2) We are only interested in rescaling *nonsymmetric* matrices because if a real symmetric matrix is not positive definite, then it cannot be rescaled positive definite. Indeed, if M can be rescaled positive definite, then (by Facts 2.3 and 2.4) all its principal minors are positive. If a real symmetric matrix is not positive definite, then at least one of its leading principal minors is not positive and thus it cannot be rescaled positive definite.

(3) If M can be rescaled positive definite, then it is easy to see that M is nonsingular and the diagonal elements of M are positive (by Fact 2.4, $d_im_{ii} > 0$ for all i, which implies $m_{ii} > 0$ for all i). Therefore, without loss of generality we assume that the matrix M to be rescaled is nonsymmetric, nonsingular and has only positive diagonal elements.

2.3 An Algorithm and its Convergence

First we show that solving the matrix rescaling problem is equivalent to finding a solution of an infinite system of linear inequalities.

Theorem 2.1. *Suppose that all diagonal elements of the matrix M are positive. Then M*

can be rescaled positive definite if and only if the infinite system of linear inequalities

$$\text{(ISLI)}: \quad [D(u)Mu]^T x \geq 1 \text{ for all } u \in S^{n-1}$$

has a solution. Moreover, if $\bar{x}$ is any solution of (ISLI), *then $D(\bar{x})$ rescales M positive definite.*

Proof. We will make frequent use of the identity $D(u)x \equiv D(x)u$ for all u and x. If there exists a positive diagonal matrix $D(d)$ such that $D(d)M$ is positive definite, let $f(u) = u^T D(d)Mu$ for $u \in S^{n-1}$. Since $f(u)$ is a continuous function of u and S^{n-1} is a compact set, $f(u)$ achieves its infimum on S^{n-1}, i.e., there exists $\bar{u} \in S^{n-1}$ such that $f(u) \geq f(\bar{u}) > 0$ for all $u \in S^{n-1}$. Let $\bar{x} = d/f(\bar{u})$, then

$$\begin{aligned}
[D(u)Mu]^T \bar{x} &= [D(u)Mu]^T d/f(\bar{u}) \\
&= u^T M^T D(u)d/f(\bar{u}) \\
&= u^T M^T D(d)u/f(\bar{u}) \\
&= [u^T D(d)Mu]^T/f(\bar{u}) \\
&= f(u)/f(\bar{u}) \\
&\geq 1
\end{aligned}$$

for all $u \in S^{n-1}$. Thus (ISLI) has a solution. On the other hand, if $\bar{x}$ is a solution of (ISLI), then because $u^T D(\bar{x})Mu$ is a scalar, we have

$$\begin{aligned}
u^T D(\bar{x})Mu &= [u^T D(\bar{x})Mu]^T \\
&= u^T M^T D(\bar{x})u \\
&= u^T M^T D(u)\bar{x} \\
&= [D(u)Mu]^T \bar{x} \\
&\geq 1
\end{aligned}$$

for all $u \in S^{n-1}$. By Fact 2.2, $D(\bar{x})M$ is positive definite. To complete the proof, we only need to show that any feasible x, in particular $\bar{x}$, is positive. Let e^i be the unit vector with 1 in component i. Choose $u = e^i \in S^{n-1}$, then $\bar{x}_i = (e^i)^T D(\bar{x})Me^i/m_{ii} \geq 1/m_{ii} > 0$. ∎

Theorem 2.1 tells us that M can be rescaled positive definite if and only if (ISLI) has a solution; and, moreover, for any $\bar{x}$ solving (ISLI), $D(\bar{x})$ rescales M positive definite. The algorithm we are going to present is designed to test whether (ISLI) has a solution or not and to find such a solution if it exists.

It is well known that deciding whether a finite system of linear inequalities has a solution is equivalent to solving a linear program (see, e.g., Dantzig (1963), Chapter 5). In an analogous way, in order to solve the infinite system of linear inequalities (ISLI), we convert it into a semi-infinite linear program (SILP) by assigning to it an objective function to be minimized such as below:

(SILP) minimize $e^T x$

subject to

$$[D(u)Mu]^T x \geq 1 \text{ for all } u \in S^{n-1},$$

where $e^T = (1, 1, \ldots, 1)$ is a row vector with n ones. The dual of a semi-infinite linear program (DILP) was defined earlier in Section 1.2 as an infinite collection of finite dual linear programs: For all finite subsets $\{u^i : i \in \Delta\}$ of S^{n-1},

(DILP) maximize $\sum_{i \in \Delta} y_i$

subject to

$$\sum_{i \in \Delta} (D(u^i)Mu^i) y_i = e$$
$$y_i \geq 0 \text{ for all } i \in \Delta$$
$$\Delta \text{ is finite and } \{u^i : i \in \Delta\} \subseteq S^{n-1}.$$

The algorithm generates and solves a sequence of linear programs LP(k), the restriction of (DILP) to $n+k$ columns. We will show that if the value of the objective, $v(\text{LP}(k))$, tends to infinity, then M cannot be rescaled positive definite (Theorem 2.3). Otherwise, we will show that a positive vector $\bar{x}$ which rescales M positive definite can be found after finitely many iterations (Theorem 2.2).

As noted earlier, we assume that the input matrix M is nonsingular and has only positive diagonal elements (Remark 2.1).

Algorithm 2.1.

Step 1 (Initialization).

Let $k := 0$;

let $\epsilon < 1$ be a small positive number (e.g., $\epsilon = 10^{-6}$);

let LP(0) be the linear program

maximize $\sum_{i=1}^{n} y_i$

subject to

$$\sum_{i=1}^{n} (D(e^i)Me^i)y_i = e$$

$$y_i \geq 0 \text{ for all } i = 1, \ldots, n.$$

Step 2.

Let x^k be an optimal dual solution of LP(k).

Find a λ^k satisfying $|\lambda^k - \lambda[D(x^k)M + M^T D(x^k)]| < (1/2)\epsilon$.

If $\lambda^k > (1/2)\epsilon$, then $D(x^k)M$ is positive definite, stop.

Step 3.

Find a vector u^{k+1} such that

$\|u^{k+1} - V[D(x^k)M + M^T D(x^k)]\| < \epsilon$ and $(u^{k+1})^T D(x^k)Mu^{k+1} < \epsilon$.

Form LP($k+1$) by adjoining the column, $D(u^{k+1})Mu^{k+1}$,

to the constraint matrix of LP(k) with "cost" coefficient = 1;

solve LP($k+1$).

If v(LP($k+1$)) = ∞, then M cannot be rescaled positive definite, stop.

Else, $k := k + 1$, go to Step 2.

Comments on Algorithm 2.1.

(1) Since we assume that $m_{ii} > 0$ for all $i = 1, \ldots, n$, LP(0) is feasible. Therefore, LP(k) is feasible for all k.

(2) Efficient algorithms for calculating approximate eigenvalues and eigenvectors of a matrix are discussed in Wilkinson (1965). The amount of computational effort to compute these approximations depends on the error bound ϵ.

(3) A symmetric matrix B is positive definite if and only if $\lambda[B] > 0$; in general this is not true for nonsymmetric matrices. Therefore, we calculate the smallest eigenvalue of the symmetric matrix $D(x^k)M + M^T D(x^k)$ in order to know whether $D(x^k)M$ is positive definite or not (Fact 2.1).

(4) If the algorithm does not stop at a certain iteration k, then it generates u^{k+1} satisfying $[D(u^{k+1})Mu^{k+1}]^T x^k < \epsilon < 1$. While $[D(u^i)Mu^i]^T x^k \geq 1$ for all $i = 1, \ldots, k$ since x^k is an optimal dual solution of LP(k). Therefore, no column $D(u^k)Mu^k$ can be brought in more than once.

Next, we prove the correctness and convergence of the algorithm and discuss conditions that guarantee termination in a finite number of iterations.

Theorem 2.2. *If there exists a positive diagonal matrix D such that DM is positive definite, then the algorithm can find such a D after finitely many iterations.*

Proof. If M can be rescaled positive definite, then the corresponding program (SILP) is feasible by Theorem 2.1. Let $\bar{x}$ be a feasible solution of (SILP). Then, for all feasible solutions of (DILP), we have

$$\sum_{i \in \Delta} y_i \leq \sum_{i \in \Delta} y_i [D(u^i)Mu^i]^T \bar{x} = e^T \bar{x}.$$

Namely, the objective function of (DILP) is bounded from above by $e^T \bar{x}$. The algorithm generates a sequence of linear programs LP(k). Each LP(k) is feasible and is the restriction of (DILP) to certain columns. Hence, the objective function of LP(k) is also bounded from above by $e^T \bar{x}$. Recall that the first n columns of the constraint matrix of LP(k) are $m_{ii}e^i$ for $i = 1, \ldots, n$ and therefore any dual solution x satisfies $m_{ii}x_i \geq 1$, $m_{ii} > 0$. By the duality theorem of linear programming, we can find an optimal dual solution x^k of LP(k) which satisfies

$$0 < (m_{ii})^{-1} \leq x_i^k < e^T x^k \leq e^T \bar{x} \text{ for all } i = 1, \ldots, n.$$

Let $T = \{x \in R^n : 0 \leq x_i \leq e^T \bar{x} \text{ for all } i = 1, \ldots, n\}$ and $F(x, u) = [D(u)Mu]^T x$. Then $F(x, u)$ is uniformly continuous on $T \times S^{n-1}$, i.e., for any $\delta > 0$, there exists $\eta > 0$ such

that

$$\|(x,u)-(\bar{x},\bar{u})\|<\eta \text{ implies } |F(x,u)-F(\bar{x},\bar{u})|<\delta \quad \text{for all } (x,u) \text{ and } (\bar{x},\bar{u})\in T\times S^{n-1}. \tag{2.1}$$

In particular, for $\bar{\delta}=1-\epsilon>0$, there exists $\bar{\eta}>0$ such that (2.1) holds. If, on the contrary, we were to assume that the algorithm goes on infinitely, then it generates $u^k\in S^{n-1}$ for $k=0,1,\ldots$. Because S^{n-1} is compact, for the $\bar{\eta}>0$, there exists u^i and u^j in the sequence satisfying $\|u^i-u^j\|<\bar{\eta}$. Without loss of generality, we assume that $i<j$. Because the algorithm does not stop at iteration $j-1$ and $i<j$, we have

$$F(x^{j-1},u^i)=[D(u^i)Mu^i]^Tx^{j-1}\geq 1, \tag{2.2}$$

and

$$F(x^{j-1},u^j)=[D(u^j)Mu^j]^Tx^{j-1}<\epsilon. \tag{2.3}$$

However, (2.2) and (2.3) imply that

$$|F(x^{j-1},u^i)-F(x^{j-1},u^j)|>1-\epsilon=\bar{\delta} \text{ while } \|(x^{j-1},u^i)-(x^{j-1},u^j)\|<\bar{\eta},$$

which contradicts the uniform continuity of $F(x,u)$ on $T\times S^{n-1}$. It follows that the algorithm must be finite for matrices which can be rescaled positive definite. ∎

Remark 2.2. We have proved the finiteness of the algorithm under the assumption that M can be rescaled positive definite. In fact, the boundedness of $v(\mathrm{LP}(k))=e^Tx^k$ for $k=1,2,\ldots$ is *the only assumption* we need for the proof. Since the feasible region of $\mathrm{LP}(k)$ is contained in the feasible region of $\mathrm{LP}(k+1)$, $v(\mathrm{LP}(k+1))\geq v(\mathrm{LP}(k))$ for all k. Therefore, in the case M cannot be rescaled positive definite, the algorithm generates a sequence of feasible solutions of (DILP) whose objective function values tend *increasingly* to infinity. Hence:

Theorem 2.3. *The following are equivalent:*

(1) *M cannot be rescaled positive definite;*

(2) $v(\text{DILP}) = \infty$;

(3) $lim_{k\to\infty} v(\text{LP}(k)) = \infty$. ∎

We have seen that the algorithm is finite in the case M can be rescaled positive definite. We now give a condition which ensures the finiteness of the algorithm in the case M cannot be rescaled positive definite, i.e., there exists a finite j such that $v(\text{LP(j)}) = \infty$.

Condition 2.1. M cannot be rescaled positive definite if for every nonnegative and nonzero diagonal matrix D, there exists a $u \in S^{n-1}$ such that $u^T DMu < 0$. Equivalently, if there does not exist a nonnegative and nonzero diagonal matrix D such that DM is positive semidefinite or positive definite.

Theorem 2.4. *If M satisfies Condition 2.1, then the algorithm terminates after finitely many iterations.*

Proof. Define $G(x) = \lambda[(D(x)M + M^T D(x))/2] = \lambda[D(x)M + M^T D(x)]/2$. It follows from Fact 2.7 that $G(x)$ is a continuous function of x. Since M satisfies Condition 2.1, we have (Fact 2.6)

$$G(x) = \underset{u\in S^{n-1}}{\text{minimum}}\, u^T D(x)Mu < 0 \text{ for all } x \in S_+^{n-1},$$

where $S_+^{n-1} = \{x \in S^{n-1} : x \geq 0\}$. Therefore,

$$\beta = \underset{x\in S_+^{n-1}}{\text{maximum}}\, G(x) = \underset{x\in S_+^{n-1}}{\text{maximum}}\ \underset{u\in S^{n-1}}{\text{minimum}}\, u^T Mu < 0.$$

Let $F(x,u) = [D(u)Mu]^T x = u^T M^T D(x)u = [D(x)Mu]^T u$. Let $\delta = -\beta/2 > 0$, then there exists $\eta > 0$ such that

$$\begin{aligned} &\|(x,u) - (\bar{x},\bar{u})\| < \eta \text{ implies } |F(x,u) - F(\bar{x},\bar{u})| < \delta = -\beta/2 \\ &\qquad \text{for all } (x,u) \text{ and } (\bar{x},\bar{u}) \in S_+^{n-1} \times S^{n-1}. \end{aligned} \tag{2.4}$$

As indicated in the proof of Theorem 2.2, if we assume on the contrary that the algorithm goes on infinitely, it generates u^i and u^j $(i < j)$ satisfying $\|u^i - u^j\| < \eta$ and $F(x^{j-1}, u^i) \geq 1$. Let $\bar{x}^{j-1} = x^{j-1}/\|x^{j-1}\|$. Since $F(x,u)$ is homogeneous of degree one in x,

$$F(\bar{x}^{j-1}, u^i) \geq 1/\|x^{j-1}\| > 0. \tag{2.5}$$

Let $w = V[D(x^{j-1})M + M^T D(x^{j-1})]$. Since w is normalized, $w = V[(D(\bar{x}^{j-1})M + M^T D(\bar{x}^{j-1}))/2]$. Now choose the ϵ of Step 1 of the algorithm such that $\epsilon < \eta$. Then, $\|u^j - w\| < \epsilon < \eta$ in Step 3 and hence we have

$$|F(\bar{x}^{j-1}, u^j) - F(\bar{x}^{j-1}, w)| < \delta = -\beta/2$$

by the uniform continuity of $F(x, u)$ on $S_+^{n-1} \times S^{n-1}$. Thus,

$$\begin{aligned} F(\bar{x}^{j-1}, u^j) &< -\beta/2 + w^T D(\bar{x}^{j-1}) M w \\ &= -\beta/2 + (1/2) w^T [D(\bar{x}^{j-1})M + M^T D(\bar{x}^{j-1})] w \\ &= -\beta/2 + \lambda[(D(\bar{x}^{j-1})M + M^T D(\bar{x}^{j-1}))/2] \\ &= -\beta/2 + G(\bar{x}^{j-1}) \\ &\leq -\beta/2 + \beta \\ &= \beta/2. \end{aligned} \tag{2.6}$$

However, (2.5) and (2.6) imply that $|F(\bar{x}^{j-1}, u^i) - F(\bar{x}^{j-1}, u^j)| > -(1/2)^\rho = \delta$ while $\|(\bar{x}^{j-1}, u^i) - (\bar{x}^{j-1}, u^j)\| = \|u^i - u^j\| < \eta$, which contradicts the uniform continuity of $F(x, u)$ on $S_+^{n-1} \times S^{n-1}$. Therefore, if M satisfies Condition 2.1, then the algorithm terminates after finitely many iterations. ∎

In the rest of this section, we discuss other necessary and sufficient conditions for M to be rescaled positive definite and equivalent statements of Condition 2.1. First, we state a theorem that we are going to use.

Alternative Theorem. *Let $h_i(t)$ for $i = 1, \ldots, n$ be n real-valued functions of $t \in T$ where T may be finite or infinite. If $U = \{(h_1(t), \ldots, h_n(t)) : t \in T\}$ is closed, then*

(1) The system $(h_1(t), \ldots, h_n(t))^T x > 0$ for all $t \in T$ has a solution if and only if the origin is not contained in conv(U), the convex hull of U.

(2) The system $(h_1(t), \ldots, h_n(t))^T x \geq 0$ for all $t \in T$ has a non-trivial (i.e., nonzero) solution if and only if the origin is not contained in the interior of conv(U) (Dines and McCoy (1933)). ∎

Lemma 2.1. *The matrix M with a positive diagonal cannot be rescaled positive definite if and only if $\{x : [D(u)Mu]^T x > 0$ for all $u \in S^{n-1}\}$ is empty.*

Proof. Let $P = \{x : [D(u)Mu]^T x > 0$ for all $u \in S^{n-1}\}$. If P is empty, then (ISLI) has no solution. By Theorem 2.1, M can not be rescaled positive definite. On the other hand, if P is not empty, let $\bar{x} \in P$. Then, $u^T D(\bar{x})Mu = [D(u)Mu]^T \bar{x} > 0$ for all $u \in S^{n-1}$. Hence, $D(\bar{x})M$ *is positive definite and* $\bar{x}$ *is a positive vector.* ∎

Lemma 2.2. *M cannot be rescaled positive definite if and only if the origin is contained in conv$(D(u)Mu : u \in S^{n-1})$.*

Proof. It is easy to show that $\{D(u)Mu : u \in S^{n-1}\}$ is closed. The lemma then follows easily from the Alternative Theorem and Lemma 2.1. ∎

Condition 2.2. $0 \in \text{int}\{\text{conv}[D(u)Mu : u \in S^{n-1}]\}$ where int$\{S\}$ denotes the interior of a set S.

Lemma 2.3. *Condition 2.1 and Condition 2.2 are equivalent.*

Proof. If M satisfies Condition 2.2, then the system

$$u^T D(x)Mu = [D(u)Mu]^T x \geq 0 \quad \textit{for all } u \in S^{n-1}$$

has no non-trivial solutions (by the Alternative Theorem). This implies for any $0 \neq d \geq 0$, there exists $u \in S^{n-1}$ such that $u^T D(d)Mu < 0$, i.e., M satisfies Condition 2.1. On the other hand, if M does not satisfy Condition 2.2, then (by the Alternative Theorem) there exists $\bar{x} \neq 0$ such that $u^T D(\bar{x})Mu = [D(u)Mu]^T \bar{x} \geq 0$ for all $u \in S^{n-1}$. In particular, $(e^i)^T D(\bar{x})Me^i = \bar{x}_i m_{ii} \geq 0$ for all $i = 1, \ldots, n$. This implies that $\bar{x}_i \geq 0$ for all $i = 1, \ldots, n$. (we are assuming that all diagonal elements of M are positive). Hence, M does not satisfy Condition 2.1. ∎

Remark 2.3. Notice that $G(x) =$ minimum $_{x \in S^{n-1}} u^T D(x)Mu$ is a continuous function of x. It is not hard to show the following:

(1) M can be rescaled positive definite if and only if

$$\underset{x \in S_+^{n-1}}{\text{maximum}} \ \underset{u \in S^{n-1}}{\text{minimum}} u^T M u > 0.$$

(2) Condition 2.1 or 2.2 is equivalent to

Condition 2.3. $\underset{x \in S_+^{n-1}}{\text{maximum}} \ \underset{u \in S^{n-1}}{\text{minimum}} u^T M u < 0.$

We now summarize all necessary and sufficient conditions for M to be rescaled positive definite in Theorem 2.5.

Theorem 2.5. *If all diagonal elements of M are positive, then the following are equivalent:*

(1) *M can be rescaled positive definite;*

(2) (ISLI): *$[D(u)Mu]^T x \geq 1$ for all $u \in S^{n-1}$ has a solution;*

(3) *$[D(u)Mu]^T x > 0$ for all $u \in S^{n-1}$ has a solution;*

(4) *the origin is not contained in $conv(D(u)Mu : u \in S^{n-1})$;*

(5) *maximum $_{x \in S_+^{n-1}}$ minimum $_{u \in S^{n-1}} u^T M u > 0$.* ∎

We have proved that the algorithm solves the matrix rescaling problem correctly and completely. It is finite if M is a class 1 or class 2 matrix. It is possibly infinite only if M is a class 3 matrix. Can one give an upper bound on the total number of iterations needed in the case of finite termination? Can one prove that the algorithm is finite for class 3 matrices? The answers will be yes if we have a positive answer to the following open question.

Open Question. In the case (ISLI) has solutions (equivalently, M can be rescaled positive definite), can one give an upper bound on one solution of (ISLI) in terms of M? Namely, can one find a real number $h(M)$ determined by M such that there exists a solution $\bar{x}$ of (ISLI) satisfying $\bar{x}_i \leq h(M)$ for all i?

2.4 Computational Results

We have coded the algorithm in FORTRAN. We use the subroutine MINOS (from the Systems Optimization Laboratory, Department of Operations Research, Stanford University) to solve LP(k) and the subroutine F02ABF (from Department of Computer Science, Stanford University) to calculate eigenvalues and eigenvectors. The data were randomly generated and the program was executed on a DEC 20 computer with the following results, see Table 2.1.

Problem dimension	Number of iterations	CPU time (seconds)
3 x 3	5	3.15
5 x 5	14	7.18
6 x 6	9	4.69
8 x 8	9	6.26
16 x 16	8	6.23

Table 2.1

2.5 Accelerating the Convergence

Algorithm 2.1 solves the semi-infinite linear program (DILP) by generating and solving a sequence of linear programs LP(k), for $k = 0, 1, \ldots$. If the algorithm does not stop at a certain iteration k, then a new column $D(u^{k+1})Mu^{k+1}$ is generated and brought in. This is a cut, $[D(u^{k+1})Mu^{k+1}]^T x \geq 1$, on (SILP). If we want to accelerate the convergence of the algorithm, we have to find ways to generate more efficient cuts.

Let's look at the problem geometrically. Suppose the algorithm does not stop at iteration k. Let $f_k(u) = u^T D(x^k)Mu$. Then, $f_k(u^i) > 0$, $i = 1, \ldots, k$ and $f_k(u^{k+1}) < 0$. Since $f_k(u)$ is a continuous function, for each u^i, $i = 1, \ldots, k$, there exists a relatively open neighborhood $N_k(u^i) \subseteq S^{n-1}$ such that $f_k(u) > 0$ for all $u \in N_k(u^i)$. $f_k(u^{k+1}) < 0$ means that $\bigcup_{i=1}^{k} N_k(u^i)$ does not cover S^{n-1}. Suppose M can be rescaled positive definite and the algorithm stops at iteration k. That means $\bigcup_{i=1}^{k} N_k(u^i)$ covers S^{n-1}. Therefore, we want to make $N_k(u^i)$ bigger so that we need fewer $N_k(u^i)$ to cover S^{n-1}. Let's consider

the "cut" $[D(u^{k+1})Mu^{k+1}]^T x \geq \alpha$ for some $\alpha > 1$, and hope that it will give a bigger $N_{k+1}(u^i)$. However, since we are solving linear programs, changing all the cost coefficients to α will result in a solution αx^{k+1} and therefore has no influence on choosing u^{k+2}. If we go over the proofs of Theorems 2.1 and 2.2, we find that if we change $[D(u^k)Mu^k]^T x \geq 1$ to $[D(u^k)Mu^k]^T x \geq \alpha_k$, where $0 < \delta \leq \alpha_k \leq L$ for all k, then Theorems 2.1 and 2.2 still hold. Since

$$f_k(u^{k+1}) = (1/2)u^{k+1}[D(x^k)M + M^T D(x^k)]u^{k+1}$$

$$= (1/2)\lambda[D(x^k)M + M^T D(x^k)],$$

if $f_k(u^{k+1}) < 0$, a natural way to choose α_{k+1} is

$$\alpha_{k+1} = -\theta f_k(u^{k+1}) = -\theta\lambda[D(x^k)M + M^T D(x^k)],$$

where θ is a positive constant (if $-\theta f_k(u^{k+1}) < \delta$, just let $\alpha_{k+1} = \delta$).

A number of randomly generated problems were computed using the above idea with $\theta = 2$ (revised method) and compared with $\alpha \equiv 1$ (original method), see Table 2.2.

Problem dimension	No. of iterations original method	No. of iterations revised method
5 x 5	14	2
6 x 6	9	8
8 x 8	9	2
16 x 16	8	3

Table 2.2

Chapter 3

Positive Definite Least Square Estimations

3.1 Introduction and Problem Formulation

We solve the following matrix estimation problem:

Given two sequences of vectors a^t and b^t in R^n with $t = 1, \ldots, L$, a small positive number ϵ and a large number $K >> \epsilon$, find a real symmetric matrix $X = (x_{ij})$ such that $\sum_{t=1}^{L} \|X a^t - b^t\|^2$ is minimized among all the real square matrices X satisfying conditions (3.1) and (3.2):

$$X^T = X \text{ and } -K \leq x_{ij} \leq K \text{ for all } i, j = 1, \ldots, n. \tag{3.1}$$

$$\text{The smallest eigenvalue of } X \text{ is no less than } \epsilon. \tag{3.2}$$

This problem arises from mathematical economics (see Dantzig, McAllister and Stone (1988) and Lau (1978)). It differs from the ordinary least square estimation in that the estimated matrix X is required to be positive definite (Fact 2.7). One way of solving this kind of problem was given by Lau (1978). He transformed the problem into an unconstrained nonlinear least square problem by matrix factorization and suggested that the latter can be solved by any algorithm designed for unconstrained nonlinear programs. However, since the objective function for his transformed problem is not convex, global convergence is not guaranteed. Herein we transform the problem into an equivalent semi-infinite convex quadratic program and develop a method to solve it. The solution method essentially solve the problem by ignoring the positive definitness condition, then check to see for the solution $X = X^0$ whether there is a vector $u = u^0$ such that ${u^0}^T X^0 u^0 < \epsilon$. If there is, the linear

inequality in X, namely ${u^0}^T X u^0 \geq \alpha > \epsilon$ become an additional constraint that X must satisfy and the problem is iteratively solved again obtaining $X = X^k$ until on some major iteration k no vector $u = u^k$ can be found such that ${u^k}^T X^k u^k < \epsilon$.

As in Chapter 2, $S^{n-1} = \{x \in R^n : x^T x = 1\}$ denotes the unit sphere in R^n and $\|x\|$ denotes the Euclidean norm of x. For a real symmetric matrix B, $\lambda[B]$ stands for the smallest eigenvalue of B and $V[B]$ a corresponding eigenvector of unit length. Superscripts on vectors are used to denote different vectors, while subscripts are used to denote components of a vector.

To solve this estimation problem, we first transform it into an equivalent (vector form) semi-infinite convex quadratic program.

For the given data a^t, $b^t \in R^n$ for $t = 1, \ldots, L$, let $A = (a^1 \cdots a^L)^T$ and $B^i = (b_i^1 \cdots b_i^L)^T$ for all $i = 1, \ldots, n$. Then A is an $L \times n$ matrix, B^i is an L-vector, and $A^T A$ is an $n \times n$ square matrix. Let M be an $n^2 \times n^2$ block-diagonal matrix with diagonal blocks $A^T A$ and E be an $nL \times n^2$ block-diagonal matrix with diagonal blocks A, namely,

$$M = \begin{pmatrix} A^T A & 0 & \cdots & 0 \\ 0 & A^T A & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A^T A \end{pmatrix}$$

and

$$E = \begin{pmatrix} A & 0 & \cdots & 0 \\ 0 & A & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A \end{pmatrix}.$$

Let $X_{i\cdot}$ be the i-th row of matrix X for all $i = 1, \ldots, n$, and $Y = F(X) = (X_{1\cdot}, \ldots, X_{n\cdot})^T = (x_{11}, \ldots, x_{1n}, \ldots, x_{n1}, \ldots, x_{nn})^T$. For example, if

$$X = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

then $Y = F(X) = (1, 2, 3, 4, 5, 6, 7, 8, 9)^T$. In terms of vectors, condition (3.1) becomes:

$$\begin{gathered} Y_{(i-1)n+j} = x_{ij} = x_{ji} = Y_{(j-1)n+j} \text{ for all } i, j = 1, \ldots, n \\ \text{and } -K \leq Y_k \leq K \text{ for all } i = 1, \ldots, n^2. \end{gathered} \tag{3.1'}$$

By Fact 2.6, condition (3.2) is equivalent to: $u^T X u \geq \epsilon$ for all $u \in S^{n-1}$. Therefore, condition (3.2) becomes:

$$u^T X u = \sum_{i=1}^{n} u_i u^T (X_{i\cdot})^T = (u_1 u^T \dots u_n u^T) Y \geq \epsilon \text{ for all } u \in S^{n-1}. \qquad (3.2')$$

The objective function becomes:

$$\begin{aligned}
\sum_{t=1}^{l.} \|X a^t - b^t\|^2 &= \sum_{t=1}^{l.} \sum_{i=1}^{n} (X_{i\cdot} a^t - b_i^t)^2 \\
&= \sum_{i=1}^{n} \sum_{t=1}^{l.} (X_{i\cdot} a^t - b_i^t)^2 \\
&= \sum_{i=1}^{n} \sum_{t=1}^{l.} ({a^t}^T {X_{i\cdot}}^T - b_i^t)^2 \\
&= \sum_{i=1}^{n} \|A (X_{i\cdot})^T - B^i\|^2 \\
&= \sum_{i=1}^{n} \{X_{i\cdot} A^T A (X_{i\cdot})^T - 2 (B^i)^T A (X_{i\cdot})^T + (B^i)^T B^i\} \\
&= Y^T M Y - 2((B^1)^T \cdots (B^n)^T) E Y + \sum_{i=1}^{n} (B^i)^T B^i.
\end{aligned}$$

Consequently, the equivalent semi-infinite quadratic program is

(SIQP) minimize $Y^T M Y - 2((B^1)^T \cdots (B^n)^T) E Y + \sum_{i=1}^{n} (B^i)^T B^i$

subject to

$$(u_1 u^T \cdots u_n u^T) Y \geq \epsilon \text{ for all } u \in S^{n-1}$$
$$Y_{(i-1)n+j} - Y_{(j-1)n+i} = 0 \text{ for all } i, j = 1, \dots, n$$
$$-K \leq Y_i \leq K \text{ for all } i = 1, \dots, n^2.$$

Remark 3.1.

(1) M is positive semidefinite because each of its diagonal blocks $A^T A$ is positive semidefinite. This implies that the objective function of (SIQP) is convex. The feasible region of (SIQP) is compact and convex, and is defined by an infinite number of linear constraints. Moreover, it is nonempty since $\bar{Y} = F(\epsilon I)$ is a feasible solution, where I is the identity matrix. Therefore, (SIQP) is a feasible semi-infinite convex quadratic program and optimal solutions for (SIQP) exist.

(2) Since $Y = F(X) = (X_1 \cdot \cdots X_n \cdot)$ is a bijection, $F^{-1}(\cdot)$ exists. Therefore, $X = F^{-1}(Y)$ and $u^T X u = u^T F^{-1}(Y) u = (u_1 u^T \cdots u_n u^T) Y$. For convenience, we use $u^T F^{-1}(Y) u$ and $(u_1 u^T \cdots u_n u^T) Y$ interchangeably.

We have shown that this positive definite least square estimation problem is equivalent to the semi-infinite convex quadratic program (SIQP). In Section 3.2, we propose an algorithm for solving (SIQP) and prove its convergence. In Section 3.3, we estimate the total number of major iterations and the final objective function value to attain a prescribed level of approximation to an optimal solution. In Section 3.4, we present computational results for randomly generated data.

3.2 An Algorithm and its Convergence

We propose an algorithm for solving the semi-infinite quadratic program (SIQP). This algorithm solves (SIQP) by generating and solving a sequence of feasible convex quadratic programs QP(k) for $k = 1, 2, \ldots$. Every QP(k) has the same objective function as that of (SIQP) and the feasible region of QP(k) contains that of QP($k+1$).

Algorithm 3.1.

Step 1 (Initialization).

Let $k := 0$;

let α be a constant such that $\epsilon \leq \alpha << K$;

let QP(0) be the quadratic program

$$\text{minimize} \quad H(Y) = Y^T M Y - 2((B^1)^T \cdots (B^n)^T) E Y + \sum_{i=1}^{n} (B^i)^T B^i$$

subject to

$$Y_{(i-1)n+j} - Y_{(j-1)n+i} = 0 \text{ for all } i, j = 1, \ldots, n$$

$$-K \leq Y_i \leq K \text{ for all } i = 1, \ldots, n^2.$$

Step 2.

Find an optimal solution Y^k of QP(k);

let $X^k = F^{-1}(Y^k)$, i.e., $x_{ij}^k = Y_{(i-1)n+j}^k$ for all $i, j = 1, \ldots, n$;

calculate $\lambda[X^k]$ and $V[X^k]$;

if $\lambda[X^k] \geq \epsilon$, go to Step 4.

Step 3.

Let $u^k = V[X^k]$;

form QP($k+1$) by adding a cut, $(u^k)^T F^{-1}(Y)u^k \geq \alpha$, to QP(k);

$k := k + 1$;

go to Step 2.

Step 4.

If $\alpha > \epsilon$, Y^k is an approximate optimal solution of (SIQP); stop.

If $\alpha = \epsilon$, Y^k is an optimal solution of (SIQP); stop.

Comments on Algorithm 3.1.

(1) For any k, the feasible region of QP(k) is a nonempty polytope since $Y = F(\alpha I)$ is a feasible solution. Therefore, optimal solutions exist for all QP(k). Furthermore, since the objective function of QP(k) is *quadratic* and *convex*, there are *finite* algorithms for finding one of its optimal solutions.

(2) Efficient finite algorithms for calculating eigenvalues and eigenvectors of a matrix to any specified level of approximation can be found in Wilkinson (1965).

(3) k counts the number of major iterations (Step 2–Step 3), or equivalently, the number of cuts added before termination. Each major iteration can be processed finitely.

(4) α is a constant and $\epsilon \leq \alpha << K$. If $\alpha > \epsilon$, then an approximate optimal solution will be found after a finite number of major iterations (see Theorem 3.1). If $\alpha = \epsilon$, then any cluster point of of the sequence Y^0, Y^1, Y^2, ... is an optimal solution for (SIQP) (see Theorems 3.2 and 3.3).

Theorem 3.1. *If $\alpha > \epsilon$, then Algorithm 3.1 can find an approximate optimal solution of* (SIQP) *after a finite number of major iterations.*

Proof. Let $H(Y) = Y^T MY - 2((B^1)^T \cdots (B^n)^T)EY + \sum_{i=1}^{n}(B^i)^T(B^i)$ be the objective

functions of (SIQP) and QP(k) for all k. Let

$$C = \{Y : -K \leq Y_i \leq K, i = 1, \ldots, n^2\} \times S^{n-1} \text{ and } G(Y,u) = u^T F^{-1}(Y)u.$$

Then $G(Y,u)$ is uniformly continuous on C. Therefore, for $\delta = \alpha - \epsilon > 0$, there exists $\eta > 0$ such that

$$\|(Y,u) - (\bar{Y},\bar{u})\| < \eta \text{ implies } |G(Y,u) - G(\bar{Y},\bar{u})| < \delta \quad \text{for all } (Y,u) \text{ and } (\bar{Y},\bar{u}) \text{ in } C. \tag{3.3}$$

If Algorithm 3.1 goes on infinitely, then it generates a sequence $u^k \in S^{n-1}$ for $k = 0, 1, 2, \ldots$. Since S^{n-1} is compact, for the $\eta > 0$, there exist u^{k_i} and u^{k_j} in the sequence such that $\|u^{k_i} - u^{k_j}\| < \eta$. Without loss of generality, we assume that $k_i < k_j$. Since Algorithm 3.1 does not stop at iteration k_j and $k_i < k_j$, we have:

$$G(Y^{k_j}, u^{k_i}) = (u^{k_i})^T F^{-1}(Y^{k_j})u^{k_i} \geq \alpha; \tag{3.4}$$

$$G(Y^{k_j}, u^{k_j}) = (u^{k_j})^T F^{-1}(Y^{k_j})u^{k_j} < \epsilon. \tag{3.5}$$

However, (3.4) and (3.5) contradict (3.3). Therefore, if $\alpha > \epsilon$, then Algorithm 3.1 must terminate finitely. Suppose that it stops at a certain iteration k. Then, $\lambda[F^{-1}(Y^k)] \geq \epsilon$ and by Fact 2.6, Y^k is a feasible solution of (SIQP). However, since $\alpha > \epsilon$, the constraints $(u^i)^T F^{-1}(Y)u^i \geq \alpha$ for $i = 0, 1, \ldots, k-1$ may be violated by certain feasible solutions of (SIQP). We can not guarantee that $H(Y^k) \leq H(Y)$ holds for all feasible Y. But, if a feasible solution of (SIQP) satisfying $\lambda[F^{-1}(Y)] \geq \alpha$, then $H(Y^k) \leq H(Y)$ is guaranteed to hold. Therefore, Y^k is only an approximate optimal (or α–suboptimal) solution in the case $\alpha > \epsilon$. ∎

Remark 3.2. When α increases, the number of cuts required to be added before termination decreases, but the final objective function value increases. For given data and choice of α, if the algorithm does not terminate by a specified number of iterations, we can always increase α and try again.

Theorem 3.2. *If $\alpha = \epsilon$ and Algorithm 3.1 stops at a certain iteration k, then Y^k is an optimal solution of* (SIQP).

Proof. Let $H(Y)$ be the objective functions of (SIQP) and QP(i) for all i. If Algorithm 3.1 stops at a certain iteration k, then $\lambda[X^k] \geq \epsilon$. By Fact 2.6, Y^k is a feasible solution of (SIQP). Next, suppose that Y is an *arbitrary* feasible solution of (SIQP). Because $\alpha = \epsilon$, all cuts generated in Step 3 are necessary. Hence, Y is feasible for all QP(i). In particular, Y is feasible for QP(k). Since Y^k is an optimal solution of QP(k), we have $H(Y^k) \leq H(Y)$. It follows that Y^k is an optimal solution of (SIQP). ∎

Theorem 3.3. *Suppose that $\alpha = \epsilon$ and that Algorithm 3.1 does not stop finitely. Let the sequence Y^0, Y^1, Y^2, ... be generated by Algorithm 3.1. Then any cluster point of this sequence is an optimal solution of* (SIQP).

Proof. Let Y^k and u^k for $k = 0, 1, 2, \ldots$ be generated by Algorithm 3.1 with $\alpha = \epsilon$. Since Y^k for $k = 0, 1, \ldots$ are in a compact set, there exist cluster points. Let Y^* be a cluster point, and without loss of generality we assume that $\lim_{k\to\infty} Y^k = Y^*$. We claim that for any β satisfying $0 < \beta < \epsilon$, there exists an integer $N(\beta)$ such that $\lambda[F^{-1}(Y^k)] \geq \beta$ for all $k \geq N(\beta)$. We skip the proof here since it is similar to the proof in Theorem 3.1. By this claim and the continuity of $F^{-1}(\cdot)$ and $\lambda[\cdot]$ (Fact 2.7), we have $\lambda[F^{-1}(Y^*)] \geq \beta$ for any β satisfying $0 < \beta < \epsilon$. Consequently, $\lambda[F^{-1}(Y^*)] \geq \epsilon$ and Y^* is feasible for (SIQP). It remains to show that Y^* is optimai. Let Y be an arbitrary feasible solution of (SIQP). Then Y is feasible for all QP(k). Because Y^k is an optimal solution of QP(k) and the objective functions $H(Y)$ of QP(k) and (SIQP) are the same, we have $H(Y^k) \leq H(Y)$ for $k = 0, 1, 2, \ldots$. It follows that $H(Y^*) \leq H(Y)$ holds for all feasible Y and Y^* solves (SIQP). ∎

Remark 3.3. Since $\alpha = \epsilon$, the feasible region of QP(k) contains that of (SIQP). As the algorithm goes on, Y^k becomes more and more close to the feasible region of (SIQP) and $H(Y^k)$ tends *increasingly* to $H(Y^*)$.

3.3 Estimation of Upper Bounds

We have proved that if $\alpha > \epsilon$, then the algorithm can find an approximate optimal solution of (SIQP) after finitely many iterations. Let Y^* denote an optimal solution of (SIQP) and

Y^α denote the approximate optimal solution for some $\alpha > \epsilon$. We know that Y^α is feasible for (SIQP) and that for any feasible Y satisfying $\lambda[F^{-1}(Y)] \geq \alpha$, $H(Y) \geq H(Y^\alpha) \geq H(Y^*)$ holds. Now we want to know how good Y^α is, namely, we want to have an upper bound for $|H(Y^\alpha) - H(Y^*)|$. Remember that when $\alpha = \epsilon$, the algorithm generates a sequence Y^i for $i = 1, 2, \ldots$, and $H(Y^i)$ tends *increasingly* to $H(Y^*)$. Therefore, if we want to know how good Y^α is, we reset $\alpha = \epsilon$ and execute the algorithm until some iteration k. Then we have $|H(Y^\alpha) - H(Y^*)| \leq H(Y^\alpha) - H(Y^k)$, and $|H(Y^\alpha) - H(Y^*)|/H(Y^*) \leq (H(Y^\alpha) - H(Y^k))/H(Y^k)$.

In the rest of this section, we give an upper bound of the total number of major iterations needed for finding Y^α. We begin with Lemma 3.1, which was stated and proved by Alan J. Hoffman (verbal communication).

Lemma 3.1. *If* $u, \bar{u} \in S^{n-1}$, *then* $\sum_{i,j=1}^{n} |(u_i - \bar{u}_i)u_j| \leq n\|u - \bar{u}\|$.

Proof. Let $v_i = |u_i - \bar{u}_i|$ for $i = 1, \ldots, n$, $x^1 = (v_1, \ldots, v_n)$, $x^2 = (v_2, \ldots, v_n, v_1)$, $\ldots$, $x^n = (v_n, v_1, \ldots, v_{n-1})$, and $y = (|u_1|, \ldots, |u_n|)$. Then,

$$\begin{aligned}
\sum_{i,j=1}^{n} |(u_i - \bar{u}_i)u_j| &= \sum_{i,j=1}^{n} v_i|u_j| \\
&= y^T x^1 + y^T x^2 + \cdots + y^T x^n \\
&\leq \|y\| \sum_{i=1}^{n} \|x^i\| \\
&= n\|y\|\|x^1\| \\
&= n\|u - \bar{u}\|. \quad \blacksquare
\end{aligned}$$

Lemma 3.2. *Suppose that* $\|u - \bar{u}\| \leq (\alpha - \epsilon)/2nK$ *where* $u, \bar{u} \in S^{n-1}$, $\alpha > \epsilon > 0$, *and* K *is the given bound on* Y. *If* Y *is feasible for* (SIQP) *and* $G(Y, \bar{u}) \geq \alpha$, *then* $G(Y, u) \geq \epsilon$.

Proof.

$$\begin{aligned}|G(Y,u) - G(Y,\bar{u})| &= |\sum_{i,j=1}^{n}(u_iu_j - \bar{u}_i\bar{u}_j)Y_{(i-1)n+j}| \\ &\le K\sum_{i,j=1}^{n}|(u_i - \bar{u}_i)u_j| + |\bar{u}_i(u_j - \bar{u}_j)| \\ &\le 2nK\|u - \bar{u}\| \text{ (by Lemma 3.1)} \\ &\le \alpha - \epsilon.\end{aligned}$$

Therefore, $G(Y,u) \ge G(Y,\bar{u}) - (\alpha - \epsilon) \ge \alpha - (\alpha - \epsilon) = \epsilon$. ∎

Let $B_n(u,r)$ denote the n-dimensional Euclidean ball with center $u \in R^n$ and radius r. At iteration k, $G(Y,u^k) \ge \alpha$ is added into the constraints of QP$(k+1)$. If the algorithm does not stop, then it will find a $u^{k+1} \in S^{n-1}$ satisfying $G(Y^{k+1},u^{k+1}) < \epsilon$. Since Y^{k+1} is feasible for QP$(k+1)$, we have $G(Y^{k+1},u^i) \ge \alpha$ for $i = 1, \ldots, k$. From Lemma 3.2, we know that for all $k = 1, 2, \ldots,$

$$u^{k+1} \notin \bigcup_{i=1}^{k} B_n(u^i, (\alpha - \epsilon)/2nK) \cap S^{n-1}. \tag{3.6}$$

Hence, if at a certain iteration k we have

$$\bigcup_{i=1}^{k} B_n(u^i,(\alpha - \epsilon)/2nK) \cap S^{n-1} = S^{n-1}, \tag{3.7}$$

then the algorithm must stop since it cannot find a $u^{k+1} \in S^{n-1}$ satisfying $G(Y^{k+1},u^{k+1}) < \epsilon$. To estimate a k such that (3.7) holds, we need to know a lower bound of the $(n-1)$-content (volumn) of $\cup_{i=1}^{k} B_n(u^i,(\alpha - \epsilon)/2nK) \cap S^{n-1}$. In order to obtain this lower bound, we need a lower bound of the $(n-1)$-content of $B_n(u,r) \cap S^{n-1}$. In the following discussion, we use $V_j(T)$ to denote the j-content of a j-dimensional set T.

Fact 3.1.

(1) $V_n(B_n(u,r)) = r^n\sqrt{\pi^n}/\Gamma((n/2)+1)$,

(2) $V_{n-1}(S^{n-1}) = 2\sqrt{\pi^n}/\Gamma(n/2)$,

where $\Gamma(\beta) = \int_0^{+\infty} x^{\beta-1}e^{-x}dx$ for $\beta > 0$ is the gamma function.

Lemma 3.3. *If* $u \in S^{n-1}$ *and* $r \le \sqrt{2}$, *then,*

$$V_{n-1}(B_n(u,r) \cap S^{n-1}) \ge (\sqrt{\pi/2}\, r)^{n-1}/\Gamma((n+1)/2).$$

Proof. Without loss of generality, let $u = (0, 0, \ldots, 1)$. Then,

$$\begin{aligned} V_{n-1}(B_n(u,r)) &= \int\cdots\int_{x_1^2+\cdots+x_{n-1}^2 \le \rho^2} (1 - x_1^2 - \cdots - x_{n-1}^2)^{-1/2} dx_1 \cdots dx_{n-1} \\ &\ge \int\cdots\int_{x_1^2+\cdots+x_{n-1}^2 \le \rho^2} dx_1 \cdots dx_{n-1}, \end{aligned}$$

where ρ is obtained by solving the following system of equations:

$$\begin{aligned} x_1^2 + \cdots + x_n^2 &= 1 \\ x_1^2 + \cdots + x_{n-1}^2 + (1 - x_n)^2 &= \rho^2 \\ x_1^2 + \cdots + x_{n-1}^2 &= \rho^2. \end{aligned} \tag{3.8}$$

Solving (3.8) we get $\rho^2 = r^2(4 - r^2)/4$. Since $r \le \sqrt{2}$, we have $\rho \ge r/\sqrt{2}$. Therefore,

$$\begin{aligned} V_{n-1}(B_n(u,r) \cap S^{n-1}) &\ge \int\cdots\int_{x_1^2+\cdots+x_{n-1}^2 \le r/\sqrt{2}} dx_1 \cdots dx_{n-1} \\ &= (\sqrt{\pi/2}\, r)^{n-1}/\Gamma((n+1)/2). \quad \blacksquare \end{aligned}$$

Theorem 3.4. *If* $\alpha > \epsilon$, *then an upper bound for the total number of major iterations needed for finding* Y^α *is:* $(\pi/2)\sqrt{n}\,(4\sqrt{2}\,nK/(\alpha - \epsilon))^{n-1}$.

Proof. From (3.6) we know that $B_n(u^i, (\alpha - \epsilon)/4nK) \cap B_n(u^j, (\alpha - \epsilon)/4nK)$ is empty for all $i \ne j$. Therefore,

$$\begin{aligned} & V_{n-1}\Big(\bigcup_{i=1}^{k} B_n(u^i, (\alpha - \epsilon)/2nK) \cap S^{n-1}\Big) \\ \ge\ & V_{n-1}\Big(\bigcup_{i=1}^{k} B_n(u^i, (\alpha - \epsilon)/4nK) \cap S^{n-1}\Big) \\ =\ & kV_{n-1}(B_n(u^1, (\alpha - \epsilon)/4nK) \cap S^{n-1}) \\ \ge\ & k\big(\sqrt{\pi/2}\,(\alpha - \epsilon)/4nK\big)\big)^{n-1}/\Gamma((n+1)/2). \end{aligned} \tag{3.9}$$

It follows from (3.7), (3.9), and Fact 3.1 that an upper bound for the total number of major iterations for finding Y^α is:

$$\frac{2\sqrt{\pi^n}/\Gamma(n/2)}{\left(\sqrt{\pi/2}\,(\alpha-\epsilon)/4nK\right)^{n-1}/\Gamma((n+1)/2)} = \frac{2\sqrt{\pi}(4\sqrt{2}\,nK)^{n-1}\Gamma((n+1)/2)}{(\alpha-\epsilon)^{n-1}\Gamma(n/2)}. \tag{3.10}$$

Since $G(Y,u) = G(-Y,u)$ for all $u \in R^n$, only half of S^{n-1} needs to be covered by $\bigcup_{i=1}^{k} B_n(u^i, (\alpha-\epsilon)/2nK \cap S^{n-1})$. Therefore, we can cut the bound in (3.10) by half. Also, it is not hard to show that $\Gamma((n+1)/2)/\Gamma(n/2) \leq \sqrt{n\pi}/2$ (see Appendix A). Thus Theorem 3.4 is established. ∎

3.4 Computational Results

We have coded Algorithm 3.1 in FORTRAN. We use the subroutine QPSOL (from the Systems Optimization Laboratory, Department of Operations Research, Stanford University) to solve QP(k) and the subroutine F02ABF (from Department of Computer Science, Stanford University) to calculate eigenvalues and eigenvectors. The program was executed on a DEC 20 computer. All components of the input data a^t and b^t are *i.i.d.* $U(-0.5, 0.5)$. First, we compute one problem six times with different α values to demonstrate the influence of α on the number of major iterations and on the final objective values, see Table 3.1.

Given data:

$L = 8, \quad \epsilon = 1.0, \quad K = 10^9, \quad n^2 = 16;$

$a^1 =$ (-0.3052, 0.1087, -0.3915, -0.4383)

$a^2 =$ (0.1379, 0.1707, -0.1208, 0.3839)

$a^3 =$ (0.2999, -0.4803, 0.1790, -0.2021)

$a^4 =$ (-0.1334, 0.1864, -0.0431, 0.4557)

$a^5 =$ (-0.0681, -0.4627, -0.1384, 0.0547)

$a^6 =$ (-0.4691, 0.0743, 0.3823, 0.1650)

$a^7 =$ (-0.2117, -0.3549, 0.4991, -0.1264)

$a^8 =$ (-0.0865, 0.0886, -0.4886, -0.3304)

$b^1 =$ (0.2325, -0.1774, -0.3115, 0.2133)

b^2 = (-0.4512, -0.1078, 0.0383, -0.0906)

b^3 = (-0.0641, -0.3664, -0.1086, -0.3182)

b^4 = (-0.3645, -0.1941, -0.1331, -0.3830)

b^5 = (-0.2327, -0.0301, -0.0613, 0.2470)

b^6 = (-0.3909, 0.3732, -0.0953, -0.1953)

b^7 = (-0.1478, -0.2652, -0.3996, 0.3307)

b^8 = (-0.2671, 0.3283, 0.0569, -0.3668)

Value of alpha	No. of major iterations	Final objective function value	CPU time (seconds)
1.1	4	5.2713	1.89
1.01	7	4.8017	2.77
1.001	13	4.7585	4.94
1.0001	14	4.7537	5.44
1.00001	15	4.7532	5.88
1.000001	16	4.7532	6.63

Table 3.1

Next we solve a number of problems in different dimensions, see Table 3.2. Again, all components of the input data a^t and b^t are *i.i.d.* $U(-0.5, 0.5)$. $\alpha = 1.01$, $\epsilon = 1$, and problem dimension $= [n^2, L]$.

Problem dimension	No. of major iterations	Final objective function value	CPU time (seconds)
[16,6]	7	3.3775	2.9
[16,10]	8	5.9837	3.32
[36,8]	15	9.0326	25.18
[36,12]	17	14.1864	28.19
[64,12]	33	19.1114	219.29
[64,18]	26	29.6169	175.32

Table 3.2

A practical problem based on economic data was also solved. It was quickly solved in two major iterations.

Chapter 4

Semi-Infinite Programming

4.1 Introduction and Preliminaries

The primal problem (SIL) of semi-infinite linear programming is defined as

(SIL) minimize $c^T x$

subject to

$$a(u)^T x - b(u) \geq 0 \text{ for all } u \in U,$$

where c, $x \in R^n$, $U \subseteq R^m$ is an infinite parameter set, $a : U \to R^n$, and $b : U \to R^1$. The dual of (SIL) is: *for all* finite subsets $\{u^i : i \in \Delta\}$ of U

(SID) maximize $\sum_{i \in \Delta} b(u^i) y_i$

subject to

$$\sum_{i \in \Delta} a(u^i) y_i = c$$
$$\Delta \text{ is finite and } \{u^i : i \in \Delta\} \subseteq U$$
$$y_i \geq 0 \text{ for all } i \in \Delta .$$

We can always rewrite (SID) in the format of generalized linear programming (see Dantzig (1963) Chapter 22) as follows:

(GLP) maximize y_0

subject to

$$p_0 y_0 + py = q$$

p may be freely chosen from S

$$y \geq 0,$$

where $S = \text{conv}\{(a(u)^T, -b(u))^T : u \in U\}$, $p_0 = (0, \ldots, 0, 1)$, and $q = (c^T, 0)^T$. Indeed, for any $\{u^i : i \in \Delta\} \subseteq U$ with finite Δ and any $\{y_i \geq 0 : i \in \Delta\}$ satisfying $\sum_{i\in\Delta} a(u^i)y_i = c$, let $y_0 = \sum_{i\in\Delta} b(u^i)y_i$, $y = \sum_{i\in\Delta} y_i > 0$ (since $c \neq 0$) and $p = \sum_{i\in\Delta}(a(u^i)^T, -b(u^i))^T y_i/y$. Then, $p \in S$ and $p_0 y_0 + py = q$. Conversely, if $p_0 y_0 + py = q$, where $y \geq 0$ and $p \in S$, then there exist $u^i \in U$ and $\lambda_i > 0$ for $i = 1, \ldots, t$ ($t \leq n + 1$) such that $\sum_{i=1}^t \lambda_i = 1$ and $p = \sum_{i=1}^t (a(u^i)^T, -b(u^i))^T \lambda_i$ (see, e.g., Rockafellar (1972) p.155). Let $y_i = \lambda_i y \geq 0$ for $i = 1, \ldots, t$. Then, $\sum_{i=1}^t a(u^i)y_i = c$ and $y_0 = \sum_{i=1}^t b(u^i)y_i$. Hence, (SID) and (GLP) are equivalent.

The first algorithm for solving (GLP) and therefore (SID) was given by Dantzig and Wolfe, see Dantzig (1963), Chapters 22 and 24, and Dantzig (1960). Starting with a non-degenerate basic feasible solution of (GLP), the Dantzig-Wolfe algorithm solves (GLP) by generating a p^k with the smallest negative reduced cost at each iteration k and then solves the new restricted master program. Although one can start this algorithm with any basic feasible solution, the nondegeneracy of the starting basic feasible solution is required by the convergence proof. How to find a starting nondegenerate basic feasible solution is still an open question.

Gustafson and Kortanek proposed an algorithm, *the alternating procedure*, for solving the primal semi-infinite linear program (SIL) (see Gustafson and Kortanek (1973)) under the following assumptions:

(A1) U is compact,

(A2) $a(u)$ and $b(u)$ are continuous,

(A3) the feasible region of (SIL) is nonempty,

(A4) the feasible region of (SIL) is contained in T, where $T \subseteq R^n$ is a nonempty

polytope.

The alternating procedure solves (SIL) as follows: let x^0 be an optimal solution of the linear program

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} \\ &&& x \in T. \end{aligned}$$

At iteration k, if $a(u)^T x^k - b(u) \geq 0$ for all $u \in U$, then x^k is an optimal solution of (SIL) and the procedure is stopped. Otherwise, find

$$u^{k+1} \in \operatorname{argmin}\{a(u)^T x - b(u) : u \in U\}$$

and let x^{k+1} be an optimal solution of the linear program

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} \\ &&& a(u^i)^T x - b(u^i) \geq 0 \text{ for all } i = 1, \ldots, k+1 \\ &&& x \in T. \end{aligned}$$

Under the above assumptions, Gustafson and Kortanek (1973) proved that $c^T x^* = v(\text{SIL})$, where x^* is a cluster point of the sequence x^k generated by the alternating procedure and $v(\text{SIL})$ stands for the optimal objective function value of program (SIL). The convergence proof of the alternating procedure provided by them, however, is incomplete: they did not prove the *feasibility* of x^* (although it can be shown that x^* is a feasible solution by the same approach as the proof of Theorem 4.1).

Herein we present a one-phase algorithm for solving a large class of semi-infinite linear programs. This algorithm is based on Algorithm 2.1 and Algorithm 3.1 and can be generalized to solve nonlinear semi-infinite programs. It has several advantages:

1. It handles feasibility and optimality together and thus eliminates the extra work of finding a starting feasible solution that might be far away from an optimal solution.
2. It can detect infeasibility after a finite number of iterations.
3. It does not impose any restriction on U, $a(u)$, and $b(u)$, i.e., U could be any nonempty set in R^m, $a(u)$ could be any vector in R^n, and $b(u)$ could be any real number.

4. The cut it generates at each iteration need not be the most violated cut or near the most violated cut. The strength of the cuts can be controlled by a parameter.
5. It solves the primal and the dual programs simultaneously.

The rest of this chapter is organized as follows. In Section 4.2 we present an algorithm for solving a large class of semi-infinite programs and prove its convergence in two steps. First, we show that it can find an ϵ-optimal solution after finitely many iterations, and then use this result to show that it can find an optimal solution in the limit. In Section 4.3 we prove a duality theorem and show that the algorithm solves the primal and the dual programs simultaneously. In Section 4.4 we estimate how good an ϵ-optimal solution is compared to an optimal solution and give an upper bound on the total number of iterations needed for finding an ϵ-optimal solution under certain assumptions. In Section 4.5 we show that the algorithm can be generalized to solve a class of nonlinear semi-infinite programming problems.

4.2 An Algorithm for Semi-Infinite Linear Programming

We first present a one-phase algorithm for solving the primal problem (SIL) of semi-infinite linear programming that satisfies the assumption that the feasible region is contained in a polytope T.

Algorithm 4.1.

Step 1 (Initialization).

Let $k := 0$;

let α be a constant such that $0 < \alpha < 1$;

let LP(0) be the linear program

$$\text{minimize} \quad c^T x$$

subject to

$$x \in T.$$

Step 2.

If LP(k) is infeasible, then (SIL) is infeasible, stop.

Else, find an optimal solution x^k of LP(k);

if $a(u)^T x^k - b(u) \geq 0$ for all $u \in U$,

then x^k is an optimal solution of (SIL), stop.

Step 3.

Find a $u^k \in U$ such that

either $a(u^k)^T x^k - b(u^k) \leq -\alpha$

or $a(u^k)^T x^k - b(u^k) \leq \alpha \cdot \inf\{a(u)^T x^k - b(u) : u \in U\}$;

form LP($k+1$) by adding a cut, $a(u^k)^T x - b(u^k) \geq 0$, to LP($k$);

$k := k + 1$;

go to Step 2.

Comments on Algorithm 4.1.

(1) We assume that, given x^k, there are algorithms that can find an approximate solution of the nonlinear program $\inf\{a(u)^T x^k - b(u) : u \in U\}$. Namely, given any small positive number δ, these algorithms can find a $\bar{u} \in U$ such that $a(\bar{u})^T x^k - b(\bar{u}) < \inf\{a(u)^T x^k - b(u) : u \in U\} + \delta$.

(2) In Step 2, to test whether $a(u)^T x^k - b(u) \geq 0$ for all $u \in U$, one can proceed as follows: given a small positive number δ, find a $\bar{u} \in U$ such that

$$a(\bar{u})^T x^k - b(\bar{u}) < \inf\{a(u)^T x^k - b(u) : u \in U\} + \delta.$$

If $a(\bar{u})^T x^k - b(\bar{u}) \geq \delta$, then $\inf\{a(u)^T x^k - b(u) : u \in U\} \geq 0$ and x^k is optimal. If $a(\bar{u})^T x^k - b(\bar{u}) \leq 0$, then $\inf\{a(u)^T x^k - b(u) : u \in U\} < 0$ and one can go to Step 3. Otherwise, $0 \leq a(\bar{u})^T x^k - b(\bar{u}) \leq \delta$, which implies that $\inf\{a(u)^T x^k - b(u) : u \in U\} \geq -\delta$, and therefore, x^k is an δ-optimal solution (see Definition 3.1 below). If one wants an exact optimal solution, then one needs an algorithm that can find $\inf\{a(u)^T x^k - b(u) : u \in U\}$ starting from a δ-optimal solution.

(3) In Step 3, either there is a u^k that satisfies $a(u^k)^T x^k - b(u^k) \leq -\alpha$ or else $0 > \inf\{a(u)^T x^k - b(u) : u \in U\} \geq -\alpha > -\infty$. In the latter case,

$$\alpha \cdot \inf\{a(u)^T x^k - b(u) : u \in U\} > \inf\{a(u)^T x^k - b(u) : u \in U\} (\text{ since } \alpha < 1),$$

and hence there exists a $u^k \in U$ such that

$$a(u^k)^T x^k - b(u^k) \leq \alpha \cdot \inf\{a(u)^T x^k - b(u) : u \in U\}. \tag{4.1}$$

To find a u^k satisfying (4.1) by any algorithm that has the property stated in (1), a stopping rule is

$$a(u^k)^T x^k - b(u^k) \leq \inf\{a(u)^T x^k - b(u) : u \in U\} + \delta \quad \text{and}$$

$$\delta \leq (\alpha - 1) a(u^k)^T x^k - b(u^k).$$

Indeed, if this rule is satisfied, then

$$\begin{aligned} \delta &\leq (\alpha - 1)(a(u^k)^T x^k - b(u^k)) \\ &\leq (\alpha - 1) \inf\{a(u)^T x^k - b(u) : u \in U\}, \end{aligned}$$

and therefore,

$$\begin{aligned} a(u^k)^T x^k - b(u^k) &\leq \inf\{a(u)^T x^k - b(u) : u \in U\} + \delta \\ &\leq \alpha \cdot \inf\{a(u)^T x^k - b(u) : u \in U\}. \end{aligned}$$

(4) The most violated cut $a(u^*)^T x^k - b(u^*) \geq 0$, where $a(u^*)^T x^k - b(u^*) = \inf\{a(u)^T x^k - b(u) : u \in U\}$, *may not exist. Even if it does exist,* it could be in general very hard to find. The parameter α specified in Step 1 controls the strength of the cut generated in Step 3. When α is close to one, the cut is near the most violated cut when it exists. When α is sufficiently close to zero, the cut is very weak, nevertheless the algorithm will still converge.

It is easy to see that, if Algorithm 4.1 stops at a certain iteration k, then it either finds an optimal solution of (SIL) or detects the infeasibility of (SIL). We are now going to prove that if the algorithm does not stop finitely, then any cluster point of the sequence x^k for all $k = 1, 2, \ldots$ generated in Step 2 is an optimal solution of (SIL).

Definition 4.1 (ϵ-optimal solution). Let $\epsilon > 0$. A vector $\bar{x}$ is an ϵ-optimal solution of (SIL) if $a(u)^T \bar{x} - b(u) \geq -\epsilon$ for all $u \in U$ and $c^T \bar{x} \leq v(\text{SIL})$, where $v(\text{SIL})$ denotes the optimal objective function value of (SIL).

Theorem 4.1. *Suppose that* $\{\|(a(u), b(u))\|_\infty : u \in U\}$ *is bounded by* $\bar{K} > 0$ *and that the algorithm does not stop finitely. Let the sequence* $x^1, x^2, \ldots$ *be generated by Step 2 of Algorithm 4.1. Then,*

(a) (Finite ϵ-convergence) *For any ϵ satisfying $0 < \epsilon < \alpha$, there exists an integer $N(\epsilon)$ such that x^k is an ϵ-optimal solution of* (SIL) *for all $k \geq N(\epsilon)$.*

(b) (Convergence) *Any cluster point of the sequence $x^1, x^2, \ldots$ is an optimal solution of* (SIL).

Proof. Let $W = \{s \in R^{n+1} : \|s\|_\infty \leq \bar{K}\}$ and $G(x, y, z) = y^T x - z$, where $x, y \in R^n$ and $z \in R^1$. Then, $G(x, y, z)$ is uniformly continuous on $T \times W$. If (a) does not hold, then for some $0 < \epsilon' < \alpha$, $N(\epsilon')$ does not exist. Therefore, the set $J = \{k : x^k$ is not an ϵ'-optimal solution of (SIL)$\}$ contains infinitely many integers. Let $\eta = \alpha\epsilon' > 0$. By the uniform continuity of $G(x, y, z)$, there exists $\delta(\eta) > 0$ such that

$$\begin{aligned} \|(x, y, z) - (\bar{x}, \bar{y}, \bar{z})\| < \delta(\eta) \text{ implies that } |G(x, y, z) - G(\bar{x}, \bar{y}, \bar{z})| < \eta \\ \text{for all } (x, y, z) \text{ and } (\bar{x}, \bar{y}, \bar{z}) \in T \times W. \end{aligned} \tag{4.2}$$

Since $\{(a(u), b(u)) : u \in U\}$ is bounded and J is infinite, we can find $i, j \in J$ with $i < j$ such that

$$\begin{aligned} \|(x^j, a(u^i), b(u^i)) - (x^j, a(u^j), b(u^j))\| &= \|(a(u^i), b(u^i)) - (a(u^j), b(u^j))\| \\ &< \delta(\eta). \end{aligned} \tag{4.3}$$

Because $i < j$, we have

$$G(x^j, a(u^i), b(u^i)) = a(u^i)^T x^j - b(u^i) \geq 0. \tag{4.4}$$

Because $j \in J$ and $c^T x^j \leq v$(SIL), there exists $\bar{u} \in U$ such that $a(\bar{u})^T x^j - b(\bar{u}) < -\epsilon'$ and, thus, $\inf\{a(u)^T x^j - b(u) : u \in U\} < -\epsilon'$. Therefore, by Step 3 of the algorithm we have either

$$G(x^j, a(u^j), b(u^j)) = a(u^j)^T x^j - b(u^j) \leq -\alpha < -\eta \tag{4.5a}$$

or

$$\begin{aligned} G(x^j, a(u^j), b(u^j)) &= a(u^j)^T x^j - b(u^j) \\ &\leq \alpha \cdot \inf\{a(u)^T x^j - b(u) : u \in U\} \\ &< -\alpha\epsilon' = -\eta. \end{aligned} \tag{4.5b}$$

However, (4.3), (4.4), (4.5a), and (4.5b) contradict (4.2). Consequently, (a) is proved.

It remains to prove (b). Since the sequence x^k, $k = 1, 2, \ldots$ is bounded, it has at least one cluster point. Let x^* be a cluster point of this sequence. From (a) we know that for *any* ϵ satisfying $0 < \epsilon < \alpha$ and $k \geq N(\epsilon)$ we have $a(u)^T x^k - b(u) \geq -\epsilon$ for all $u \in U$. Hence, for *any* ϵ satisfying $0 < \epsilon < \alpha$, we have $a(u)^T x^* - b(u) \geq -\epsilon$ for all $u \in U$, which implies that x^* is a feasible solution of (SIL). To complete the proof, we need only to show that $c^T x^* = v(\text{SIL})$. First,we know that $c^T x^* \geq v(\text{SIL})$ since x^* is a feasible solution of (SIL). Secondly, we know that $c^T x^k \leq v(\text{SIL})$ holds for all k since the feasible region of LP(k) contains that of (SIL). It follows that $c^T x^* = v(\text{SIL})$. ∎

Remark 4.1.

(1) We have proved that, if the algorithm goes on infinitely, then any cluster point of the sequence x^1, x^2, ... is a feasible solution of (SIL). Therefore, if (SIL) is infeasible, then the algorithm will detect infeasibility after a *finite* number of major iterations (i.e., Step 2–Step 3).

(2) Without loss of generality, we may always assume that $\{\|(a(u), b(u))\|_\infty : u \in U\}$ is bounded. For we can pre-multiply $(a(u), b(u))$ by a positive number, e.g., $\|(a(u), b(u))\|^{-1}$ and the feasible region remains the same.

(3) We proved the convergence of Algorithm 4.1 *without* assuming the continuity of $a(u)$ and $b(u)$, and *without* assuming the compactness of U. We are able to do so because $G(x, y, z) = y^T x - z$ is uniformly continuous on $T \times W$.

In the rest of this section, we discuss how to solve a semi-infinite linear program that does not have the constraint $x \in T$.

Suppose we know that for $\bar{u}^1, \ldots, \bar{u}^l \in U$ the linear program

LP(0)′ minimize $c^T x$

subject to

$$a(\bar{u}^i)^T x \geq b(\bar{u}^i) \text{ for } i = 1, \ldots, t$$

has an optimal solution. Then we can start the algorithm with LP(0)′ as LP(0). Since the feasible region of LP(k) contains that of LP($k+1$), either LP($k+1$) is infeasible or LP($k+1$) is optimal. Hence, Algorithm 4.1 can be carried out. If the algorithm stops finitely, then it correctly solves the problem. Otherwise, a sequence $x^1, x^2, \ldots$ will be generated. It is easy to see that, if this sequence is bounded, then both (a) and (b) of Theorem 4.1 hold. If this sequence is not bounded but has a cluster point, then (b) of Theorem 4.1 still holds and (a) should be stated as

(a)′ For any $0 < \epsilon < \alpha$, there exists an x^i in this sequence such that x^i is an ϵ-optimal solution of (SIL).

A sufficient condition that guarantees the boundedness of the sequence $x^1, x^2, \ldots$ was given by Dantzig (see Dantzig (1963) pp.476-477) for the Dantzig-Wolfe algorithm. This condition works for our algorithm after some minor modifications.

Lemma 4.1. *Let* DP(k) *be the dual program of* LP(k) *for all* k. *If* (SIL) *is feasible and a nondegenerate basic feasible solution exists for some* DP(k), *then the sequence* x^k *is bounded.*

Proof. As in ordinary linear programming, if (SIL) is feasible, then $v(\text{SID}) < +\infty$. Let B^k be the basis associated with a nondegenerate basic feasible solution of some DP(k) and b^k be the vector of corresponding cost coefficients. Then, by nondegeneracy, we have $(B^k)^{-1}c > 0$. Let $\gamma^i = (x^i)^T B^k - (b^k)^T$ for all $i = 1, 2, \ldots$. Since x^i is feasible for LP(i) and for all $i \geq k$ the feasible region of LP(i) contains

that of LP(k), we know that $\gamma^i \geq 0$ for all $i \geq k$. Hence, for all $i \geq k$, we have

$$\begin{aligned} 0 &\leq \gamma^i (B^k)^{-1} c \\ &= (x^i)^T c - (b^k)^T (B^k)^{-1} c \\ &= v(\text{DP(i)}) - (b^k)^T (B^k)^{-1} c \\ &\leq v(\text{SID}) - (b^k)^T (B^k)^{-1} c. \end{aligned}$$

It follows that γ^i is bounded and thus x^i is bounded. ∎

Other simple sufficient conditions for the boundedness of x^k or for the existence of a cluster point of x^k are:

Lemma 4.2.

(1) *If $a(u^j) < 0$ for some u^j generated by the algorithm and $x^k \geq 0$ for all sufficiently large k, then the sequence x^k is bounded.*

(2) *If $a(u^j) < 0$ for some u^j generated by the algorithm and $x^k \geq 0$ for infinitely many k, then the sequence x^k has a cluster point.*

(3) *The assumption, $a(u^j) < 0$ for some u^j generated by the algorithm, in (1) and (2) may be replaced by : there exist u^{j_1}, ..., u^{j_l} generated by the algorithm such that $a(u^{j_i}) \leq 0$ for $i = 1, \ldots, l$ and $\sum_{i=1}^{l} a(u^{j_i}) < 0$.*

Proof. (1). For all $x^k \geq 0$ and $k \geq j$, we have, by the feasibility of x^k, $a(u^j)^T x^k \geq b(u^j)$. Therefore, $x_i^k \leq b(u^j)/a(u^j)_i$ for all $i = 1, \ldots, n$ and thus (1) is proved. (2) and (3) can be proved similiarly. ∎

4.3 A Duality Thoerem

It is well known that, if a primal linear program has an optimal solution, then its dual program also has an optimal solution and the optimal objective function values of the primal program and the dual program are equal. This nice property does not hold for all semi-infinite linear programs (see, e.g., Duffin and Karlovitz (1965)). Herein we show that

if Algorithm 4.1 converges for a primal semi-infinite linear program, then there is no duality gap between the primal program and its dual program and Algorithm 4.1 solves the primal program and the dual program simultaneously.

Let DP(k) be the dual program of LP(k) for all k, where LP(k) is generated by Algorithm 4.1, and y^k be an optimal solution of DP(k). If at a certain iteration k, LP(k) is infeasible, then by the duality theorem of linear programming, we have $v(\text{DP}(k)) = \infty$. Since the feasible region of DP(k) is a subset of that of (SID), we know that in this case $v(\text{SID}) = \infty$. If at a certain iteration k, x^k is an optimal solution of (SIL), then by the duality theory of linear programming, we have $v(\text{SIL}) = c^T x^k = v(\text{LP}(k)) = v(\text{DP}(k))$. On the other hand, it is easy to show that $v(\text{SID}) \leq v(\text{SIL})$. Since y^k is a feasible solution of (SID) and the objective function value of (SID) at y^k is equal to $v(\text{SIL})$, it follows that y^k is an optimal solution of (SID). If Algorithm 4.1 generates an infinite sequence x^k for all $k = 1, 2, \ldots$ with a cluster point x^*, then we have

$$c^T x^k = v(\text{LP}(k)) = v(\text{DP}(k)) \leq v(\text{SID}) \leq v(\text{SIL}). \tag{4.6}$$

Since the feasible region of LP(k) contains that of LP($k+1$), $v(\text{LP}(k)) = c^T x^k$ is nondecreasing. Hence, we have $\lim_{k\to\infty} c^T x^k = c^T x^*$. By Theorem 4.1, $c^T x^* = v(\text{SIL})$. It follows from (4.6) that $c^T x^* = v(\text{SIL}) = v(\text{SID})$. Now suppose that the method used for solving LP(k) in Step 2 can solve LP(k) and DP(k) simultaneously (e.g., the simplex method) and y^k is an optimal solution of DP(k) generated by this method. Then, y^k for $k = 1, 2, \ldots$ is a sequence of feasible solutions of (SID) on which the objective function tends to $v(\text{SID})$. Thus, the following theorem is established.

Theorem 4.2. *Suppose that Algorithm 4.1 convergences.*

(1) *If* (SIL) *is infeasible, then* $v(\text{SID}) = \infty$.

(2) *If* (SIL) *has an optimal solution, then* $v(\text{SIL}) = v(\text{SID})$.

(3) *Algorithm 4.1 solves* (SIL) *and* (SID) *simultaneously.* ∎

Corollary 4.1. *Suppose that* $\{(a(u), b(u)) : u \in U\}$ *is compact, the feasible region of* (SIL)

has an interior point, v(SID) is finite, and Algorithm 1 converges. Then, v(SID) can be attained.

Proof. Let $\hat{x}$ be an interior point of the feasible region of (SIL), then $a(u)^T\hat{x} > b(u)$ for all $u \in U$. As $\{(a(u), b(u)) : u \in U\}$ is compact, we know that there exists a positive number θ such that $a(u)^T\hat{x} - b(u) \geq \theta > 0$ for all $u \in U$. Let y^k be the optimal solution of DP(k) generated by Algorithm 4.1 and Δ_k be the index set such that $y_i^k > 0$ if and only if $i \in \Delta_k$. If we solve LP(k) and DP(k) by the simplex method, then Δ_k contains no more than n elements. Let $|\Delta_k|$ denote the cardinal number of Δ_k and $Y^k = (y^k_{i_{k_1}}, \ldots, y^k_{i_{k_{|\Delta_k|}}}, 0, \ldots, 0) \in R^n$, that is, the first $|\Delta_k|$ components of Y^k are $y^k_{i_{k_1}}, \ldots, y^k_{i_{k_{|\Delta_k|}}}$ and the rest are zero in the case $|\Delta_k| < n$.

We claim that $\{Y^k \in R^n : k = 1, 2, \ldots\}$ is bounded. Indeed,

$$\begin{aligned}
\theta \sum_{i \in \Delta_k} y_i^k &\leq \sum_{i \in \Delta_k} a(u^i)^T\hat{x} y_i^k - \sum_{i \in \Delta_k} b(u^i) y_i^k \\
&= c^T\hat{x} - v(\mathrm{DP(k)}) \\
&= c^T\hat{x} - c^T x^k \\
&\leq c^T\hat{x} - c^T x^1.
\end{aligned}$$

Let $A^k = (a(u^{i_{k_1}}), \ldots, a(u^{i_{k_{|\Delta_k|}}}), 0, \ldots, 0) \in R^{n \times n}$ and $b^k = (b(u^{i_{k_1}}), \ldots, b(u^{i_{k_{|\Delta_k|}}}), 0, \ldots, 0)^T \in R^n$ for all $k = 1, 2, \ldots$. Then, we have $A^k Y^k = c$ and $(b^k)^T Y^k = v(\mathrm{DP(k)})$ for all $k = 1, 2, \ldots$. Without loss of generality we assume that $\lim_{k\to\infty} A^k = (a(u^{j_1}), \ldots, a(u^{j_r}), 0, \ldots, 0) \equiv A^*$, $\lim_{k\to\infty} b^k = (b(u^{j_1}), \ldots, b(u^{j_r}), 0, \ldots, 0) \equiv b^*$, and $\lim_{k\to\infty} Y^k = (Y_1^*, \ldots, Y_r^*, 0, \ldots, 0) \equiv Y^*$, where $r \leq n$. Then, $A^*Y^* = c$, $(b^*)^T Y^* = v(\mathrm{SID})$, and thus, $y_{j_1} = Y_1^*, \ldots, y_{j_r} = Y_r^*$, $y_i = 0$ for all $i \notin \{j_1, \ldots, j_r\}$ is an optimal solution of (SID). ∎

4.4 Estimation of Upper Bounds

We have shown that for any ϵ satisfying $0 < \epsilon < \alpha$, the algorithm can find an ϵ-optimal solution of (SIL) after finitely many iterations (Theorem 4.1). Now we estimate how good

an ϵ-optimal solution is compared to an optimal solution. In this section, we assume that U is compact, $a(u)$ and $b(u)$ are continuous on U, and the feasible region of (SIL) is n-dimensional. First, we state two lemmas that we are going to use in the following discussion.

Lemma 4.3A (Hoffman). *Let A be an $(m \times n)$-matrix and b an m-vector. If $Ax \geq b$ has a solution, then there exists a constant τ such that for any $x \in R^n$, there exists an x^o satisfying*

$$Ax^o \geq b \text{ and } \|x - x^o\| \leq \tau \|(Ax - b)^-\|_\infty,$$

where $(Ax - b)^-$ stands for the negative part of the vector $(Ax - b)$, see Hoffman (1952). ∎

Lemma 4.3B. *If $\{(a(u), b(u)) : u \in U\}$ is compact and $F = \{x \in R^n : a(u)^T x \geq b(u)\ u \in U\}$ is n-dimensional, then a given halfspace $(a^*)^T x \geq b^*$ (where $a^* \neq 0$) contains F if and only if there exist $u^i \in U$ and $\lambda_i > 0$, $i = 1, \ldots, t$ (where $t \leq n$) such that*

$$\lambda_1 a(u^1) + \cdots + \lambda_t a(u^t) = a^*$$

and

$$\lambda_1 b(u^1) + \cdots + \lambda_t b(u^t) \geq b^*.$$

(see, e.g., Rockafellar (1972), p.160.) ∎

Definition 4.2. Let $g : F \to R^n$. A vector $x \in F$ is a stationary point of the pair (F, g) if $x^T g(x) \leq y^T g(x)$ for all $y \in F$.

Suppose that x^k is an ϵ-optimal solution of (SIL). Consider an auxiliary program

$$\text{(P)} \quad \min\{\|x - x^k\|^2 : a(u)^T x - b(u) \geq 0 \quad \text{for all } u \in U\}.$$

Let x^* be the optimal solution of the auxiliary program (P). It is easy to see that x^* is a stationary point of $(F, \nabla\|x - x^k\|^2)$, where $F = \{x \in R^n : a(u)^T x \geq b(u)\ u \in U\}$ and $\nabla\|x - x^k\|^2 = 2x - 2x^k$ is the gradient of $\|x - x^k\|^2$. Therefore, the halfspace

$$2(x^* - x^k)^T x \geq 2(x^* - x^k)^T x^*$$

contains F. It follows from Lemma 4.3B that there exist $u^i \in U$ and $\lambda_i > 0$, $i = 1, \ldots, t$ (where $t \leq n$) such that

$$\begin{aligned} \lambda_1 a(u^1) + \cdots + \lambda_t a(u^t) &= 2(x^* - x^k) \\ \lambda_1 b(u^1) + \cdots + \lambda_t b(u^t) &\geq 2(x^* - x^k)^T x^*. \end{aligned} \tag{4.7a}$$

We claim that

$$a(u^i)^T x^* = b(u^i) \text{ for } i = 1, \ldots, t. \tag{4.7b}$$

If not, say, $a(u^i)^T x^* > b(u^i)$ for $i = 1, \ldots, r$, and $a(u^i)^T x^* = b(u^i)$ for $i = r+1, \ldots, t$. Then, we have

$$\begin{aligned} 2(x^* - x^k)^T x^* &= \sum_{i=1}^{t} \lambda_i a(u^i)^T x^* \\ &> \sum_{i=1}^{t} \lambda_i b(u^i) \\ &\geq 2(x^* - x^k)^T x^*, \end{aligned}$$

which is a contradiction. Thus (4.7b) is proven. Since $\|x - x^k\|^2$ is a convex function of x, (4.7a) and (4.7b) imply that x^* is an optimal solution of the following program:

$$\mathrm{P}(k) \quad \min\{\|x - x^k\|^2 : A_k x \geq b_k\},$$

where $A_k = (a(u^1), \ldots, a(u^t))^T$ and $b_k = (b(u^1), \ldots, b(u^t))^T$. Applying Hoffman's Lemma to the finite system of linear inequalities $A_k x \geq b_k$, we know that there exists a constant τ and an x^o satisfying

$$A_k x^o \geq b_k \text{ and } \|x^k - x^o\| \leq \tau \|(A_k x^k - b_k)^-\|_\infty \leq \tau\epsilon.$$

Because x^* is an optimal solution of $\mathrm{P}(k)$ and x^o is a feasible solution of $\mathrm{P}(k)$, we have

$$\|x^k - x^*\| \leq \|x^k - x^o\| \leq \tau\epsilon$$

and therefore,

$$|c^T x^k - c^T x^*| \leq \tau\epsilon\|c\| \text{ and } -\tau\epsilon\|c\| + c^T x^* \leq c^T x^k \leq v(SIL) \leq c^T x^*.$$

Thus, the following theorem is proved.

Theorem 4.3. *The distance between an ϵ-optimal solution and the feasible region is no greater than $\tau\epsilon$, and the difference between the objective function value of an ϵ-optimal solution and the optimal objective function value is no greater than $\tau\epsilon\|c\|$.* ∎

There is a simple way to estimate ϵ for the k-th approximate solution x^k such that x^k is an ϵ-optimal solution. Suppose that

$$a(u^k)^T x^k - b(u^k) \leq \alpha \cdot \inf\{a(u)^T x^k - b(u) : u \in U\}$$

of Step 3 holds for x^k and u^k. Let

$$w^k = \min_{i<k}\{|(a(u^i)^T x^k - b(u^i)) - (a(u^k)^T x^k - b(u^k))|\}.$$

Since $a(u^i)^T x^k - b(u^i) \geq 0$ for all $i < k$, we have $a(u^k)^T x^k - b(u^k) \geq -w^k$. This implies that $\inf\{a(u)^T x^k - b(u) : u \in U\} \geq -w^k\alpha^{-1}$, i.e., x^k is a $(w^k\alpha^{-1})$-optimal solution.

Finally, we give an upper bound of the total number of iterations needed for finding an ϵ-optimal solution. We need the following assumptions:

(A5) $a(u)$ and $b(u)$ are Lipschitz continuous on U with Lipschitz constants L_2 and L_3 respectively, i.e., $\|a(u) - a(\bar{u})\| \leq L_2\|u - \bar{u}\|$ and $|b(u) - b(\bar{u})| \leq L_3\|u - \bar{u}\|$ for all $u, \bar{u} \in U$.

(A6) $\|x^k\| \leq K$ for all $k = 1, 2, \ldots$, where x^k is generated by Step 2 of Algorithm 1.

(A7) Every cut, $a(u^k)^T x - b(u^k) \geq 0$, generated by Step 3 of Algorithm 1 is the most violated cut.

Lemma 4.4. *If assumptions (A5) and (A6) are satisfied and $a(\bar{u})^T x^k - b(\bar{u}) \geq 0$, then $a(u)^T x^k - b(u) \geq -\epsilon$ for all u that satisfies $\|u - \bar{u}\| \leq \epsilon(L_2K + L_3)^{-1}$.*

Proof.

$$\begin{aligned} |a(u)^T x^k - b(u) - (a(\bar{u})^T x^k - b(\bar{u}))| &\leq \|a(u) - a(\bar{u})\|\|x^k\| + |b(u) - b(\bar{u})| \\ &\leq (L_2K + L_3)\|u - \bar{u}\| \\ &\leq \epsilon. \quad ∎ \end{aligned}$$

Let $B_m(u, r)$ denote the m-dimensional Euclidean ball with center $u \in R^m$ and radius r. If x^k is not an ϵ-optimal solution, then $a(u^k)^T x^k - b(u^k) < -\epsilon$ (by (A7)). Since $a(u^i)^T x^k - b(u^i) \geq 0$ for all $i < k$, we have, by Lemma 4.4,

$$u^k \notin \bigcup_{i=1}^{k-1} B_m(u^i, \epsilon(L_2 K + L_3)^{-1}). \tag{4.8}$$

On the other hand, since $U \subset R^m$ is compact, we can find a box $Q \subset R^m$ containing U. Let β be the side length of Q and M be the smallest integer that is greater than $\sqrt{m}\,\beta/(\epsilon(L_2 K + L_3)^{-1})$, i.e.,

$$M = [\sqrt{m}\,\beta/(\epsilon(L_2 K + L_3)^{-1})] + 1.$$

Let us divide Q into M^m subboxes of the same m-content, Q^i, $i = 1, \ldots, M^m$, satisfying $\text{int}(Q^i) \cap \text{int}(Q^j) = \emptyset$ for all $i \neq j$, where $\text{int}(Q^i)$ denotes the interior of Q^i. For every u^k generated by the algorithm, we can find a subbox Q^{i_k} containing u^k. Since the diameter of Q^{i_k} is $\sqrt{m}\,\beta/M$ and $\sqrt{m}\,\beta/M \leq \epsilon(L_2 K + L_3)^{-1}$, we have

$$B_m(u^k, \epsilon(L_2 K + L_3)^{-1}) \supset Q^{i_k}. \tag{4.9}$$

From (4.8) and (4.9) we know that $u^k \notin \bigcup_{j=1}^{k-1} Q^{i_j}$ and $Q^{i_k} \notin \{Q^{i_1}, \ldots, Q^{i_{k-1}}\}$. Consequently, the following theorem is established.

Theorem 4.4. *If assumptions (A5),(A6), and (A7) are satisfied, then an upper bound of the number of major iterations needed for finding an ϵ-optimal solution is:* $([\sqrt{m}\,\beta/(\epsilon(L_2 K + L_3)^{-1})] + 1)^m$. ∎

4.5 Nonlinear Semi-Infinite Programming

We now show that Algorithm 4.1 can be generalized to solve the following nonlinear semi-infinie program

(SIN) minimize $f(x)$

subject to

$$g(x,u) \geq 0, \text{ for all } u \in U$$

$$x \in S,$$

where U and S are nonempty compact convex sets, $f(x)$ is continuous on S, and $g(x,u)$ is continuous on $S \times U$.

Definition 4.3 (ϵ-optimal solution). Let $\epsilon > 0$. A vector $\bar{x}$ is an ϵ-optimal solution of (SIN) if $g(\bar{x},u) \geq -\epsilon$ for all $u \in U$ and $f(\bar{x}) \leq v(\text{SIN})$, where $v(\text{SIN})$ denotes the optimal objective function value of (SIN).

Algorithm 4.2.

Step 1 (Initialization).

Let $k := 0$;

let α be a constant such that $0 < \alpha < 1$;

let NP(0) be the nonlinear program

minimize $f(x)$

subject to

$$x \in S.$$

Step 2.

If NP(k) is infeasible, then (SIN) is infeasible, stop.

Else, find an optimal solution x^k of NP(k);

if $g(x^k,u) \geq 0$ for all $u \in U$,

then x^k is an optimal solution of (SIN), stop.

Step 3.

Find a $u^k \in U$ such that $g(x^k,u^k) \leq \alpha \cdot \min\{g(x^k,u) : u \in U\}$;

form NP($k+1$) by adding a cut, $g(x,u^k) \geq 0$, to NP(k);

$k := k+1$;

go to Step 2.

Notice that the convergence proof of Algorithm 4.1 is based on the uniform continuity of $G(x, y, z)$ on $T \times W$, and now $g(x, u)$ is uniformly continuous on $S \times U$. It is easy to see that the following theorem holds.

Theorem 4.5. *Suppose that U and S are compact, $g(x, u)$ is continuous on $S \times U$, and the algorithm does not stop finitely. Let the sequence $x^1, x^2, \ldots$ be generated by Step 2 of Algorithm 4.2. Then,*

(a) (Finite ϵ-convergence) *For any ϵ satisfying $0 < \epsilon < \alpha$, there exists an integer $N(\epsilon)$ such that x^k is an ϵ-optimal solution of* (SIN) *for all $k \geq N(\epsilon)$.*

(b) (Convergence) *Any cluster point of the sequence $x^1, x^2, \ldots$ is an optimal solution of* (SIN). ∎

Remark 4.2. If NP(k) is a difficult nonlinear program to solve, one might want to find instead an $\bar{x}$ such that $g(\bar{x}, u) \geq -\epsilon$ for all $u \in U$ and $f(\bar{x}) \leq v(\text{SIN}) + \delta$, where $\delta > 0$. In this case, the optimal solution x^k of NP(k) generated in Step 2 of Algorithm 4.2 may be replaced by any $\hat{x}^k$ that is feasible for NP(k) and that satisfies $f(\hat{x}^k) \leq v(\text{NP}(k)) + \delta$.

Chapter 5

Applications to Convex Programming

5.1 *Introduction*

The idea of solving an ordinary convex program by semi-infinite linear programming comes from Dantzig (1963) Chapter 24. There he showed how to solve an ordinary convex program by generalized linear programming. His algorithm starts with a feasible solution of the convex program that provides a nondegenerate basic feasible solution for an initial restricted master program and then generates and solves a sequence of linear programs. As we have seen in Chapter 4, a generalized linear program can be considered as a semi-infinite linear program of the dual type. We now show how to apply Algorithm 4.1 to a certain semi-infinite linear program so as to obtain a feasible solution of the convex program. Using this feasible solution as a starting point, we then apply Algorithm 4.1 to another semi-infinite linear program and obtain an optimal solution of the convex program. In particular, for a strongly consistent convex program Algorithm 4.1 can find a feasible solution after a finite number of iterations.

5.2 Feasibility

We want to find a feasible solution of the convex program

$$\begin{aligned}(\text{CP}) \quad & \text{minimize} \quad f(x) \\ & \text{subject to} \\ & \qquad g_i(x) \le 0 \text{ for } i = 1, \ldots, m \\ & \qquad x \in T,\end{aligned}$$

where $T \subseteq R^n$ is a polytope, and $f(x)$ and $g_i(x)$ for $i = 1, \ldots, m$ are real valued convex functions on R^n.

First, we construct a semi-infinite linear program whose optimal solutions can provide feasible solutions to (CP).

Let $G(x) = (1, -g_1(x), \ldots, -g_m(x))^T$ and e^i be the i-th unit vector in R^{m+1}. Consider the following dual pair of semi-infinite linear programs:

$$\begin{aligned}(\text{SL}) \quad & \text{maximize} \quad {e^1}^T y \\ & \text{subject to} \\ & \qquad G(x)^T y \le 0 \text{ for all } x \in T \\ & \qquad 0 \le y_i \le 1 \text{ for } i = 2, \ldots, m+1,\end{aligned}$$

and for all finite subset $\{x^i : i \in \Delta\}$ of T

$$\begin{aligned}(\text{SD}) \quad & \text{minimize} \quad s_1 + \cdots + s_m \\ & \text{subject to} \\ & \qquad \textstyle\sum_{i \in \Delta} G(x^i)\lambda_i - \sum_{i=1}^m e^{i+1}\mu_i + \sum_{i=1}^m e^{i+1}s_i = e^1 \\ & \qquad \lambda_i \ge 0 \text{ for all } i \in \Delta \\ & \qquad \mu_i, s_i \ge 0 \text{ for } i = 1, \ldots, m.\end{aligned}$$

Lemma 5.1. *Algorithm 4.1 converges for* (SL) *and* (SD), *and* $v(\text{SL}) = v(\text{SD})$.

Proof. Let us pick any $x^0 \in T$ and start Algorithm 4.1 with the linear program

LP(0) maximize $e^{1T}y$

subject to

$$G(x^0)^T y \leq 0$$

$$0 \leq y_i \leq 1 \text{ for } i = 2, \ldots, m+1.$$

It is easy to see that LP(0) has an optimal solution. Then, Algorithm 4.1 generates a sequence of linear programs

LP(k) maximize $e^{1T}y$

subject to

$$G(x^i)^T y \leq 0 \text{ for } i = 0, 1, \ldots, k$$

$$0 \leq y_i \leq 1 \text{ for } i = 2, \ldots, m+1,$$

where $x^i \in T$ for $i = 0, 1, \ldots, k$. The dual of LP(k) is

DP(k) minimize $s_1 + \cdots + s_m$

subject to

$$\sum_{i=0}^{k} G(x^i)\lambda_i - \sum_{i=1}^{m} e^{i+1}\mu_i + \sum_{i=1}^{m} e^{i+1}s_i = e^1$$

$$\lambda_i \geq 0 \text{ for } i = 0, 1, \ldots, k$$

$$\mu_i, s_i \geq 0 \text{ for } i = 1, \ldots, m.$$

Let y^k be the optimal solution of LP(k) and (λ^k, μ^k, s^k) be the optimal solution of DP(k) generated by Algorithm 4.1. To show that Algorithm 4.1 converges for (SL) and (SD), we only need to show that y_1^k for $k = 0, 1, 2, \ldots$ is bounded. Indeed, by the duality theorey of linear programming, we have

$$y_1^k = e^{1T}y^k = s_1^k + \cdots + s_m^k \geq 0.$$

On the other hand, since y^k is a feasible solution of LP(k), we have

$$y_1^k \leq g_1(x^0)y_2^k + \cdots + g_m(x^0)y_{m+1}^k \leq \sum_{i=1}^{m} |g_i(x^0)|.$$

Hence, y^k for $k = 0, 1, 2, \ldots$ is bounded. Let y^* be a cluster point of the sequence y^k. Then, by Theorems 4.1 and 4.2, we know that Algorithm 4.1 converges for (SL) and (SD) and $v(\text{SL}) = v(\text{SD})$. ∎

Lemma 5.2. (CP) *is feasible if and only if* $v(\mathrm{SD}) = 0$.

Proof. If (CP) is feasible, i.e., there exists an $x^1 \in T$ satisfying $g_i(x^1) \leq 0$ for $i = 1, \ldots, m$, let $\Delta = \{1\}$, $\lambda_1^1 = 1$, $\mu_i^1 = -g_i(x^1) \geq 0$ for $i = 1, \ldots, m$, and $s_i^1 = 0$ for $i = 1, \ldots, m$. Then, we have

$$G(x^1)\lambda_1^1 - \sum_{i=1}^{m} e^{i+1}\mu_i^1 + \sum_{i=1}^{m} e^{i+1}s_i^1 = e^1 \text{ and } \sum_{i=1}^{m} s_i^1 = 0.$$

Since $v(\mathrm{SD}) \geq 0$ and we have a feasible solution of (SD) whose objective function value attains zero, we know that $v(\mathrm{SD}) = 0$. Conversely, if $v(\mathrm{SD}) = 0$, then by Lemma 5.1 we have

$$\lim_{k\to\infty} v(\mathrm{DP}(k)) = v(\mathrm{SD}) = v(\mathrm{SL}) = 0.$$

First, let us consider the case $v(\mathrm{DP}(k)) = 0$ for some k. Let (λ^k, μ^k, s^k) be the optimal solution of DP(k) generated by Algorithm 4.1 and $\hat{x}^k = \sum_{i=0}^{k} \lambda_i^k x^i$. By the feasibility of λ^k, we have $\lambda_i^k \geq 0$ and $\sum_{i=0}^{k} \lambda_i^k = 1$, and thus $\hat{x}^k \in T$. By the convexity of $g_j(x)$ and the feasibility of (λ^k, μ^k, s^k), we have (for all $j = 1, \ldots, m$)

$$\begin{aligned} g_j(\hat{x}^k) &= g_j(\sum_{i=0}^{k} \lambda_i^k x^i) \\ &\leq \sum_{i=0}^{k} \lambda_i^k g_j(x^i) \\ &= -\mu_j^k \\ &\leq 0. \end{aligned}$$

Namely, $\hat{x}^k$ is a feasible solution of (CP). Now let us consider the case $v(\mathrm{DP}(k)) > 0$ for all k and $\lim_{k\to\infty} v(\mathrm{DP}(k)) = 0$. Let (λ^k, μ^k, s^k) be the optimal solution of DP(k) generated by Algorithm 4.1, and $\hat{x}^k = \sum_{i=0}^{k} \lambda_i^k x^i \in T$ for all $k = 0, 1, \ldots$. Since $\sum_{i=1}^{k} s_i^k \leq \sum_{i=1}^{k} s_i^0$ and $s^k \geq 0$ for all k, the sequence s^k is bounded. Therefore, the sequence $\mu^k = \sum_{i=0}^{k} G(x^i)\lambda_i^k + s^k$ is bounded. As T is convex and compact, we know that the sequence $\hat{x}^k$ has a cluster point x^*. Without loss of generality, we assume that $\hat{x}^k \to x^*$, $\mu^k \to \mu^*$, and $s^k \to s^*$. Then, by

the convexity and continuity of $g_j(x)$ and the feasibility of (λ^k, μ^k, s^k), we have

$$\begin{aligned} g_j(x^*) &= g_j(\lim_{k\to\infty} \hat{x}^k) \\ &= \lim_{k\to\infty} g_j(\hat{x}^k) \\ &\le \lim_{k\to\infty} \sum_{i=0}^{k} \lambda_i^k g_j(x^i) \\ &= \lim_{k\to\infty} (-u_j^k + s_j^k) \\ &\le 0. \end{aligned}$$

Hence, x^* is a feasible solution of (CP). ∎

So far we have shown how to find a feasible solution of (CP) by applying Algorithm 4.1 to (SL) and (SD). If (CP) has a feasible solution, then in the case $v(\mathrm{DP}(k)) = 0$ for some k, $\hat{x}^k$ is a feasible solution of (CP) and Algorithm 4.1 stops finitely; otherwise, $v(\mathrm{DP}(k)) \to 0$ as $k \to \infty$ and hence, for any $\epsilon > 0$, there exists a number $N(\epsilon) > 0$ such that $0 < v(\mathrm{DP}(k)) < \epsilon$ for all $k \ge N(\epsilon)$. This implies that for all $k \ge N(\epsilon)$, we have

$$g_j(\hat{x}^k) \le \sum_{i=0}^{k} \lambda_i^k g_j(x^i) = -\mu_j^k + s_j^k \le \epsilon.$$

That is, for *any* $\epsilon > 0$ Algorithm 4.1 can find an x that satisfies $g_j(x) \le \epsilon$ for all $j = 1, \ldots, m$ after a *finite number of iterations*. If we know that there exists a feasible solution x^o of (CP) and a small positive number δ such that $g_j(x^o) \le -\delta < 0$ for all $j = 1, \ldots, m$, then let $\bar{g}_j(x) = g_j(x) + \delta$. As we have just shown, for $\epsilon = \delta/2 > 0$, Algorithm 4.1 can find an $\bar{x}$ that satisfies $\bar{g}_j(\bar{x}) \le \epsilon$ after finitely many iterations. Since $\bar{g}_j(\bar{x}) = g_j(\bar{x}) + \delta$, we have $g_j(\bar{x}) \le -\delta/2 < 0$ for all $j = 1, \ldots, m$. Thus, the following lemma is established.

Lemma 5.3. *If $g_j(x) \le \delta$ for all $j = 1, \ldots, m$ has solutions, where δ is a small positive number, then Algorithm 4.1 can find a feasible solution of* (CP) *after a finite number of iterations.*

5.3 Optimality

We have seen how to find a feasible solution of a convex program (CP) by solving a semi-infinite linear progeram (SD). Now we show how to find an optimal solution of (CP). Let

$\bar{G}(x) = (1, g_1(x), \ldots, g_m(x))^T$ and e^i be the i-th unit vector in R^{m+1}. Consider the dual pair of semi-infinite linear programs:

(CSL) maximize $e^{1T}y$

subject to

$$\bar{G}(x)^T y \leq f(x) \text{ for all } x \in T$$

$$y_i \leq 0 \text{ for } i = 2, \ldots, m+1,$$

and for all finite subset $\{x^i : i \in \Delta\}$ of T

(CSD) minimize $\sum_{i\in\Delta} f(x^i)\lambda_i$

subject to

$$\sum_{i\in\Delta} \bar{G}(x^i)\lambda_i + \sum_{i=1}^{m} e^{i+1}\mu_i = e^1$$

$$\lambda_i \geq 0 \text{ for all } i \in \Delta$$

$$\mu_i \geq 0 \text{ for } i = 1, \ldots, m.$$

Lemma 5.4. *If* (CP) *is strongly consistent, then Algorithm 4.1 converges for* (CSL) *and* (CSD), *and* v(CSL) = v(CSD).

Proof. Find an x^0 satisfying $g_i(x^0) < 0$ for $i = 1, \ldots m$ and start Algorithm 4.1 with the linear program

LP(0) maximize $e^{1T}y$

subject to

$$\bar{G}(x^0)^T y \leq f(x^0)$$

$$y_i \leq 0 \text{ for } i = 2, \ldots, m+1.$$

It is easy to see that LP(0) has an optimal solution. Then, Algorithm 4.1 generates a sequence of linear programs

LP(k) maximize $e^{1T}y$

subject to

$$\bar{G}(x^i)^T y \leq f(x^i) \text{ for } i = 0, 1, \ldots, k$$

$$y_i \leq 0 \text{ for } i = 2, \ldots, m+1,$$

where $x^i \in T$ for $i = 0, 1, \ldots, k$. The dual of LP(k) is

DP(k) minimize $\sum_{i=0}^{k} f(x^i)\lambda_i$

subject to

$$\sum_{i=0}^{k} \bar{G}(x^i)\lambda_i + \sum_{i=1}^{m} e^{i+1}\mu_i = e^1$$

$$\lambda_i \geq 0 \text{ for } i = 0, 1, \ldots, k$$

$$\mu_i \geq 0 \text{ for } i = 1, \ldots, m.$$

Let y^k be the optimal solution of LP(k) and (λ^k, μ^k, s^k) be the optimal solution of DP(k) generated by Algorithm 4.1. We want to show that the sequence y^k is bounded. By the optimality of y^k and convexity of $f(x)$, we have

$$y_1^k = e^{1^T} y^k = \sum_{i=0}^{k} f(x^i)\lambda_i^k \leq \sum_{i=0}^{k} |f(x^i)|\lambda_i^k \leq \max_{x \in T} |f(x)|, \text{ and}$$

$$y_1^k = \sum_{i=0}^{k} f(x^i)\lambda_i^k \geq f(\sum_{i=0}^{k} \lambda_i^k x^i) \geq \min_{x \in T} f(x).$$

Thus, y_1^k is bounded. By the feasibility of y^k, we have

$$y_1^k + g_1(x^0)y_2^k + \cdots + g_m(x^0)y_{m+1}^k \leq f(x^0).$$

Since $g_i(x^0) < 0$ for all $i = 1, 2, \ldots, m$ and $y_i^k \leq 0$ for all $i = 2, \ldots, m+1$, we have (for all $k = 0, 1, \ldots$)

$$y_{i+1}^k \geq (f(x^0) - y_1^k)/g_i(x^0) \text{ for } i = 1, \ldots, m.$$

Thus y^k is bounded and Algorithm 4.1 converges for (CSL) and (CSD). ∎

Remark 5.1. If y^k is an optimal solution of (CSL), then $\hat{x}^k = \sum_{i=0}^{k} \lambda_i^k x^i$ ia an optimal solution of (CP). If Algorithm 4.1 converges in the limit, then any cluster point of the sequence $\hat{x}^j = \sum_{i=0}^{j} \lambda_i^k x^i$ is an optimal of (CP) (see Dantzig (1963) Chapter 24).

Remark 5.2. Suppose that (CP) is not strongly consistent and that one can find $x^i \in T$ $i = 1, \ldots, m$ satisfying $g_j(x^i) \leq 0$ for all $i = 1, \ldots, m$ and $j = 1, \ldots, m$, and $g_i(x^i) < 0$ for all $i = 1, \ldots, m$. Then, Algorithm 4.1 converges for (CSL) and (CSD) (see Lemma 4.2).

Bibliography

E.J. Anderson (1985). "A new primal algorithm for semi-infinite linear programming," in: E.J. Anderson and A.B. Philpott, ed., *Infinite Programming*, Springer-Verlag, Berlin, 1985, pp.108–122.

M. Araki (1975). "Application of M-matrices to stability problems of composite dynamical systems," *Journal of Mathematical Analysis and Applications* Vol.52, 1975, pp.309–321.

M. Avriel (1976). *Nonlinear programming: analysis and methods*, Prentice-Hall Inc., New Jersey, 1976.

G.P. Barker, A. Berman and R.J. Plemmons (1978). "Positive diagonal solutions to the Lyapunov equations," *Linear and Multi-linear Algebra* Vol.5, 1978, pp.249–256.

A. Berman and D. Hershkowitz (1983). "Matrix diagonal stability and its implications," *SIAM Journal on Algebraic and Discrete Methods* Vol.4, 1983, pp.377–382.

A. Charnes, W.W. Cooper and K. Kortanek (1965). "On representations of semi-infinite programs which have no duality gaps," *Management Science* Vol.12, 1965, pp.113–121.

A.R. Conn and N.I.M. Gould (1987). "An exact penalty function for semi-infinite programming," *Mathematical Programming* Vol.37, 1987, pp.19–40.

I.D. Coope and G.A. Watson (1985). "A projected Lagrangian algorithm for semi-infinite programming," *Mathematical Programming* Vol.32, 1985, pp.337–356.

R.W. Cottle (1964). "Nonlinear programs with positively bounded Jacobians," Ph.D. Thesis, University of California, Berkeley, 1964.

G.W. Cross (1978). "Three types of matrix stability," *Linear Algebra and its Applications* Vol.20, 1978, pp.253–263.

G.B. Dantzig (1983). "Can Leontief and P-matrices be rescaled positive definite," Technical Report, SOL 83-23, Department of Operations Research, Stanford University, 1983.

G.B. Dantzig (1963). *Linear programming and extensions*, Princeton University Press, Princeton, New Jersey, 1963.

G.B. Dantzig (1960). "General convex objective forms," in: K.J. Arrow, S. Karlin, and P. Suppes, ed., *Mathematical Methods in Social Sciences*, Stanford University Press, Stanford, California, 1960, pp.151–158.

G.B. Dantzig, P.H. McAllister, and J.C. Stone (1988). "Deriving a utility function for the economy," Technical Report, SOL 88-6, Department of Operations Research, Stanford University, 1988.

L.L. Dines and N.H. McCoy (1933). "On linear inequalities," *Transactions Royal Society of Canada Section III* , 1933, pp.37-70.

R.J. Duffin, R.G. Jeroslow and L.A. Karlovitz (1983). "Duality in semi-infinite linear programming," in: A.V. Fiacco and K.O. Kortanek, ed., *Semi-infinite programming and applications*, Springer-Verlag, New York, 1983, pp.50-62.

R.J. Duffin and L.A. Karlovitz (1965). "An infinite linear program with a duality gap," *Management Science* Vol.12, 1965, pp.122-134.

R. Fletcher (1981). "A nonlinear programming problem in statistics (educational testing)," *SIAM Journal on Scientific and Statistical Computing* Vol.2 No.3, 1981, pp.257-267.

P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright (1984). "User's guide for QPSOL: a FORTRAN package for quadratic programming," Technical Report, SOL 84-6, Department of Operations Research, Stanford University, 1984.

S.-A. Gustafson and K.O. Kortanek (1973). "Numerical treatment of a class of semi-infinite programming problems," *Naval Research Logistics Quarterly* Vol.20, 1973, pp.477-504.

D. Hershkowitz and H. Schneider (1985). "Scalings of vector spaces and the uniqueness of Lyapunov scaling factors," *Linear and Multi-linear Algebra* Vol.17, 1985, pp.203-226.

R. Hettich (1979). "A comparison of some numerical methods for semi-infinite programming," in: R. Hettich ed., *Semi-infinite programming*, Springer-Verlag, New York, 1979, pp.112-125.

A.J. Hoffman (1952). "On approximate solutions of systems of linear inequalities," *Journal of Research of the National Bureau of Standards* Vol.49, 1952, pp.263-265.

E. Isaacson and H.B. Keller (1966). *Analysis of numerical methods*, John Wiley & Sons, Inc., New York, 1966.

F. John (1948). "Extremum problem with inequality as subsidiary conditions," *Courant Anniversary Volume, Interscience*, 1948.

D.E. Karney (1983). "A duality theorem for semi-infinite convex programs and their finite subprograms," *Mathematical Programming* Vol.27, 1983, pp.75-82.

D.F. Karney (1981). "Duality gaps in semi-infinite linear programming — an approximation problem," *Mathematical Programming* Vol.20, 1981, pp.129–143.

H.K. Khalil (1982). "On the existence of positive diagonal P such that $PA + A^T P < 0$," *IEEE Transactions on Automatic Control* Vol.27, 1982, pp.181–184.

L.J. Lau (1978). "Testing and imposing monoticity, convexity and quasi-convexity constraints," in M. Fuss and D. McFadden, ed., " *Production economics: a dual approach to theory and applications,"* North-Holland Publishing Company, New York, 1978.

B.A. Murtagh and M.A. Saunders (1983). "MINOS 5.0 user's guide," Technical Report, SOL 83–20, Department of Operations Research, Stanford University, 1983.

E. Polak (1983). "Semi-infinite optimization in engineering design," in: A.V. Fiacco and K.O. Kortanek, ed., *Semi-infinite programming and applications*, Springer-Verlag, New York, 1983, pp.236–248.

R.T. Rockafellar (1972). *Convex analysis*, Princeton University Press, Princeton, New Jersey, 1972.

L. Tartar (1971). "Une nouvelle caracterisation des M matrices," *Revue Française d'Informatique et de Recherche Opérationnelle* Vol.5, 1971, pp.127–128.

G.A. Watson (1985). "Lagrangian methods for semi-infinite programming problems," in: E.J. Anderson and A.B. Philpott, ed., *Infinite Programming*, Springer-Verlag, Berlin, 1985, pp.90–107.

J. H. Wilkinson (1965). *The algebraic eigenvalue problem*, Oxford University Press, Oxford, 1965.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report SOL 89-2	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Semi-Infinite Programming		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Hui Hu		8. CONTRACT OR GRANT NUMBER(s) N00014-89-J-1659
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305-4022		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1111MA
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		12. REPORT DATE March 1989
		13. NUMBER OF PAGES 59 pages
		14. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Semi-Infinite Programming, Duality, Positive Definite Matrix, Least-Square Estimation.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(Please see reverse side)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOL 89-2:
Semi-Infinite Programming, by Hui Hu

Semi-Infinite programming, that allows for either infinitely many constraints or infinitely many variables but not both, is a natural extension of ordinary mathematical programming. There are many practical as well as theoretical problems in which the constraints depend on time or space and thus can be formulated as semi-infinite programs. The focus of this dissertation is on formulating and solving semi-infinite programming problems. The main results include:

(1) An algorithm for solving a matrix rescaling problem formulated as a semi-infinite linear program. Sufficient conditions that guarantee finite termination are discussed and computational results are reported.

(2) An algorithm for solving a matrix estimation problem equivalent to a semi-infinite quadratic program. For a specified constant, this algorithm will find an approximate solution after finitely many iterations, or will tend to an optimal solution in the limit. An upper bound on the total number of iterations needed for finding an approximate solution is given. Computational results are reported.

(3) A one-phase algorithm for solving a large class of semi-infinite linear programming problems. This algorithm has several advantages: it handles feasibility and optimality together and can detect infeasibility after a finite number of iterations; it has very weak restrictions on the constraints; it allows cuts that are not near the most violated cut; and it solves the primal and the dual problems simultaneously. Upper bounds for finding an ϵ-optimal solution and for the distance between an ϵ-optimal solution and an optimal solution are given.

(4) Applications of the above algorithm to convex programming. First, a certain semi-infinite linear program is solved by this algorithm so as to obtain a feasible solution of a convex program. Then, another semi-infinite linear program is solved by this algorithm so as to obtain an optimal solution of the convex program. In particular, it is shown that for a strongly consistent convex program this algorithm can find a feasible solution after a finite number of iterations.