A CONTINUATION APPROACH FOR SOLVING NONLINEAR OPTIMIZATION PROBLEMS WITH DISCRETE VARIABLES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MANAGEMENT SCIENCE AND ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Kien-Ming Ng
June 2002

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Walter Murray
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Richard W. Cottle

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Michael A. Saunders

Approved for the University Committee on Graduate Studies:

Abstract

A new approach to solving nonlinear optimization problems with discrete variables using continuation methods is described. Our focus is on pure integer nonlinear optimization problems with linear equality constraints (ILENP) but we show how the technique can be extended to more general classes of problems such as those involving linear inequality and mixed integer constraints.

We transform the ILENP into that of finding the global minimizer of a problem with continuous variables over the unit hypercube. To reduce the difficulties caused by the introduction of undesirable local minimizers, we employ a special class of continuation methods, called smoothing methods. These methods deform the original objective function into a function whose smoothness is controlled by a parameter. The success of the approach depends on finding a good smoothing function. We show that the logarithmic barrier function is ideal for the type of global optimization problem that arises from transforming discrete problems of interest.

The continuous problems arising from the smoothed function are solved by a modified Newton method. Since there are only linear equality constraints, we just need to solve the reduced Newton equations for each smoothing parameter. The conjugate gradient algorithm is applied to this set of equations to obtain a descent direction before applying linesearch procedures. When nonconvexity of the transformed objective function arises, it is necessary to obtain a good direction of negative curvature for the linesearch procedures. Issues related to the implementation of the algorithm, such as the termination criteria and the use of preconditioners, are also discussed.

We show the effectiveness of the approach by applying it to a number of real

problems and also test problems taken from the literature. These include the binary unconstrained quadratic problem, the frequency assignment problem and the quadratic assignment problem. The results are compared to those from alternative methods, indicating that the new approach is able to produce good-quality solutions for diverse classes of nonlinear discrete optimization problems.

Acknowledgements

I would like to take this chance to express my heartfelt gratitude to everyone who has made it possible for me to successfully complete this thesis. First, I would like to thank the Department of Engineering-Economic Systems and Operations Research (EESOR) and its faculty for admitting me to the PhD. program in 1997, which opened the door to my Ph.D. studies here at Stanford University.

I am deeply indebted to my advisor, Professor Walter Murray, for taking me as his student and giving me all the motivation and guidance I needed in my research work. His influence was instrumental to my academic performance, especially in making me realize the importance of striking a proper balance between theory and practice in my research. It amazes me how he is able to view research problems from many perspectives, and develop a variety of new techniques to solve them. Because he is never short of unique insights and creative ideas, it is always very enlightening to have a conversation with him. I thank Professor Murray for all the feedback, support and guidance he has given me in writing this thesis too.

I would also like to thank Professor Richard W. Cottle and Professor Michael A. Saunders for taking on the responsibilities of being on both my dissertation reading committee and my oral examination committee, and carefully proofreading my thesis for errors in technical details, English grammar and style despite their busy schedules. I am particularly grateful to Professor Saunders for taking the time to discuss my research work with me, and for the many helpful suggestions he has given to improve the efficiency of the computer programs that I wrote in the course of my research. I am also grateful to Professor Cottle for all the opportunities he has given me to work as his course assistant, and all the useful knowledge I have gained from these experiences.

In addition, I would like to thank both Professor Steven M. Gorelick and Professor Gerd Infanger who also served on my oral examination committee, as well as Professor B. Curtis Eaves and Professor Arthur F. Veinott, Jr. for their advice and insights about my thesis research. I would also like to thank all the staff in both the Department of EESOR and the Department of Management Science and Engineering (MS&E) who assisted me with many administrative issues pertaining to my research work. The funding from both departments, the National Science Foundation Grant NSF CCR-9988205, the Office of Naval Research Grant N00014-96-1-0274, and the General Motors Corporation have also been essential in supporting my research financially.

The Systems Optimization Laboratory has been a great working environment, and I particularly appreciate the friendship and many thought-provoking discussions I have enjoyed with my colleagues Del Gatto Antonino, Erik Boman, Alexis Guillaume Collomb, Angel-Victor de Miguel, Titus Dorstenstein, Maureen Doyle, Michael Friedlander, Byunggyoo Kim, Chih-Hung Lin, Heiko Pieper, Vinayak Vishnu Shanbhag and Che-Lin Su. There are many more friends I have made at Stanford University who gave me help and support but were not mentioned here. I would like to take this opportunity to thank all of them.

My girlfriend Lee Hwei Yi has always provided me with inspiration and emotional support since we met during my 4th year at Stanford University. I want to thank her for giving me motivation and patiently waiting for me to complete my thesis.

Lastly, my parents' love and constant encouragement has played a great role in sustaining me through the rigours of my 5 years of Ph.D. studies. Despite the fact that I am thousands of miles away from them, my welfare has always been uppermost in their minds. I want to say a big thank you to them for all their sacrifices they have made over the years in order to see me succeed in my quest for a career in academia.

Contents

List of Tables

List of Figures

Notation

$\mathbb{R}$	set of all real numbers
$\mathbb{Z}$	set of all integers
$\mathbb{Z}^+$	set of all positive integers
e	$[1, 1, \ldots, 1]^{\mathrm{T}}$
x	decision variables (an n-vector)
$\|x\|$	norm of x (2-norm unless otherwise stated)
$\|A\|$	norm of A (2-norm unless otherwise stated)
$f(x)$	objective function (a scalar)
∇f	gradient of f (a column vector)
$\nabla^2 f$	Hessian of f (an $n \times n$ matrix)
C^n	set of functions with continuous nth order derivatives
$X, \mathrm{Diag}(x)$	diagonal matrix with diagonal elements being vector x
$\mathrm{diag}(X)$	vector of the diagonal elements of square matrix X
$A \otimes B$	Kronecker product of matrices A and B
$\mathrm{vec}(X)$	vector of length mn formed by columns of matrix $X \in \mathbb{R}^{m \times n}$

Chapter 1

Introduction

Our interest is in nonlinear optimization problems in which some or all of the variables are discrete. The most general form of the problem of interest to us may be stated as

$$
\begin{array}{ll}
\text{Minimize} & f(x,y) \\
\text{subject to} & g(x,y) = 0 \\
 & h(x,y) \leq 0 \\
 & x \in D \subset \mathbb{Z}^p, y \in \mathbb{R}^q,
\end{array}
\tag{P1.1}
$$

where $f : \mathbb{R}^{p+q} \to \mathbb{R}$, $g : \mathbb{R}^{p+q} \to \mathbb{R}^m$, $h : \mathbb{R}^{p+q} \to \mathbb{R}^l$ are assumed to be C^2 functions, and D is a bounded subset of $\mathbb{Z}^p$. The decision variables represented by x and y are called *discrete* and *continuous* variables respectively.

If the variables x do not arise in (P1.1) so that it becomes a nonlinear optimization problem with continuous variables only, then there is a plethora of algorithms to solve the problem [BSS93, GMW81, NW99b]. The introduction of discrete variables to (P1.1), even if it is a relatively small number when compared to the continuous variables, makes it much harder to solve. In fact, there are few algorithms dealing with such problems because of their inherent difficulty. Examples include algorithms underlying the solver DICOPT in the software GAMS[1], as well as the global optimization solver BARON[2]. These algorithms are discussed later in this chapter.

[1]http://www.gams.com

[2]http://archimedes.scs.uiuc.edu/baron/baron.html

To focus on aspects of the algorithm pertaining to the inclusion of discrete variables, we study a subclass of (P1.1) in which all the variables are discrete and the constraints are linear:

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & Ax = b \\ & Cx \leq d \\ & x \in D \subset \mathbb{Z}^p. \end{array} \tag{P1.2}$$

Here and in subsequent sections, we shall assume without a loss of generality that A is of full rank. How the ideas we introduce may be extended to solve problems that include continuous variables is discussed in Chapter 7. However, the class of problems represented by (P1.2) is of interest in its own right and many practical problems are of this form.

1.1 How Discrete Variables Arise

Discrete variables arise in many optimization problems, and they sometimes, but not always, occur in conjunction with continuous variables. Unlike continuous variables, discrete variables are of various types, and this distinction can be important. How they arise in the problem can also vary. For example, if we have a function $f(x)$, $x \in \mathbb{R}^n$ and say $x_1 \in \{0, 1\}$, it may or may not be possible to evaluate $f(x)$ unless $x_1 \in \{0, 1\}$. We shall examine these different characteristics later in this chapter.

A common reason for discrete variables occurring is when resources of interest have to be measured in terms of integer quantities, such as the number of components to be assembled in a production line, or the number of people to be assigned for certain jobs. If a variable is defined to represent the amount of such resources to be used, it follows that this variable is discrete.

Discrete variables may be introduced to facilitate the modeling process, such as using binary or 0-1 variables (i.e., variables that can only take the values of 0 or 1) to represent "yes–no" decisions. A classical example that employs binary variables in this way is the knapsack problem. In this problem, there are n items that could be placed into a knapsack. The jth item has weight w_j and value c_j. The objective is to maximize the total value of the items placed in the knapsack subject to a constraint that the weight of the items not exceed b. To formulate this problem, one can let x_j be the binary variable such that

$$x_j = \begin{cases} 1 & \text{if item } j \text{ is placed in the knapsack,} \\ 0 & \text{otherwise.} \end{cases}$$

Then the problem becomes the following

$$\begin{array}{ll} \text{Maximize} & c^{\mathrm{T}}x \\ \text{subject to} & w^{\mathrm{T}}x \leq b \\ & x \in \{0,1\}^n. \end{array} \tag{P1.3}$$

Note that this is a special case of (P1.2).

It is also possible to use discrete variables to help model constraints that involve logical conditions. For example, suppose we want $x_1 \geq 0 \Leftrightarrow x_2 \leq 0$, and also $x_2 \geq 0 \Leftrightarrow x_1 \leq 0$. We can certainly introduce the constraint $x_1 x_2 \leq 0$ to represent such a logical condition, but it may be desirable to preserve the linearity of the optimization problem. To this end, we can instead include the two linear constraints $-M(1-y) \leq x_1 \leq My$ and $-My \leq x_2 \leq M(1-y)$, where y is a binary variable and M is a sufficiently large positive number that does not affect the feasibility of the problem. By this definition of M, if $y = 1$, we will have $x_1 \geq 0$ and $x_2 \leq 0$, while if $y = 0$, we will have $x_2 \geq 0$ and $x_1 \leq 0$.

Another common situation requiring integer variables is when the problem involves set-up costs. As an example, consider a generator supplying electricity to a local region with I nodes for T periods. Suppose at period t, the generator incurs a cost of s_t when it is turned on, a cost of p_t for producing electricity after it is turned on, a cost of s_i for supplying electricity to node i after it is turned on, and a cost of d_t for shutting it down. For $t \in \{1,\ 2,\ \dots,\ T\}$, let x_t, y_t and z_t denote the binary variables such that

$$x_t = \begin{cases} 1 & \text{if generator is turned on in period } t, \\ 0 & \text{otherwise.} \end{cases}$$

$$y_t = \begin{cases} 1 & \text{if generator is operating in period } t, \\ 0 & \text{otherwise,} \end{cases}$$

$$z_t = \begin{cases} 1 & \text{if generator is shut down in period } t, \\ 0 & \text{otherwise.} \end{cases}$$

If we let w_{it} be variables that represent the percentage of the generator's capacity c_i for node $i \in \{1, 2, \ldots, I\}$ that is used in period t, then the total costs incurred would be $\sum_{t=1}^{T}(s_t x_t + p_t y_t + d_t z_t + \sum_{i=1}^{I} c_i s_i w_{it})$. The objective is to minimize the total costs subject to the constraints that the total demand, d_t, for electricity in each period t be met, i.e., $\sum_{i=1}^{I} c_i w_{it} \geq d_t$. In order to ensure that $w_{it} > 0$ only if y_t is 1, we include the constraint $0 \leq w_{it} \leq y_t$ for each i and each t. Thus if $w_{it} > 0$, y_t is forced to be 1 by that constraint and the fact that it is binary. We can also impose other constraints that would ensure a proper 0-1 value for each variable x_t and z_t whenever we have a feasible vector. The formulation of this problem is summarized below:

$$\begin{aligned} \text{Minimize} \quad & \sum_{t=1}^{T}(s_t x_t + p_t y_t + d_t z_t + \sum_{i=1}^{I} c_i s_i w_{it}) \\ \text{subject to} \quad & \sum_{i=1}^{I} c_i w_{it} \geq d_t \\ & 0 \leq w_{it} \leq y_t \\ & x_t \geq y_t - y_{t-1} \\ & z_t \geq y_{t-1} - y_t \\ & x_t \geq 0,\ z_t \geq 0,\ y_t \in \{0, 1\}. \end{aligned} \tag{P1.4}$$

To gain a better understanding of (P1.1), we first look at linear discrete optimization problems. Considerable work has been done in that area, and it is helpful to understand some of the common techniques used there.

1.2 Linear Discrete Optimization Problems

In many contexts, the term "integer programming" is used to describe linear discrete optimization problems. Such problems can be expressed in the form (P1.1) with f, g and h being linear. This means that the function f is of the form $c^{\mathrm{T}}x+d$ for some real column vector c and real number d, and the functions g and h are of the form $Ax-b$ for some real matrix A and real column vector b. This problem has been studied intensively [NW99a, PR88, Sch98]. It is a reflection of the difficulty of even the linear problem that it has proven necessary to develop a wide variety of algorithms to solve various subclasses of (P1.1), some of which are discussed in the next section.

1.2.1 Applications

There are many applications involving linear discrete optimization [BW01, CNW02, CSD01, HRS00, RS01, Shi00, TM99, Van01, VD02, YC02] and some of the problem classes that have been studied extensively are:

1. **Set Partitioning Problem**: Given a finite set X and a family $\mathcal{F}$ of subsets of X with a cost of c_j associated with each element j of $\mathcal{F}$, find a collection of members of $\mathcal{F}$ that is a partition of X and has the minimal cost sum of these members. Defining x to be a vector such that $x_j = 1$ if member j of $\mathcal{F}$ is to be included in the partition of X and 0 otherwise, we find that the problem is of the form of (P1.1) with $f(x) = c^{\mathrm{T}}x$, where $c = (c_j)_{j\in\mathcal{F}}$. Here A is a 0-1 matrix such that each row i corresponds to an element of X, and each column j corresponds to an element of $\mathcal{F}$, i.e., $a_{ij} = 1$ if $i \in j$ and 0 otherwise. Also, $b = e$, the vector of ones and $D = \{0,1\}^{|\mathcal{F}|}$. Such problems arise frequently in airline crew scheduling problems. In these problems, each row represents a flight leg (takeoff and landing) that must be flown and each column represents a round-trip shift, i.e., a sequence of flight legs beginning and ending at the same base location and allowable under work regulations that an airline crew might fly. Each assignment of a crew to a particular round-trip shift j will have a certain cost c_j, and the matrix A consists of elements a_{ij} that take the value of 1 if flight leg i is on shift j and 0 otherwise.

2. **Generalized Linear Assignment Problem**: This class of problems involves assigning n workers to n jobs in such a way that exactly one worker has to be assigned to each job. Each worker i has a capacity b_i, while each job j has a size a_{ij} and a cost of c_{ij} when it is assigned to the ith worker. The aim is to find an assignment of workers to jobs that minimizes the overall cost. Defining x_{ij} to be 1 if the ith worker is assigned to the jth job and 0 otherwise, the problem can be formulated as

$$\begin{array}{rll} \text{Minimize} & \sum_i \sum_j c_{ij} x_{ij} & \\ \text{subject to} & \sum_j a_{ij} x_{ij} \leq b_i, & \text{for all } i \\ & \sum_i x_{ij} = 1, & \text{for all } j \\ & x_{ij} \in \{0,1\}, & \text{for all } i,\ j. \end{array} \tag{P1.5}$$

3. **Integer Network Flow Problem**: Given a network $G = (V, E)$, an arc flow x_{ij} is a nonnegative real number associated with an arc $a_{ij} \in E$, where $i, j \in V$. The flow that can pass through arc a_{ij} is constrained by an upper bound u_{ij} and a lower bound l_{ij}. A node $s \in V$ at which flow originates is called a *source* while a node $d \in V$ at which flow terminates is called a *destination*. The aim is to minimize the total cost of shipment through the network. If we restrict x_{ij} to take only integer values, and assuming the unit transport costs are c_{ij} and are linear in x_{ij}, the problem can be formulated as

$$\begin{array}{rll} \text{Minimize} & \sum_{a_{ij} \in E} c_{ij} x_{ij} & \\ \text{subject to} & \sum_{j: a_{ij} \in E} x_{ij} - \sum_{j: a_{ji} \in E} x_{ji} = 0, & \text{for all } i \neq s, d \\ & x_{ij} \in \mathbb{Z}^+ \cap [l_{ij}, u_{ij}], & \text{for all } i,\ j. \end{array} \tag{P1.6}$$

4. **Shortest Path Problem**: Assume that we have the same network G as in the Integer Network Flow Problem. A path P is defined to be a sequence of nodes

$i_1, \ldots, i_n$ in V, such that $a_{i_k i_{k+1}} \in E$ for each $k = 1, \ldots, n-1$, and $i_k \neq i_l$ for $k \neq l$. The length of a path P is defined as the sum of lengths of arcs in P, i.e., $l(P) = \sum_{a_{ij} \in P} c_{ij}$. The problem is to find a path P^* in G from s to d such that the length of this path is minimized, i.e., $l(P^*) = \min_P\{l(P)\}$, where the minimum is taken over all paths P in G from s to d. Defining x_{ij} to be 1 if arc a_{ij} is part of the shortest path and 0 otherwise, the problem can be formulated as

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{a_{ij} \in E} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{a_{sj} \in E} x_{sj} = 1, \ \text{for all } j \\
\text{subject to} \quad & \sum_{j: a_{ij} \in E} x_{ij} - \sum_{j: a_{ji} \in E} x_{ji} = 0, \ \text{for all } i \neq s, d \\
& x_{ij} \in \{0, 1\}, \ \text{for all } i, j \text{ such that } a_{ij} \in E.
\end{aligned}
\tag{P1.7}
$$

For certain classes of linear discrete optimization problems, a relaxation of (P1.1) without the constraint $x \in D$ may have an optimal solution vector $x^* \in D$; i.e., the optimal solution to the relaxed (P1.1) problem may be the optimal solution to the original (P1.1) problem. (we do not rely on such a property, but of course always welcome its occurrence.) The term "relaxation" has been defined in many contexts related to linear discrete optimization (see e.g., [NW99a]), but it can be extended to the following definition when general optimization problems are considered.

Definition 1.1. Given an optimization problem $\mathcal{P}$ defined by $\min\{f(x) : x \in X\}$, the optimization problem $\mathcal{R}$ defined by $\min\{\bar{f}(x) : x \in Y\}$ is said to be a *relaxation* of $\mathcal{P}$ if and only if $X \subset Y$ and $\bar{f}(x) \leq f(x)$ for all $x \in X$.

One important class of linear discrete optimization problems in which the optimal solution to the relaxation of (P1.1) gives the optimal solution to the original problem is

$$
\begin{array}{ll}
\text{Minimize} & c^{\mathrm{T}}x + d \\
\text{subject to} & Ax = b \\
 & x \geq 0 \\
 & x \in D \subset \mathbb{Z}^n,
\end{array}
\tag{P1.8}
$$

where the matrix A is unimodular as defined below.

Definition 1.2. A matrix $A \in \mathbb{Z}^{m \times n}$ is said to be *totally unimodular* if and only if $\det(B) = \pm 1$ for every nonsingular square submatrix B of A.

For completeness, the following theorem is stated and proved (see e.g., [VD68]):

Theorem 1.1. *In problem (P1.8), assume that A is totally unimodular, $b \in \mathbb{Z}^n$ and $\mathbb{Z}^n \cap \{x : Ax = b,\ x \geq 0\} \subset D$. If x^* is an optimal basic feasible solution to (P1.8) without the constraint $x \in D$, then x^* is also the optimal solution to (P1.8).*

Proof. Note that (P1.8) without the constraint $x \in D$ is a linear program and if x^* is an optimal basic feasible solution of this problem, then $x^* = B^{-1}b$, where B is a matrix formed from m columns of A that are linearly independent and such that

$$c_N - c_B B^{-1} N \geq 0.$$

Consider the adjoint matrix of B, $\operatorname{adj}(B)$, which is the transposed matrix of cofactors of A. Each entry of $\operatorname{adj}(B)$ is formed from the determinants of square submatrices of B. Since any square submatrix of B is also a square submatrix of A, we find by the totally unimodular property of A that $\operatorname{adj}(B) \in \mathbb{Z}^{n \times n}$ and $\det(B) = 1$ or -1. This implies that $x^* = B^{-1}b = \frac{1}{\det(B)}\operatorname{adj}(B)b \in \mathbb{Z}^n$. Since x^* is feasible, we have $Ax^* = b$, $x^* \geq 0$ and hence $x^* \in \mathbb{Z}^n \cap \{x : Ax = b\}$, i.e., $x^* \in D$. Thus, x^* is also an optimal solution to (P1.8). $\square$

1.2.2 Techniques to Solve Linear Discrete Problems

Though discrete optimization problems have finite or countable feasible points, they are not necessarily easier than continuous optimization problems. Indeed, they are usually considerably harder. A reflection of the degree of difficulty to solve problem (P1.8) can be seen from the large number of special algorithms that have been developed to solve special categories of problems (such as those described earlier), rather than a single all-purpose algorithm. To understand the effort required to solve discrete optimization problems, it is useful to employ the terminology used in complexity theory, which is described below informally.

A *decision problem* is one that returns an answer of "yes" or "no" for its solution. An algorithm is said to be *polynomially bounded* if there exists a polynomial function p such that for each input of size n, the algorithm terminates after at most $p(n)$ steps. A decision problem is said to be in the class $\mathcal{P}$ if it can be solved by a polynomially bounded algorithm. The class $\mathcal{NP}$ refers to decision problems whose solutions can be *verified* in time that is polynomial in the size of the input. A problem L_1 is said to be *polynomial-time reducible* to problem L_2 if there is a mapping f from the inputs of L_1 to the inputs of L_2 such that f can be computed in polynomial-time, and the answer to L_1 on input x is yes if and only if the answer to L_2 on input $f(x)$ is yes. A problem L is $\mathcal{NP}$-hard if for any problem $L' \in \mathcal{NP}$, L' is polynomial-time reducible to L. If problem L is $\mathcal{NP}$-hard and $L \in \mathcal{NP}$, then L is $\mathcal{NP}$-complete. Though $\mathcal{P} \subset \mathcal{NP}$, it is still an open question whether $\mathcal{P} = \mathcal{NP}$, which is equivalent to asking if there is some $\mathcal{NP}$-complete problem that can be solved by a polynomially bounded algorithm.

Most of the discrete optimization problems are $\mathcal{NP}$-hard or $\mathcal{NP}$-complete, even if they are linear problems. As an example, it is shown in [PS82, page 358] that the problem of determining if $\mathbb{Z} \cap \{Ax = b, x \geq 0\} \neq \emptyset$ is $\mathcal{NP}$-complete. Solving such optimization problems can be difficult because of the complexity involved. Since there is a finite set of possible solutions, one possibility is simply to examine all such solutions, i.e., perform an exhaustive enumeration of all the solutions. However, if we have m binary decision variables, we might have to perform up to 2^m function evaluations to determine the optimal solution by enumeration. So for a problem with a modest size of 1000 binary decision variables, it would require at least millions of years for a computer that can execute 10^{15} operations per second to find the optimal solution by enumeration. Though one can probably eliminate some enumeration possibilities by clever observation (such as the branch-and-bound method to be discussed below), even a radical reduction may still leave an untenable number of choices. Also, there are specialized algorithms to solve certain types of linear discrete optimization problems, like the airline crew scheduling problem. However, the algorithms are combinatorial in nature and also require a vast amount of computational effort for large problems.

For simplicity, we only consider problem (P1.8). We also assume that $D \subset \{0,1\}^n$. It is explained in the next section why making this assumption on D does not result in any loss of generality. Two general techniques for solving (P1.8) are branch-and-bound and cutting-plane methods.

1.2.2.1 Branch-and-Bound Methods

In this approach, the feasible region is systematically partitioned into subdomains and such a partitioning process can be represented by a tree with each node representing a subproblem. The simplest way to partition the feasible region is to consider the two subproblems when a particular variable $x_j = 0$ and $x_j = 1$ respectively. These subproblems generated by the partition are used to determine bounds on the objective function and also for updating the best objective value obtained so far. To be more specific, upper and lower bounds are being generated at different levels and nodes of the tree throughout the whole branch-and-bound process, until the upper and lower bounds differ by an acceptable tolerance.

First note that if a subset of the variables are allowed to be continuous, then the optimal objective of this subproblem will be a lower bound on the solution of the original discrete problem. Also, if no feasible solution exists for the relaxation of a subproblem, then no feasible solution exists for the subproblem itself. Once a feasible solution is known, this yields an upper bound to the required solution. These observations are used in a systematic way in the branch-and-bound method. In the event that the subproblems are infeasible, the trees using these subproblems as the root nodes will be discarded. Similarly, if the subproblems obtained are shown to have objective values or bounds that are not as good as the best known objective values or upper bounds, they are also discarded. Otherwise, one continues with further partitioning to obtain new but smaller subproblems for determination of new bounds or objective value. The whole process is repeated until all the possible partitions have been carried out and an optimal solution is obtained, or if the upper and lower bounds of all partitions considered fall within a prespecified tolerance. When picking the list of candidate subproblems to be considered, it is desirable to make the selection in such a way as to reduce the gap between the upper and lower bounds quickly. Thus, a branch-and-bound algorithm could be considered as an enumerative method where intelligent choices are made to reduce the amount of work required. For an early survey and discussion of branch-and-bound methods, see [LW66] and [Mit70]. Different branch-and-bound algorithms vary in the choice of variables for partitioning with the purpose of discarding more non-optimal subproblems at an early stage.

Like other enumerative methods, the method can be terminated after at least one feasible solution has been found. While this may not be provably optimal, it may nonetheless be of value, e.g., in providing bounds on the optimal objective values.

1.2.2.2 Cutting-Plane Methods

The basic idea behind cutting-plane methods is to add constraints to the problem so that if it is solved as a continuous problem, a discrete solution is obtained.

One first solves for the continuous relaxation problem $\min\{f(x) : Ax = b,\ 0 \leq x \leq e\}$ using the simplex method. If an $x' \in \{0,1\}^n$ is obtained, then we are done. If not, then an additional linear constraint is imposed on the region $\{Ax = b,\ 0 \leq x \leq e\}$ to prevent x' from being obtained as an optimal solution of the new problem, and yet not eliminating any feasible point in $\{0,1\}^n$. This new problem is also solved by the simplex method and the process repeated if necessary until an $x' \in \{0,1\}^n$ is obtained or it is concluded that the original problem is infeasible. As an example of a cut, suppose the simplex method is applied to the continuous relaxation problem $\min\{f(x) : Ax = b,\ 0 \leq x \leq e\}$, obtaining the optimal tableau

$$x_i = g_{i0} + \sum_{j \in N} g_{ij}(-x_j), i \in B, \tag{1.1}$$

where B and N are the basic and non-basic variables in the optimal tableau respectively. Assume that x_k is fractional for some $k \in B$. Define $N_1 = \{j \in N : f_{kj} < f_{k0}\}$, where f_{kj} represents the fractional part of g_{kj}, then one possible cut can be defined by

$$\sum_{j \in N_1} \min\left\{\frac{f_{kj}}{f_{k0}}, \frac{1-f_{kj}}{1-f_{k0}}\right\} x_j \geq 1. \tag{1.2}$$

The first cutting-plane method was developed by Gomory [Gom58] and the cut (1.2) is attributed to him. However, because of the slow convergence to integer solutions, pure cutting-plane algorithms are rarely practical. Typically, branch-and-bound algorithms are combined with the cutting-plane approach in which a small number of efficient cuts are added to the problems at the nodes of the branch-and-bound tree. Such methods are known as the branch-and-cut methods and a recent survey can be found in [Mit99].

1.3 General Discrete Optimization Problems

Although many discrete optimization problems are linear, there is also an abundance of practical problems that are in the form of (P1.1) with $f : \mathbb{R}^n \to \mathbb{R}$ nonlinear. As an example, consider the linear assignment problem discussed earlier. It may turn out that one needs to factor in nonlinear costs in the objective function, making the problem nonlinear. A well-known nonlinear discrete optimization problem is that of the quadratic assignment problem. A discussion of applications of nonlinear discrete optimization problems is given in Chapter 7.

It may not always be a disadvantage if we have to deal with nonlinear discrete optimization problems. This is because we can always transform such problems into other manageable nonlinear problems. While the same idea may be applied to linear discrete optimization problems, it comes at a possible loss of any advantages to the original problem being linear.

1.3.1 The Evaluation of $f(x)$

A requirement of the approach we advocate is that it be possible to evaluate $f(x)$ at non-integer values of x. For linear functions and many nonlinear functions, that is always true. However, there are functions for which it is not true. For example, suppose

$$f(x) = \sum_{i=0}^{x_{10}} c_i(x)$$

for some functions $c_i(x)$. If we assign a non-integer value to x_{10} this expression has no meaning. Instead, we define a function $\bar{f}(x)$ such that $\bar{f}(x) = f(x)$ when x is an integer. In the above case, we can define

$$\bar{f}(x) \equiv \sum_{i=0}^{\lfloor x_{10} \rfloor} c_i(x) + (x_{10} - \lfloor x_{10} \rfloor) c_{\lfloor x_{10} \rfloor + 1}.$$

While the above transformation results in $\bar{f}(x)$ being continuous, it lacks continuous differentiability, which are crucial to the methods we wish to apply. However, the transformation is satisfactory if $x_{10} \in [0, 1]$ and this emphasizes a reason for transforming the problem into one with binary variables only.

Integer variables may be used to control a choice. For example if $x_7 = 1$, then carry out decision A; if $x_7 = 2$, then carry out decision B and so on. Such statements can sometimes be replaced by additional constraints and the introduction of binary variables.

How to reformulate the problem with continuous variables may also be deduced by altering the physics of the model. For example, consider the distillation problem described in [GMW81]. Suppose $x_5 \in \{1, 2, \ldots, 6\}$ is the tray number of the input feed in the distillation column. We could consider a new model in which x_5 is replaced by a set of continuous variables $x_{5,i} = 1, 2, \ldots, 6$, where $x_{5,i}$ is the proportion of the feed going into tray i. Another example is the frequency assignment problem we discuss later. The standard model assumes a station transmitting on one frequency (from a limited set of frequencies) and the aim is to minimize interference due to stations using the same frequency. We could instead allow a station to broadcast on all frequencies with the variables being the percentage for a given frequency. In both cases we need to force the solution to comply with the real situation. However, at points other than the solution, there is a physical interpretation of the variables. Note that in both these cases, the number of continuous variables is considerably greater than the number of discrete variables.

1.3.2 Reformulation to Nonlinear Binary Problems

Though problems with only binary variables are the simplest form of discrete problems, they are very important because any discrete problem with bounded variables can always be transformed into a binary problem. More specifically, problems with constraints $x_i \in S_i$, where S_i is a finite set of integers, can always be transformed to an equivalent problem with binary variables.

Without a loss of generality, consider the example of an integer variable x bounded by 0 and u. We can then use the substitution

$$x = \sum_{i=0}^{k} 2^i u_i,$$

where $k = \lfloor \frac{\ln u}{\ln 2} \rfloor$ and u_i are new binary variables to be introduced. Thus, we effectively remove the integer variable x and replace it with $k + 1$ binary variables. This is probably the best way to introduce the minimal number of binary variables possible in place of integer variables with an upper and lower bound. See [LB66] for more details about such a transformation.

In the event that S_i is a finite set of increasing but not necessarily consecutive integers say $\{n_1, n_2, n_3, \ldots, n_{k_i}\}$, we may introduce undesired representations of x and extra binary variables by assuming that $n_1 \leq x \leq n_{k_i}$ and using the above approach, especially if n_{k_i} is very large. Instead, we can introduce k_i binary variables as in the knapsack problem and define them as follows:

$$y_j = \begin{cases} 1 & \text{if } x_i = n_j \\ 0 & \text{otherwise,} \end{cases}$$

for $j = 1, 2, \ldots, k_i$. We will also have to include the constraint $e^{\mathrm{T}} y = 1$. Such additional constraints do not necessarily make the problem harder to solve since they are of a special structure.

Although we can convert all the bounded integer variables in this way, it may be thought a disadvantage to introduce such a large number of additional variables. However, although the problems are larger, they contain a lot of structure. The basic data defining the problem has not increased. For example, the size of the Hessian of the objective function may increase by a factor of ten (hence the number of elements increases by a factor of hundred) but the number of nonzero elements is likely to remain constant. Consequently, the increase in size is irrelevant if sparse technology is used.

Another concern about such a transformation is that information might be lost. For example, x_i may represent the number of satellites used in interferometric images. The greater the number of satellites used, the better the image obtained but the greater the costs. If 7 is the optimal choice, it is likely that 6 and 8 are good choices compared to say 20. This information may be lost if we transform the problem by the following introduction of binary variables

$$y_j = \begin{cases} 1 & \text{if } x_i = j \\ 0 & \text{otherwise.} \end{cases}$$

For example, suppose the good solution with $y_6 = 1$ and $y_j = 0$ for $j \neq 6$ is obtained. In searching for an improvement, setting $y_{20} = 1$ and $y_j = 0$ for $j \neq 20$ will seem just as likely to improve the solution as setting $y_7 = 1$ and $y_j = 0$ for $j \neq 7$, or setting $y_5 = 1$ and $y_j = 0$ for $j \neq 5$. So, the information that reflects the better choices of consecutive numbers of satellites used is lost by such a transformation. However, there are problems in which there is no relevance to the order of the integers (such problems are likely to be harder to solve). For example, consider the frequency assignment problem in which one of 4 frequences has to be assigned to 20 stations. Suppose the optimal solution is to assign frequency 1 to the first 10 stations, frequency 2 to the 11th–14th stations, frequency 3 to the 15th–17th stations, and frequency 4 to the 18th–20th stations. There is no reason to suppose assigning frequency 3 to the 20th station is better than frequency 4. Thus, the optimal solution could well have been to assign frequency 1 to the first 10 stations, frequency 2 to the 11th–14th stations, frequency 4 to the 15th–17th stations, and frequency 3 to the 18th–20th stations, and there is no difference to distinguish between these two solutions, or any other solutions obtained by re-ordering the frequencies. Indeed in such problems, we can re-order the variables without impacting the problem. So for some problems, there is no loss of information by transforming them to one with binary variables, while for others, care may need to be exercised to avoid a loss of information.

Sometimes, it is also possible to handle the transformation to binary variables efficiently. Using the satellite example again, it may be that only a window of 5 binary variables, say y_4 to y_8 need be considered when searching for an improvement to a current solution of $y_6 = 1$ and $y_j = 0$ for $j \neq 6$. If the search produces a new solution of $y_7 = 1$ and $y_j = 0$ for $j \neq 7$, then we can consider the new window of binary variables y_5 to y_9. This will be more efficient than considering all the binary variables y_i for every i each time. Thus, if we had a special algorithm that deals with binary decision variables efficiently and care is exercised to avoid loss of information in transforming the original problem into one with binary variables, it is worthwhile performing the transformation. For simplicity of discussion, we only consider problems with binary variables subsequently, unless otherwise stated.

1.3.3 Techniques To Solve Nonlinear Discrete Problems

An obvious approach to solving nonlinear discrete problems is to generalize the two methods discussed for solving the linear discrete problem (see e.g., [GR85]). Note that both these approaches capitalize on the existence of fast algorithms to solve the continuous problem. We utilize the same idea. However, we do not generalize either the branch-and-bound or the cutting-plane algorithm. There is an inherent difficulty in generalizing the branch-and-bound (and hence the branch-and-cut) method because it critically depends on the uniqueness of the solution. For convex problems, this would not be an issue but we are interested in developing algorithms for nonconvex problems.

The degree of difficulty introduced by having a nonlinear objective may be gauged from the following problem:

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & 0 \leq x \leq e \\ & x \in \{0,1\}^n. \end{array} \tag{P1.9}$$

When $f(x)$ is linear, cutting planes are unnecessary because the bound constraints define all possible integer solutions. If the problem is solved as a continuous one (i.e., dropping the integrality constraints), it is trivial to ensure that an integral solution is obtained. Indeed, the appropriate vertex of the feasible region may be found by examining the coefficients of the objective. When $f(x)$ is nonlinear, the problem is nontrivial. Solving the problem as a continuous one no longer assures an integer solution. The very rationale behind a pure cutting plane method is therefore no longer valid. However, the idea of using cutting planes within other algorithms is still valid. What this example illustrates is the difficulty of obtaining a discrete solution when solving a continuous problem with a nonlinear objective function $f(x)$, and also the inherent limitations of generalizing the techniques for linear problems to nonlinear problems.

Like the linear problem, there are also specialized algorithms to deal with certain types of nonlinear discrete problems. As an example, there are many exact algorithms to solve the quadratic assignment problem. However, it is hard or impossible to generalize such algorithms. Below is a discussion of some of the methods that have been proposed to handle more general nonlinear discrete problems, instead of special types of nonlinear discrete problems.

1.3.3.1 Decomposition Methods

One approach to solving problems with a mixture of discrete and continuous variables is to use decomposition methods. However, such methods would need to make use of available methods for solving pure integer or mixed-integer linear optimization problems, as discussed in Section 1.2.2. As an example, the Generalized Benders

Decomposition method [Geo72] decomposes a mixed-integer nonlinear programming problem into two problems that are solved iteratively – a pure integer linear master problem and a nonlinear continuous subproblem. The nonlinear subproblem is obtained by fixing the integer values and it optimizes the continuous variables to give an upper bound to the original problem. On the other hand, the master problem optimizes for the new integer variables by imposing new constraints, such as the Lagrangian dual formulation of the nonlinear problem. This master problem will yield additional combinations of the integer variables for the subsequent nonlinear subproblems, as well as estimate lower bounds to the original problem. Under convexity assumptions, the master problems generate a sequence of lower bounds that is monotonically increasing. The iterations terminate when the difference between the upper bound and lower bound is smaller than a prespecified tolerance.

Another example of the decomposition approach is the Outer Approximation Method [DG86], which has been implemented as the DICOPT solver in GAMS. It is similar to the Generalized Benders Decomposition Method in that it also involves solving alternately a master problem and a continuous nonlinear subproblem. The main difference lies in the setup of the master problem. In outer approximation, the master problems are generated by linearizations of the nonlinear constraints (using Taylor series) at those points that are the optimal solutions of the nonlinear subproblems, and so they are mixed-integer programming problems. Again, to ensure global optimality or finite termination, some convexity assumptions are required.

There is no assurance a decomposition method will obtain a reasonable solution to a nonconvex problem. Moreover, they need to solve master problems that increase in the number of constraints as the iterations proceed. Since integer variables are involved, the cost of solving the master problem may become prohibitive.

1.3.3.2 Branch-and-Reduce Methods

This class of methods also handles (P1.1) problems and it is of the branch-and-bound type discussed earlier, i.e., it requires the construction of a relaxation of the original problem that can be solved to optimality to produce a lower bound for the original problem. Usually, the relaxation is constructed by enlarging the feasible region or using an underestimation of the objective function. The approach also includes range contraction techniques like interval analysis and duality theory that systematically reduce the feasible region to be considered, and incorporates branching schemes that guarantee finite termination with the global optimal solution for certain types of problems.

At each iteration, the search domain is partitioned and both upper and lower bounds are obtained for each partition. The partitioning process continues until the upper and lower bounds over all partitions differ by a prespecified tolerance. As in branch-and-bound methods, the partitions that produce infeasible regions or regions with poor objective values are discarded.

A more detailed description of the method can be found in [TS99]. The algorithm has also been developed as a general-purpose global optimization system called the Branch and Reduce Optimization Navigator (BARON) with modules that handle different classes of problems. The manual for BARON can be found in [Sah00].

Like the branch-and-bound method, this algorithm may have the pitfall of going through an unpredictably large number of iterations even though it has good branching schemes. This may pose a heavy computational burden because of the need to solve the correspondingly large number of nonlinear relaxation problems. Moreover, the construction of the relaxation problem may involve a convex underestimation of the objective function that is highly inefficient in generating bounds.

1.4 Outline Of Remaining Chapters

The aim of this thesis is to find a generic approach to handling nonlinear discrete optimization problems. The continuation approach that is adopted is described in Chapter 2. The proposed algorithms and an analysis of their convergence are discussed in Chapter 3. Chapter 4 begins with an analysis of linear systems with large diagonal elements, while Chapter 5 examines the implementation aspects of the algorithm. Selected applications of the problems are discussed in Chapter 6, together with the numerical results and comparison with methods described in Section 1.3.3 to show the practical performance of the continuation approach. Chapter 7 discusses methods for extending the algorithm to solve more general classes of nonlinear discrete optimization problems, before concluding with suggested future work.

Chapter 2

A Continuation Approach

Continuation methods arose as a way to solving systems of nonlinear equations [Dav53, Was73]. The aim is to solve a difficult system of equations $F(x) = 0$ by first solving a simpler system of equations $G(x) = 0$. Here we assume that $F, G \in C^2$ and $x \in \mathbb{R}^n$. In continuation methods, the procedure is to find the roots of a new function $H : \mathbb{R}^n \times [0, 1] \to \mathbb{R}$, defined by

$$H(x, \lambda) = \lambda F(x) + (1 - \lambda)G(x). \tag{2.1}$$

This function has the obvious properties of $H(x, 0) = G(x)$ and $H(x, 1) = F(x)$. The basic idea is to solve a sequence of problems, $H(x, \lambda) = 0$, for $\lambda = \lambda_0 < \lambda_1 < \lambda_2 < \cdots < \lambda_t = 1$. Assuming that roots of $G(x)$ are easy to find, it should be easy to find an approximate root x_0 of H for some initial value λ_0. Given each starting point x_k that is an approximate root to the equation $H(x, \lambda_k) = 0$, one solves for an approximate root x_{k+1} to the next equation $H(x, \lambda_{k+1}) = 0$, using an iterative method such as Newton's method. The continuation process stops when one reaches $\lambda = 1$ with the root $\bar{x}$ that satisfies $F(\bar{x}) = H(\bar{x}, 1) = 0$. The hope is to find a path $\mathcal{P}$ parametrized by λ that begins with x_0 and ends with the desired $\bar{x}$, i.e., a trajectory that can be described by $\{x(\lambda) : \lambda \in [0, 1]\}$ with $x(\lambda_0) = x_0$ and $x(1) = \bar{x}$. This path is sometimes known as the *zero curve* as it passes through the roots of $H(x, \lambda) = 0$ as λ varies from 0 to 1. The existence of such paths can often be justified using the

Implicit Function Theorem when certain assumptions are satisfied.

Theorem 2.1 (Implicit Function Theorem). *Let $f : X \to \mathbb{R}^m$ be a continuously differentiable function, where $X \subset \mathbb{R}^{m+1}$ is open. If $(a, b) \in X$ is such that $f(a, b) = 0$ and rank of $f'(a, b)$ is m, then there exists a neighborhood N of (a, b) and a unique continuously differentiable function $g : N \to \mathbb{R}^m$ satisfying the conditions $g(a) = b$ and $f(x, g(x)) = 0$ for all $x \in N$.*

This theorem means that under the assumptions stated in the theorem, there will be a neighborhood of (a, b) where a unique zero curve of f is defined passing through (a, b). Although this is only a theorem describing the local behavior of (a, b), we can certainly apply the theorem to new points on the zero curve in order to extend it.

The addition of the function $G(x)$ serves two purposes: it makes the combined function better behaved and it ensures that there is a solution close to the initial point. We can for example define $G(x) = x - x_0$, where x_0 is an initial point. In this way, the combined function would tend to behave initially like a linear function with one root in a small neighborhood of x_0 when λ is sufficiently small. Many iterative methods like Newton's method converge quickly from a good initial point (one close to the solution) but may converge slowly from a poor one. Thus, it is a virtue of continuation methods to allow the choice of $G(x)$ to control the path taken by the iterates.

Despite the many advantages of continuation methods, numerical problems may be encountered when $\mathcal{P}$ has turning points (i.e., $\dot{x} = 0$), or bifurcation points (i.e., rank of $H' < n$), or if $\mathcal{P}$ stops before λ reaches 1, or if $\mathcal{P}$ is unbounded (see Figure 2.1). A term used to describe continuation methods that overcome the impact of turning or bifurcation points by allowing λ to both increase and decrease along $\mathcal{P}$ is *homotopy methods*.

Definition 2.1. A *homotopy* is a continuous map from $[0, 1]$ into a function space.

It can be easily verified that (2.1) is an example of a homotopy when F and G are bounded in the function space containing them. There is also a class of homotopies called probability-one homotopies [CMY78] known to be globally convergent, i.e., the

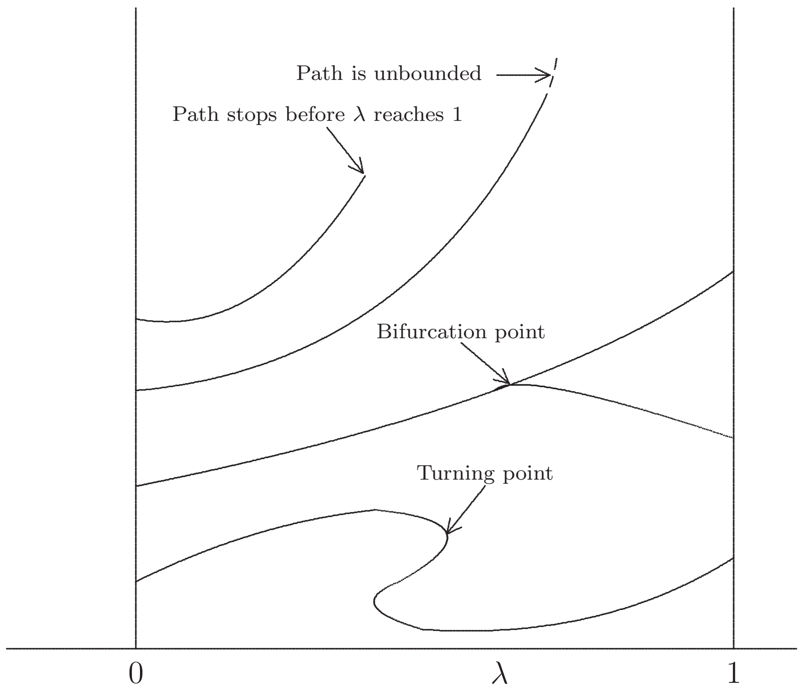

Figure 2.1: Continuation paths running into difficulties.

zero curve $\mathcal{P}$ reaches a solution $\bar{x}$ where $H(\bar{x}, 1) = 0$ from any arbitrary starting point x_0 such that $H(x_0, 0) = 0$. A discussion of these numerical continuation/homotopy methods can be found in [AG90].

Before discussing how continuation methods can be applied to optimization, we need to review the Karush-Kuhn-Tucker (KKT) conditions.

2.1 The KKT System

Many algorithms for solving smooth optimization problems involve finding points that satisfy the KKT conditions, which are first-order necessary conditions of optimality, when a certain constraint qualification holds. As an example, consider the nonlinear programming problem

$$\begin{aligned} &\text{Minimize} && f(x) \\ &\text{subject to} && g(x) \leq 0, \end{aligned} \tag{P2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, and $f, g \in C^2$. The KKT conditions can be written as

$$\begin{aligned} \nabla f(x) + \nabla g(x)^{\mathrm{T}} u &= 0 \\ g(x) &\leq 0 \\ u &\geq 0 \\ g(x)^{\mathrm{T}} u &= 0, \end{aligned} \tag{2.2}$$

where $u \in \mathbb{R}^m$.

The full statement of the theorem involving the KKT conditions for optimality requires certain constraint qualification assumptions to be satisfied. Two constraint qualifications for problem (P2.1) are defined below.

Definition 2.2. The *linear independence constraint qualification* (LICQ) is said to be satisfied for g at a solution $\bar{x}$ of (P2.1) if the vectors $\nabla g_i(\bar{x})$ for $i \in \mathcal{A}$ are linearly independent, where $\mathcal{A} = \{i : g_i(\bar{x}) = 0\}$.

Definition 2.3. The *Arrow-Hurwicz-Uzawa constraint qualification* (AHUCQ) is said to be satisfied for g at a solution $\bar{x}$ of (P2.1) if the inequalities

$$\begin{aligned} J_W(\bar{x})z &> 0 \\ J_V(\bar{x})z &\geq 0 \end{aligned}$$

have a solution $z \in \mathbb{R}^n$, where $W = \{i : g_i(\bar{x}) = 0,\ g_i \text{ is concave at } \bar{x}\}$, $V = \{i : g_i(\bar{x}) = 0,\ g_i \text{ is not concave at } \bar{x}\}$ and $J_W(\bar{x})$, $J_W(\bar{x})$ are matrices denoting the rows of the Jacobian of g at $\bar{x}$ with respect to W and V respectively.

The KKT Theorem for the first-order necessary conditions of optimality can now be stated.

Theorem 2.2. *Let X be an open set in $\mathbb{R}^n$ and $\mathcal{F} = \{x \in X : g(x) \leq 0\}$ be non-empty. Let $\bar{x}$ be a local minimizer of $\min_{x \in \mathcal{F}} f(x)$ and suppose either LICQ or AHUCQ is satisfied at $\bar{x}$. Then there exists a $\bar{u} \in \mathbb{R}^m$ such that $(\bar{x}, \bar{u})$ solves (2.2).*

Under certain convexity assumptions, the necessary conditions for optimality are also the sufficient conditions for optimality. There are also similar second-order necessary and sufficient conditions for optimality. See [Man69, BSS93] for more details and proofs of the conditions for optimality.

2.2 Continuation Methods in Optimization

In general, a KKT system, i.e., the system of equations and inequalities arising out of the KKT conditions, may be difficult to solve. But it is possible to rewrite such a system as an equivalent nonlinear system of equations. For example, the complementarity conditions $g(x) \leq 0,\ u \geq 0,\ u^{\mathrm{T}} g(x) = 0$ in (2.2) can be replaced by the equivalent system of equations [Man76]

$$C(x, u) = 0,$$

where $C_i(x, u) = -|u_i + g_i(x)|^3 + u_i^3 - (g_i(x))^3$ for $i \in \{1, 2, \ldots, l\}$.

The resulting equations or other equations arising out of the KKT system may be highly nonlinear. However, continuation methods are well-suited for handling such difficult systems of equations, and even the need to have a good starting solution to part of the KKT system is not critical. Moreover, the sparsity of the system can be preserved when continuation methods are applied to solve the system.

To make use of the above approach, Watson [Wat00] suggested applying continuation methods to the first-order necessary conditions for optimality, which are sufficient for convex functions. To that end, he proved the following two theorems. Note that his approach is closely related to the proximal-point method, where one solves the problem $\min_{x \in \mathbb{R}^n} F(x)$ by solving $P_k : \min_{x \in \mathbb{R}^n} F(x) + \gamma_k (x - x_k)^{\mathrm{T}} (x - x_k)$ iteratively with x_k being the solution to problem $P_{k-1}, k \geq 1$ and $x_0 = a$.

Theorem 2.3 ([Wat00]). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a C^3 convex function with a minimizer* *at $\tilde{x}$ and $M \in \mathbb{R}$ such that $\|\tilde{x}\| \leq M$. Then for almost all a, $\|a\| < M$, there is a zero curve $\mathcal{P}$ of the homotopy map*

$$H_a(x, \lambda) = \lambda \nabla f(x) + (1 - \lambda)(x - a),$$

along which the Jacobian matrix $DH_a(x, \lambda)$ has full rank, starting from $(x_0, \lambda_0) = (0, a)$ and having an accumulation point $(\bar{x}, 1)$, where $\bar{x}$ solves

$$\underset{x \in \mathbb{R}^n}{\text{Minimize}}\, f(x). \tag{P2.2}$$

If the Hessian matrix $\nabla^2 f(\bar{x})$ is nonsingular, then $\mathcal{P}$ has finite arc length.

Theorem 2.4 ([Wat00]). *Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ be C^3 convex functions with g satisfying the Arrow-Hurwicz-Uzawa constraint qualification at every solution of the problem $\min\{f(x) : g(x) \leq 0\}$, and assume that the the feasible region $\{x : g(x) \leq 0\}$ is nonempty and bounded. Let $a = (x^0, b^0, c^0)$, $H_a(x, u, \lambda) = H(x^0, b^0, c^0, x, u, \lambda)$ be defined by*

$$H(x^0, b^0, c^0, x, u, \lambda) = \begin{pmatrix} \lambda \left[\nabla f(x) + \nabla g(x)^{\mathrm{T}} u\right] + (1 - \lambda)(x - x^0) \\ K(x, u, \lambda, b^0, c^0) \end{pmatrix},$$

where

$$\begin{aligned} K_i(x, u, \lambda, b^0, c^0) = -|(1 - \lambda)b_i^0 - g_i(x) - u_i|^3 + \big((1 - \lambda)b_i^0 - g_i(x)\big)^3 \\ + u_i^3 - (1 - \lambda)c_i^0, \quad i = 1, \ldots, m, \end{aligned}$$

and u^0 be uniquely defined by the system $K(x^0, u^0, 0, b^0, c^0) = 0$. Then for almost all $x^0 \in \mathbb{R}^n$, almost all $b^0 \in \mathbb{R}^m$ such that $b^0 > 0$ and $g(x^0) - b^0 < 0$, and almost all $c^0 \in \mathbb{R}^m$ with $c^0 > 0$, there exists a zero curve $\mathcal{P}$ of $H_a(x, u, \lambda)$ starting from $(x^0, u^0, 0)$, along which the Jacobian matrix $DH_a(x, u, \lambda)$ has rank $n + m$, reaching a point $(\bar{x}, \bar{u}, 1)$, where $\bar{x}$ solves the problem $\min\{f(x) : g(x) \leq 0\}$. If the rank of $DH_a(\bar{x}, \bar{u}, \lambda)$ is $n + m$, then $\mathcal{P}$ has finite arc length.

Though Watson has shown similar results for nonconvex programming problems, the additional assumptions required are restrictive. If we are interested in satisfying second-order as well as first-order optimality conditions, the situation is considerably more complex. There is also a major objection to solving the KKT system directly, namely we could get a local maximum or saddle point instead. Even without attempts to satisfy second-order conditions directly, methods that “minimize” favor better stationary points.

Other examples of applying continuation methods to certain optimization problems such as complementarity problems can be found in [Kan97, KMM93, KMN91, ZL01]. In [SS92], there is also a description of applying continuation methods to a global optimization problem, with the aim of finding a trajectory $\mathcal{P}$ that would pass through all stationary points of the problem.

2.3 Smoothing Methods

In general, the presence of several local minima in an optimization problem makes the search for the global minimum very difficult. The fewer the local minima, the more likely it is that an algorithm will find the global minimum. Thus, in the choice of continuation methods applied to such problems, we would like to transform the original problem with many local minima into one that has fewer local minima or even just one local mimimum, obtaining a global optimization problem that is easier to solve (see Figure 2.2). Obviously we wish to eliminate poor local minima. Consequently, our use of continuation methods is not only to make the initial problem easier to solve (because the initial point is better and/or the initial function is better behaved) but we also wish to reduce the number of solutions. We may then solve the original problem by solving a sequence of problems in the same spirit as continuation and homotopy methods. We term such continuation methods *smoothing methods* and they can be categorized into two types: *local* or *global* smoothing.

In the subsequent discussion, we consider functions of the form

$$F(x, \mu) = f(x) + \mu g(x),$$

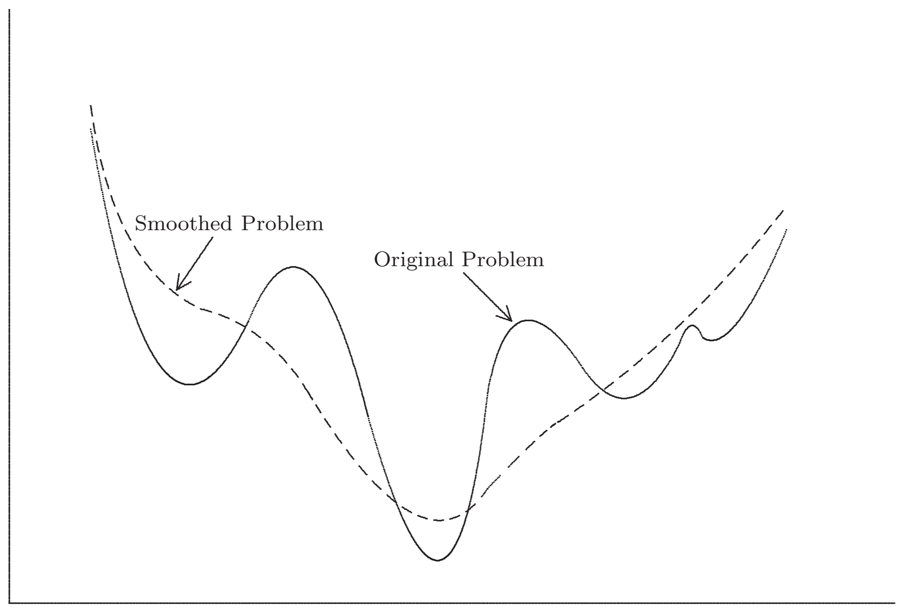

Figure 2.2: Effect of smoothing the original optimization problem.

where f and g are real-valued functions on $\mathbb{R}^n$ and $\mu \geq 0$. It may be observed that the mapping $H : [0,1] \to C^2$, where $H(\mu)$ is the function such that

$$[H(\mu)](x) = \frac{1}{1+\mu} F(x,\mu), \tag{2.3}$$

is a homotopy as defined earlier.

2.3.1 Local Smoothing

Some approaches address specific categories of problem, such as the objective function being noisy or having a local variation that creates many local minimizers. Noise in the evaluation of a function arises in several ways. One common cause is that to evaluate the objective function requires the solution of a complex numerical problem such as the solution of a partial differential equation, or involves a statistical process that is iterative in nature. In such cases evaluating the objective accurately is either expensive or impractical. When a simulation is required to obtain the objective values, then provided sufficient trials are performed, the objective would not be noisy but the number of such trials might be astronomically large. One approach to solving "noisy" problems is to smooth the local variation. This can sometimes be done directly by modifying the model. For example, one approach to finding edges in images is to minimize a potential function (see [Sch01]). In such problems the original pixel images can be replaced by a sequence of "smoothed" images by some suitable mapping of the pixels. The same idea may be applied generically, for example, using a convolution in which the function is replaced by a function call with a multiple integral (see [MW97]):

$$F(x,\mu) = \frac{1}{\pi^{n/2}} \int_{\mathbb{R}^n} f(x+\mu u) e^{-\|u\|^2} du.$$

The obvious aim is to remove poor local minimizers and retain good ones. This and similar approaches are not simply trying to remove tiny local variations, but in some instances can even be used to remove more significant but nonetheless poor local minimizers. It might be useful to define formally what we mean by a "significant" minimizer. Consider a local minimizer (or stationary point) $\bar{x}$. Let S be the set of points such that if $x \in S$, then there exists a solution to

$$\dot{x}(t) = -\nabla f(x(t)),$$

such that $x(0) \in S$ and $\lim_{t\to\infty} x(t) = \bar{x}$ and the curve $\bar{x}(t)$ is continuous. Let V_S be the volume of S. The "significance" of a minimizer is a function of the size of V_S. In a bounded region, V_S is related to the probability that a random initial point will converge to $\bar{x}$.

Such approaches usually have a parameter that can be adjusted to vary the degree of smoothing, the basic idea being first to minimize (perhaps approximately) a very smooth function and then to use that solution as an initial point for a function not quite so smooth. The use of the word "smooth" for a function f in this context means it has few roots for the equation $\nabla f = 0$. The hope is that for large values of the smoothing parameter only the best or better minimizers are preserved.

It is not difficult to show that the approach advocated in [MW97] and similar approaches can fail to find the global minimizer even on simple one-dimensional functions. For example, consider the function

$$f(x) = x^2(x+2)^2(x-2-0.3)(x-2+0.3).$$

When the initial smoothing parameter is chosen to make the function unimodal, it can be shown that the minimizer found for the original problem of minimizing f is independent of the choice of initial point. Unfortunately, the minimizer found is not the global minimizer. Even if the initial point is the global minimizer, the method will still find a local minimizer for the original problem. Figure 2.3 shows the original function and the smoothed function for three values of the smoothing parameter. It can be seen from the figures that regardless of the choice of initial point, the middle minimizer is always found. It is instructive to see why the approach fails since intuitively the global minimizer ought not to be destroyed by the process. The difficulty arises for functions with the property that

$$\lim_{\alpha\to\infty} f(x+\alpha p) = \infty$$

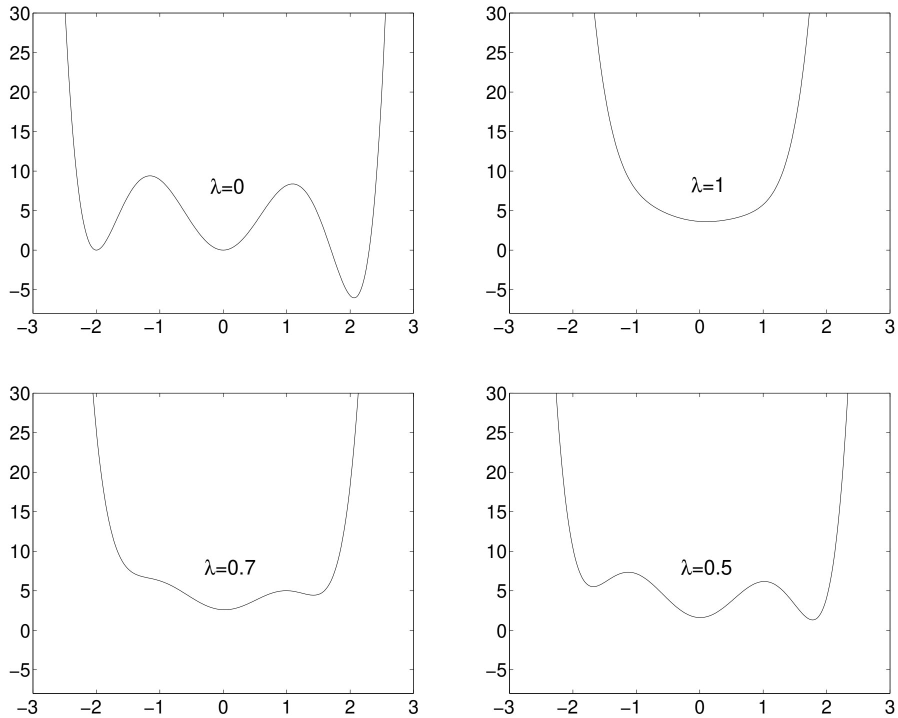

Figure 2.3: The impact of local smoothing as the parameter μ, which controls the degree of smoothing, is adjusted.

for some direction p that passes through a global (or good) minimizer, after which the function is monotonically increasing. The large values of $f(x)$ on one side of the minimizer remove the good minimizer in favor of minimizers that are more "interior". In other words, minimizers on the edge of the space in which minimizers lie are more adversely impacted by smoothing than those that are interior.

Another limitation of this type of approach arises from the dimension in which the smoothing is required being equal to the number of optimization variables. This may result in the computation of the smoothed function being expensive. In the case of the image problem the smoothing is applied only in two dimensions and is independent of the number of optimization variables. Moreover, the smoothing is done a small number of times and is not dependent on the number of iterations required by the optimization algorithm. It is often the case that the objective function is composed of several functions, only one of which is noisy. Morever, the function that is noisy might only have a range that lies in a subspace of the real line. For such problems, it is better to address the issue at the modeling level rather than within an algorithm.

In summary, local smoothing may be useful in eliminating tiny local minimizers even when there are large numbers of them. They are less useful if they also remove significant minimizers. Furthermore, the method can only be applied if the objective has favorable properties that allow efficient computation of the smoothed function.

2.3.2 Global Smoothing

The basic idea of global smoothing is to add a strictly convex function to the original objective, i.e.,

$$\mathcal{F}(x,\mu) = f(x) + \mu\Phi(x),$$

where Φ is a strictly convex function. If Φ is chosen to have a Hessian that is *sufficiently positive definite* for all x, i.e., the eigenvalues of this Hessian are uniformly bounded away from zero, it implies that for μ large enough, $\mathcal{F}(x,\mu)$ is strictly convex. For completeness, a proof of this assertion is included below, and similar results can be found, for example, in [Ber95, Lemma 3.2.1].

Theorem 2.5. *Suppose $f : [0,1]^n \to \mathbb{R}$ is a C^2 function and $\Phi : X \to \mathbb{R}$ is a C^2 function such that the minimum eigenvalue of $\nabla^2\Phi(x)$ is greater than ϵ for all $x \in X$, where $X \subset [0,1]^n$ and ϵ is a positive number. Then there exists a real $M > 0$ such that if $\mu > M$, then $f + \mu\Phi$ is a strictly convex function on X.*

Proof. Let $\{\lambda_i(H(x)) : i = 1,2,\dots,n\}$ denote the set of eigenvalues of a matrix function $H(x)$. Since f is a C^2 function, $\nabla^2 f(x)$ is continuous function of x and hence, its eigenvalues $\lambda_i(\nabla^2 f(x))$ are also continuous functions of x for all i. As $[0,1]^n$ is a compact subset of $\mathbb{R}^n$, $\lambda_i(\nabla^2 f(x))$ is bounded on $[0,1]^n$ for all i, i.e., there exists $L > 0$ such that

$$|\lambda_i(\nabla^2 f(x))| \le L$$

for all i and $x \in [0,1]^n$. Thus, for any $d \in \mathbb{R}^n$ such that $\|d\| = 1$,

$$|d^{\mathrm{T}} \nabla^2 f(x) d| \leq L \tag{2.4}$$

for all $x \in [0,1]^n$. The Hessian of $f + \mu\Phi$ is $\nabla^2 f(x) + \mu \nabla^2 \Phi(x)$, $x \in X$. Define M to be L/ϵ. If $\mu > M$, then for any $d \in \mathbb{R}^n$ such that $\|d\| = 1$,

$$\begin{aligned} d^{\mathrm{T}}(\nabla^2 f(x) + \mu\Phi(x))d &= d^{\mathrm{T}} \nabla^2 f(x) d + \mu d^{\mathrm{T}} \nabla^2 \Phi(x) d \\ &\geq -L + \mu \lambda_{\min}(\nabla^2 \Phi(x)) \|d\|^2 \quad \text{(by (2.4))} \\ &\geq -L + \mu\epsilon \\ &> -L + \frac{L}{\epsilon}\epsilon = 0. \end{aligned}$$

This implies that the Hessian of $f + \mu\Phi$ is positive definite for all $x \in X$ and hence $f + \mu\Phi$ is strictly convex on X. □

Consequently, for μ sufficiently large, any local minimizer of $\mathcal{F}(x, \mu)$ is also the unique global minimizer. Typically the minimizer of $\Phi(x)$ is known or is easy to find and hence minimizing $\mathcal{F}(x, \mu)$ for large μ is also easy. As in continuation methods, the basic idea is to solve the problem for a decreasing sequence of μ starting with a large value and ending with one close to zero. The solution $x(\mu_k)$ of $\min_x \mathcal{F}(x, \mu_k)$ is used as the starting point of $\min_x \mathcal{F}(x, \mu_{k+1})$.

The idea behind global smoothing is similar to that of local smoothing, namely, the hope is that by adding $\mu\Phi(x)$ to $f(x)$, poor local minimizers will be eliminated. There are, however, important differences between global and local smoothing. A key one is that local smoothing does not guarantee that the function is unimodal for a sufficiently large value of the smoothing parameter (although it can sometimes be the case). The algorithm in [MW97] gives an example; we see that if the algorithm is applied to the function $\cos(x)$, one will get a multiple of $\cos(x)$. Hence, the number of minimizers of the smoothed function has not been reduced.

It is easy to appreciate that the global smoothing approach is largely independent of the initial estimate of a solution, since if the initial function is unimodal, the choice of initial point is irrelevant to the minimizer found. When μ is decreased and the subsequent functions have several minimizers, the old solution is used as the initial point. Consequently, which minimizer is found is predetermined. Independence of the choice of initial point may be viewed as both a strength and a weakness. What is happening is that any initial point is being replaced by a point close to the minimizer of $\Phi(x)$. An obvious concern is that convergence will then be to the minimizer closest to the minimizer of $\Phi(x)$. The key to the success of this approach is to choose $\Phi(x)$ to have a minimizer that is not close to any of the minimizers of $f(x)$. This may not seem to be an easy task, but it does have a solution for constrained problems. If it is known that the minimizers are on the edge of the feasible region (e.g., with concave objective functions), then the "center of the feasible region" may be viewed as being removed from all of them. We show later that global optimization problems arising from the transformation of discrete problems have this characteristic.

Chapter 3

Smoothing Algorithms

In this chapter, we consider two global smoothing algorithms for nonlinear discrete optimization problems. To simplify the discussion, we consider the nonlinear binary optimization problem

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & Ax = b \\ & x \in \{0,1\}^n \end{array} \tag{P3.1}$$

and its relaxation

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & Ax = b \\ & 0 \le x \le e, \end{array} \tag{P3.2}$$

where $A \in \mathbb{R}^{m \times n}$ and $x \in \mathbb{R}^n$.

3.1 Logarithmic Smoothing

The smoothing function, $\Phi(x)$, is defined to be

$$\Phi(x) \equiv -\sum_{j=1}^{n} \ln x_j - \sum_{j=1}^{n} \ln(1 - x_j). \tag{3.1}$$

This function is clearly well-defined when $0 < x < e$. If any value of x_j is 0 or 1, we have $\Phi(x) = \infty$, which implies we can dispense with the bounds on x to get the following transformed problem:

$$\begin{aligned} &\text{Minimize} \quad f(x) - \mu \sum_{j=1}^{n} [\ln x_j + \ln(1 - x_j)] \\ &\text{subject to} \quad Ax = b, \end{aligned} \tag{P3.3}$$

where $\mu > 0$ is the smoothing parameter. When a linesearch algorithm starts with an initial point $0 < x_0 < e$, then all iterates generated by the linesearch also satisfy this property, provided care is taken in the linesearch to ensure that the maximum step taken is within the bounds $0 < x < e$.

The function $\Phi(x)$ is a *logarithmic barrier* function and is used with barrier methods (see [FM68]) to eliminate inequality constraints from a problem. In fact, (P3.3) is sometimes known as the barrier subproblem for (P3.2). Our use of this barrier function is not to eliminate the constraints but because a barrier function appears to be an ideal smoothing function. Elimination of the inequality constraints is a useful bonus. It also enables us to draw upon the extensive theoretical and practical results concerning barrier methods.

A key property of the barrier term is that for $x > 0$, $\Phi(x)$ is strictly convex. If μ is large enough, the function $f + \mu\Phi$ will also be strictly convex.

Lemma 3.1. *If $f : [0,1]^n \to \mathbb{R}$ is a C^2 function and Φ is as defined in (3.1), then there exists a real $M > 0$ such that if $\mu \geq M$, then $f + \mu\Phi$ is a strictly convex function on $(0,1)^n$.*

Proof. Let $X = (0,1)^n$ and $\epsilon = 8$. Observe that the Hessian of $\Phi(x)$ for $x \in X$ is a diagonal matrix with the jth diagonal entry being $\frac{1}{x_j^2} + \frac{1}{(1-x_j)^2}$. This function has a minimum at $x_j = \frac{1}{2}$, which implies that every diagonal entry of $\nabla^2\Phi(x)$ is at least 8. Thus $\lambda_{\min}(\nabla^2\Phi(x)) \geq \epsilon$ and the desired result follows from Theorem 2.5. $\square$

Corollary 3.1. *Suppose the assumptions in Lemma 3.1 hold and that the set $\{x : Ax = b\} \cap (0,1)^n$ is non-empty. Then problem (P3.3) has a solution $x^*(\mu) \in (0,1)^n$.*

Also, there exists an $M > 0$ such that for all $\mu > M$, the solution to problem (P3.3) is unique.

Proof. First we show the existence of $x^*(\mu)$ for any $\mu > 0$. Define $X = [\frac{1}{4}, \frac{3}{4}]^n \cap \{x : Ax = b\}$. Since X is a compact set and $f + \mu\Phi$ is a continuous function on X, where Φ is as defined in (3.1), there exist real numbers L and U such that $L \leq f(x) + \mu\Phi(x) \leq U$ for all $x \in X$. Since $f(x) + \mu\Phi(x) \to \infty$ as $x_j \to 0^+$ or $x_j \to 1^-$, there exists an $\epsilon > 0$ such that for all $x \in ((0, \epsilon] \cup [1-\epsilon, 1))^n \cap \{x : Ax = b\}$,

$$f(x) + \mu\Phi(x) > U. \tag{3.2}$$

Also, ϵ must be $< \frac{1}{4}$. Define $X_1 = [\epsilon, 1-\epsilon]^n \cap \{x : Ax = b\}$. Again by continuity of $f + \mu\Phi$ on the compact set X_1, there exists $z \in X_1$ such that $f(z) + \mu\Phi(z) \leq f(x) + \mu\Phi(x)$ for all $x \in X_1$. Moreover, $f(z) + \mu\Phi(z) \leq U$ as $X \subset X_1$. By (3.2), $f(z) + \mu\Phi(z) < f(x) + \mu\Phi(x)$ for all $x \in (0, 1)^n \backslash X_1$. Thus, z is the required $x^*(\mu)$.

The uniqueness of $x^*(\mu)$ for sufficiently large μ follows from Lemma 3.1 and the convexity of the feasible region of (P3.3). □

Consequently, (P3.3) always has a unique solution for a sufficiently large value of μ, regardless of the nonconvexity of $f(x)$. In fact, by Theorem 8 of [FM68], under the assumptions already imposed on (P3.2), if $x^*(\mu)$ is a solution to (P3.3), then there exists a solution x^* to (P3.2) such that $\lim_{\mu \searrow 0} x^*(\mu) = x^*$. Moreover, $x^*(\mu)$ is a continuously differentiable curve. The general procedure of the barrier-function method is to solve the problem (P3.3) approximately for a sequence of decreasing values of μ. Note that if $x^* \in \{0, 1\}^n$, then we need not solve (P3.3) for μ very small because the rounded solution for a modestly small value of μ should be adequate. In fact, it would be sufficient to obtain a solution $\bar{x}(\mu)$ such that $\|\bar{x}(\mu) - x^*(\mu)\| = \mathcal{O}(\mu)$.

3.1.1 Penalty Terms

In the case that fractional solutions are obtained for problem (P3.3) with no clear-cut rounding, i.e., the variables are not close to zero or one, extra penalty terms can be added to the objective to ensure a 0–1 solution. One way of doing so is to introduce the term

$$\sum_{j \in J} x_j(1 - x_j), \tag{3.3}$$

with a penalty parameter $\gamma > 0$, where J is the index set of the variables that are judged to require forcing to a bound. The problem then becomes

$$\begin{array}{rl} \text{Minimize} & F(x) \triangleq f(x) - \mu \sum_{j=1}^{n} [\ln x_j + \ln(1 - x_j)] + \gamma \sum_{j \in J} x_j(1 - x_j) \\ \text{subject to} & Ax = b. \end{array} \tag{P3.4}$$

In general, it is possible to show that under suitable assumptions, the penalty function introduced this way is "exact" in the sense that the following two problems have the same minimizers for a sufficiently large value of the penalty parameter γ:

$$\begin{array}{rl} \text{Minimize} & g(x) \\ \text{subject to} & Ax = b \\ & x \in \{0, 1\}^n \end{array} \tag{P3.5}$$

and

$$\begin{array}{rl} \text{Minimize} & g(x) + \gamma \sum_{j=1}^{n} x_j(1 - x_j) \\ \text{subject to} & Ax = b \\ & 0 \le x \le e. \end{array} \tag{P3.6}$$

For completeness, a proof is shown next.

Theorem 3.1. *Let $g : [0, 1]^n \to \mathbb{R}$ be a C^1 function and consider the two problems (P3.5) and (P3.6) with the feasible region of (P3.5) being non-empty. Then there exists $M > 0$ such that for all $\gamma > M$, problems (P3.5) and (P3.6) have the same minimizers.*

Proof. Let $b^{(l)}$ for $l = 1, 2, \ldots, 2^n$ denote the elements of the set $\{0, 1\}^n$, and B_l denote the set $\{y \in [0, 1]^n : \|y - b^{(l)}\| < \frac{1}{4}\}$. Suppose $x \in B_k$ for some k. Then for indices j

in which $b_j^{(k)} = 0$, we have $x_j = |x_j - b_j^{(k)}| \leq \|x - b^{(k)}\| \leq \frac{1}{4}$, so that

$$|x_j - b_j^{(k)}| = x_j \leq 2x_j(1 - x_j). \tag{3.4}$$

Similarly, for indices j in which $b_j^{(k)} = 1$, we have $1 - x_j \leq |x_j - b_j^{(k)}| \leq \|x - b^{(k)}\| \leq \frac{1}{4}$, i.e., $x_j \geq \frac{3}{4}$, so that

$$|x_j - b_j^{(k)}| = 1 - x_j \leq 2x_j(1 - x_j). \tag{3.5}$$

By Taylor's theorem, there exists some vector $\xi \in [0,1]^n$ such that $g(x) = g(b^{(k)}) + (\nabla g(\xi))^{\mathrm{T}}(x - b^{(k)})$. Since $\nabla g(x)$ is continuous on the compact set $[0,1]^n$, there exists some constant $m_0 > 0$ such that

$$\begin{aligned}
g(b^{(k)}) - g(x) &\leq |g(x) - g(b^{(k)})| \\
&= |(\nabla g(\xi))^{\mathrm{T}}(x - b^{(k)})| \\
&\leq m_0 \|x - b^{(k)}\| \\
&= m_0 \sqrt{\sum_{j=1}^{n} (x_j - b_j^{(k)})^2} \\
&\leq m_0 \sum_{j=1}^{n} |x_j - b_j^{(k)}| \\
&\leq 2m_0 \sum_{j=1}^{n} x_j(1 - x_j) \text{ (from (3.4) and (3.5)).}
\end{aligned}$$

So if $\gamma > 2L$,

$$g(b^{(k)}) \leq g(x) + \gamma \sum_{j=1}^{n} x_j(1 - x_j)$$

for $x \in B_k$.

Suppose $x \in X$ where $X = [0,1]^n \backslash (\cup_{l=1}^{2^n} B_l)$. By the continuity of $\sum_{j=1}^{n} x_j(1 - x_j)$ and g on the compact set X, there exist constants m_1 and m_2 such that $g(x) \geq m_1$ and $\sum_{j=1}^{n} x_j(1 - x_j) \geq m_2$ for all $x \in X$. In particular, $m_2 > 0$ since $x \neq b^{(l)}$ for all

l. This implies that for all $x \in X$,

$$\begin{aligned} g(x) + \gamma \sum_{j=1}^{n} x_j(1-x_j) &\geq m_1 + \gamma m_2 \\ &\geq g(b^{(l)}) \end{aligned}$$

for all l if $\gamma \geq \frac{m_3 - m_1}{m_2}$, where $m_3 = \max_l g(b^{(l)})$. Thus, if $\gamma > \max\{2L, \frac{m_3-m_1}{m_2}\}$, we have $m_3 \leq g(x) + \gamma \sum_{j=1}^{n} x_j(1-x_j)$ for all $x \in [0,1]^n$. Letting

$$l' = \underset{l: Ab^{(l)} = b}{\arg\min}\, g(b^{(l)})$$

(the value of the minimum is finite because the feasible region of (P3.5) is non-empty) and $M = \max\{2m_0, \frac{m_3-m_1}{m_2}\}$, we also have

$$\begin{aligned} g(b^{(l')}) + \gamma \sum_{j=1}^{n} b_j^{(l')}(1-b_j^{(l')}) &= g(b^{(l')}) \\ &\leq g(x) + \gamma \sum_{j=1}^{n} x_j(1-x_j) \end{aligned}$$

for all $x \in [0,1]^n \cap \{x : Ax = b\}$ if $\gamma > M$. The theorem follows from the observation that $b^{(l')}$ is the minimizer of both problems (P3.5) and (P3.6). $\square$

The idea of using penalty methods for discrete optimization problems is not new (see e.g., [Bor88]). However, to use the logarithmic smoothing function in combination with the penalty terms to form a path towards a local minimizer is a novel application of continuation methods for discrete optimization. It is not sufficient to introduce only the penalty terms and hope to obtain the global minimizer by solving the resulting problem, because many undesired stationary points may be introduced in the process. This flaw also applies to the process of transforming a discrete optimization problem into a global optimization problem simply by replacing the discrete requirements of the variables with a nonlinear constraint, such as replacing the integrality of the variables $x \in \{0, 1, \ldots, u\}$ by the constraint

$$\prod_{i=0}^{u}(x - i) = 0. \tag{3.6}$$

The likelihood is that every possible combination of integer variables is a local minimizer of the transformed problem. Hence, it may be extremely difficult to find the global minimizer for a problem with constraints of the form (3.6). To illustrate the danger of using the penalty function alone and the benefit of using a smoothing function, consider the following example.

Example 3.1. Consider the problem $\min\{x^2 : x \in \{0, 1\}\}$. It is clear that the global minimizer is given by $x^* = 0$. If the problem is transformed to $\min\{x^2 + \gamma x(1 - x)\}$, where $\gamma > 0$ is the penalty parameter, then the solution of the first-order optimality conditions without factoring in the boundary points $x = 0, 1$ is given by $x^*(\gamma) = \frac{\gamma}{2(\gamma-1)} > \frac{1}{2}$. A rounding of this $x^*(\gamma)$ for *any* positive value of γ would have produced the wrong solution of $x^* = 1$. This difficulty arises partly because an undesired maximizer was introduced by the penalty function. On the other hand, if the problem is transformed to $\min\{x^2 - \mu \log x - \mu \log(1-x) + \gamma x(1-x)\}$, where $\mu > 0$ is the barrier parameter, then solving the first-order optimality conditions without considering the boundary points $x = 0, 1$ for say $\gamma = 10$ and $\mu = 10$ and 1 gives the solutions lying in $[0, 1]$ as $x^*(\mu, \gamma) = 0.3588$ and 0.1072 respectively. Thus, rounding of $x^*(\mu, \gamma)$ in these cases will give the correct global minimizer of $x^* = 0$. In fact, a trajectory of $x^*(\mu, 10)$ can be obtained such that $x^*(\mu, 10) \to x^*$ as $\mu \searrow 0$.

Theorem 3.2. *Let $x(\gamma, \mu)$ be any local minimizer of (P3.4). Then $\lim_{\gamma\to\infty} \lim_{\mu\to 0} x_j(\gamma, \mu) =$ 0 or 1 for $j \in J$.*

Proof. Let $\gamma > 0$. We can rewrite the objective function in (P3.4) as

$$F(x) = f_\gamma(x) - \mu \sum_{j=1}^{n} [\ln x_j + \ln(1 - x_j)],$$

where $f_\gamma(x) = f(x) + \gamma \sum_{j \in J} x_j(1 - x_j)$. Also, let $x(\gamma)$ be a minimizer of

$$\begin{aligned} \text{Minimize} \quad & f_\gamma(x) \\ \text{subject to} \quad & Ax = b \\ & 0 \le x \le e. \end{aligned} \tag{P3.7}$$

From the discussion at the beginning of this section (replacing f with f_γ), we know that

$$\lim_{\mu \to 0} x(\gamma, \mu) = x(\gamma). \tag{3.7}$$

Observe that (P3.7) becomes a sequence of penalty subproblems for (P3.1) when we use γ as the penalty parameter. By Theorem 3.1, we know that $x_j(\gamma) \in \{0, 1\}$ for γ sufficiently large, so that $x_j(\gamma) \to 0$ or 1 as $\gamma \to \infty$ for each $j \in J$. From (3.7), we get the desired conclusion that $\lim_{\gamma \to \infty} \lim_{\mu \to 0} x_j(\gamma, \mu) = 0$ or 1 for $j \in J$. □

Note that extreme ill-conditioning arising out of $\gamma \to \infty$ is avoided because we do not need γ to be arbitrarily large as in the case of inexact penalty methods. In fact, a modestly large value of γ is sufficient to indicate whether a variable is converging to 0 or 1. A danger of the nonconvex terms arising in the objective function (usually from (P3.5)) is that we are likely to introduce local minimizers at the feasible integer points, and more significantly, saddle points at interior points. It is clearly critical that a method be used that will not converge to a saddle point.

Example 3.2. Let $f(x_1, x_2) = (x_1 - 0.6)^2 + (\sqrt{2} - 1.2)x_2$, $A = 0$, $b = 0$, $\mu = 0.1$, and $\gamma = 1$. The stationary point of the function, $(x_1, x_2) = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$, is a saddle point because $\nabla^2 F(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) = \frac{1}{5}\begin{bmatrix} 2\sqrt{2}+4 & 0 \\ 0 & 2\sqrt{2}-6 \end{bmatrix}$ is indefinite.

If the number of variables outstanding is not large, an alternative to forcing integrality by the penalty function is to examine all possible remaining combinations. Another possibility is to use the solution obtained as an initial point in an alternative method, perhaps fixing the variables that are already integral. Note that even if no term is used to force integrality, many variables of (P3.3) may be 0 or 1 at an optimal point. For example, if $f(x)$ is concave with a strictly negative definite Hessian, then there will be at least n active constraints, implying that there has to be at least $n-m$ variables being 0 or 1.

We have in effect replaced the hard problem of an integer discrete optimization problem by what at first appearance is an equally hard problem of finding a global minimizer for a problem with continuous variables and a large number of local minima. The basis for our optimism that this is not the case lies in how we can utilize the parameters μ and γ and try to obtain a global minimizer, or at least a good local minimizer of the composite objective function. Note that the term $x_j(1-x_j)$ attains its maximum at $x_j = \frac{1}{2}$ and that the logarithmic barrier term attains its minimum at the same point. Consequently, at this point, the gradient is given solely by $f(x)$. In other words, which vertex looks most attractive from the perspective of the objective is the direction we tilt regardless of the value of μ or γ. Starting at a neutral point and slowly imposing integrality is a key idea in the approach we advocate.

Also, note that provided μ is sufficiently large compared to γ, the problem will have a unique and hence global solution $x^*(\mu,\gamma)$, which is a continuous function of μ and γ. The hope is that the global or at least a good minimizer of (P3.1) is the one connected by a continuous trajectory to $x^*(\mu,\gamma)$ for μ large and γ small.

Even if a global minimizer is not identified, we hope to obtain a good local minimizer and perhaps combine this approach with traditional methods.

3.1.2 Description of the Algorithm

Since the continuous problems of interest may have many local minimizers and saddle points, first-order methods are inadequate as they are only assured of converging to points satisfying first-order optimality conditions. It is therefore imperative that second-order methods be used in the algorithm. Any second-order method that is assured of converging to a solution of the second-order optimality conditions must explicitly or implicitly compute a direction of negative curvature for the reduced Hessian matrix. A key feature of our approach is a very efficient second-order method for solving the continuous problem.

We may solve (P3.4) for a specific choice of μ and γ by starting at a feasible point and generating a descent direction, if one exists, in the null space of A. Let Z be a matrix with columns that form a basis for the null space of A. Then $AZ = 0$ and the rank of Z is $n - m$. If x_0 is any feasible point so that we have $Ax_0 = b$, the feasible region can be described as $\{x : x = x_0 + Zy,\ y \in \mathbb{R}^{n-m}\}$. Also, if we let ϕ be the restriction of F to the feasible region, the problem becomes the following unconstrained problem:

$$\underset{y \in \mathbb{R}^{n-m}}{\text{Minimize}} \quad \phi(y). \tag{P3.8}$$

Since the gradient of ϕ is $Z^{\mathrm{T}} \nabla F(x)$, it is straightforward to obtain a stationary point by solving the equation $Z^{\mathrm{T}} \nabla F(x) = 0$. This gradient is referred to as the *reduced gradient*. Likewise, the reduced Hessian, i.e., the Hessian of ϕ, is $Z^{\mathrm{T}} \nabla^2 F(x) Z$.

For small or moderately-sized problems, a variety of methods may be used (see e.g., [GM74, FGM95]). Here, we investigate the case where the number of variables is large. This is because transforming the original problem to one with 0-1 variables may increase the number of variables considerably. For example, if we have 100 variables each of which can take 20 values, the number of 0-1 variables of the transformed problem may be more than 2000.

One approach to solving the problem is to use a linesearch method, such as the truncated-Newton method (see [DS83]) we are adopting, in which the descent direction and a direction of negative curvature are computed. Instead of using the index set J for the definition of F in our discussion, we let γ be a vector of penalty parameters with zero values for those x_i such that $i \notin J$, and $\Gamma = \mathrm{Diag}(\gamma)$.

The first-order optimality conditions for (P3.4) may be written as

$$\begin{aligned} \nabla f - \mu X_g e + \Gamma\,(e - 2x) + A^{\mathrm{T}}\lambda &= 0 \\ Ax &= b, \end{aligned} \tag{3.8}$$

where $X_g = \mathrm{Diag}(x_g)$, $(x_g)_i = \frac{1}{x_i} - \frac{1}{1-x_i}, i = 1, \ldots, n$, and λ corresponds to the Lagrange multiplier of the constraint $Ax = b$. Applying Newton's method directly, we obtain the system

$$\begin{bmatrix} H & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f + \mu X_g e - \Gamma\,(e - 2x) - A^{\mathrm{T}}\lambda \\ b - Ax \end{bmatrix}, \tag{3.9}$$

where $H = \nabla^2 f + \mu X_H - 2\,\Gamma$ and X_H= Diag(x_H), $(x_H)_i = \frac{1}{x_i^2} + \frac{1}{(1-x_i)^2}$.

Assuming that x_0 satisfies $Ax_0 = b$, the second equation $A\triangle x = 0$ implies that $\triangle x = x_0 + Zy$ for some y. Substituting this into the first equation and premultiplying both sides by Z^{T}, we obtain

$$Z^{\mathrm{T}} HZy = Z^{\mathrm{T}}(-\nabla f + \mu X_g e - \Gamma\,(e - 2x)). \tag{3.10}$$

To obtain a descent direction in this method, we first attempt to solve (3.10), or from the definition of $F(x)$, the equivalent reduced-Hessian system

$$Z^{\mathrm{T}}\nabla^2 F(x_l)Zy = -Z^{\mathrm{T}}\nabla F(x_l), \tag{3.11}$$

by the conjugate gradient method, where x_l is the lth iterate. Generally, Z may be a large matrix, especially if the number of linear constraints is small. Thus, even though $\nabla^2 F(x_l)$ is likely to be a sparse matrix, $Z^{\mathrm{T}}\nabla^2 F(x_l)Z$ may be a large dense matrix. The virtue of the conjugate gradient method is that the explicit reduced Hessian need not be formed. There may be specific problems where the structure of $\nabla^2 F$ and Z does allow the matrix to be formed. Under such circumstances alternative methods such as those based on Cholesky factorization may also be applicable. Since we are interested in developing a method for general application we have pursued the conjugate gradient approach.

In the process of solving (3.11) with the conjugate gradient algorithm (see [GV96], [Bom99]), we may determine that $Z^{\mathrm{T}}\nabla^2 F(x_l)Z$ is indefinite for some l. In such a case, we shall obtain a negative curvature direction q such that

$$q^{\mathrm{T}} Z^{\mathrm{T}}\nabla^2 F(x_l)Zq < 0.$$

This negative curvature direction is used to ensure that the iterates do not converge to a saddle point. Also, the objective is decreased along this direction. In practice, the best negative curvature direction is Zq, where q is an eigenvector corresponding to the smallest eigenvalue of $Z^{\mathrm{T}}\nabla^2 F(x_l)Z$. Computing q is usually difficult but fortunately unnecessary. A good direction of negative curvature will suffice and efficient ways of computing such directions within a modified-Newton algorithm are described in [Bom99]. The descent direction in such modified-Newton algorithms can be obtained using factorization methods (e.g., [FM93]), or by solving differential equations [Del00]. In any case, it is essential to compute both a descent direction and a direction of negative curvature (when one exists). One possibility that we may encounter is that the conjugate gradient algorithm may terminate with a direction q such that $q^{\mathrm{T}}Z^{\mathrm{T}}\nabla^2 F(x_l)Zq = 0$. In that case, we may have to use other iterative methods such as the Lanczos method to obtain a direction of negative curvature.

If we let p be a suitable combination of the negative curvature direction q with a descent direction, the convergence of the iterates is still ensured with the search direction Zp. The next iterate x_{l+1} is thus defined by $x_l + \alpha_l Zp$, where α_l is being determined using a line search. The iterations will be performed until $Z^{\mathrm{T}}\nabla F(x_l)$ is sufficiently close to 0 and $Z^{\mathrm{T}}\nabla^2 F(x_l)Z$ is positive semi-definite. Also, as the current iterate x_l may still be some distance away from the actual optimal solution we are seeking, and since we do not necessarily use an exact solution of (3.11) to get the search direction, we only need to solve (3.11) approximately.

Note that we do not make use of the information provided by λ or solve for $\triangle\lambda$ in (3.9). Such a method is known as the *primal logarithmic barrier* method in the literature of interior point methods. However, in the next section, we shall reformulate problem (P3.3) as one that could make use of the Lagrange multiplier information, and the resulting method is known as the primal-dual logarithmic barrier method. Since both methods ultimately produce iterates that converge to the same limit points, we focus our attention on the primal logarithmic barrier method in this thesis.

A summary of the primal algorithm is shown on the next page. Certain implementation issues of this algorithm need to be addressed. For example, barrier methods with a conjugate gradient algorithm are generally not successful without proper preconditioning. Such matters are dealt with in Chapters 4 and 5.

3.2 The Primal-Dual Method

Primal-dual methods (see e.g., [Wri97]) have been used to solve linear and nonlinear programming problems with much success. In this section, we use a similar approach to deal with our problem. We introduce variables $s_j \triangleq 1 - x_j$ and again let γ being a vector of penalty parameters with zero values for those x_j with $j \notin J$, so that problem (P3.4) becomes

$$\begin{array}{ll} \text{Minimize} & F(x,s) \triangleq f(x) - \mu \sum_{j=1}^{n} [\ln x_j + \ln(s_j)] + \sum_{j=1}^{n} \gamma_j x_j s_j \\ \text{subject to} & Ax = b \\ & x + s = e. \end{array} \tag{P3.9}$$

The first-order optimality conditions can then be written as

$$\begin{aligned} \nabla f(x) - \mu X^{-1} e + \Gamma\, s + A^{\mathrm{T}} \lambda + \pi &= 0 \\ -\mu S^{-1} e + \Gamma\, x + \pi &= 0 \\ Ax &= b \\ x + s &= e, \end{aligned} \tag{3.12}$$

where $X = \mathrm{Diag}(x)$, $S = \mathrm{Diag}(s)$, $\Gamma = \mathrm{Diag}(\gamma)$, and λ and π correspond to the Lagrange multipliers of constraints $Ax = b$ and $x + s = e$ respectively. The direct approach is to apply Newton's method to these equations as in the previous section. However, it can be observed that the first two equations of (3.12) are highly nonlinear, implying that Newton's method may perform poorly. The key idea of the primal-dual approach is to alter these equations.

Algorithm 3.1: Primal Logarithmic Barrier Smoothing

Set ϵ_F = tolerance for function evaluation,
ϵ_μ = tolerance for barrier/penalty value,
M = maximum penalty value,
N = iteration limit for applying Newton's method,
θ_μ = reduction ratio for barrier parameter,
θ_γ = reduction ratio for penalty parameter,
μ_0 = initial barrier parameter,
γ_0 = initial penalty parameter,
r = any feasible starting point.
Set $\gamma = \gamma_0$,
$\mu = \mu_0$.
while $\gamma < M$ or $\mu > \epsilon_\mu$
  Set $x_0 = r$.
  for $l = 0, 1, \ldots, N$
    if $\|Z^{\mathrm{T}}\nabla F(x_l)\| < \epsilon_F \mu$
      Set $x_N = x_l$, $l = N$.
      Check if x_N is a direction of negative curvature.
    else
      Apply conjugate gradient algorithm to
        $[Z^{\mathrm{T}}\nabla^2 F(x_l)Z]y = -Z^{\mathrm{T}}\nabla F(x_l)$.
      Obtain p_l as a combination of directions of descent and negative curvature.
      Perform a linesearch to determine α_l and set $x_{l+1} = x_l + \alpha_l Z p_l$.
    end if
  end for
  Set $r = x_N$,
    $\mu = \theta_\mu \mu$,
    $\gamma = \frac{\gamma}{\theta_\gamma}$.
end while

If we introduce variables $\psi = \mu X^{-1}e$ and $\phi = \mu S^{-1}e$, an equivalent system is given by

$$\begin{aligned} \nabla f(x) - \psi + \Gamma\, s + A^{\mathrm{T}}\lambda + \pi &= 0 \\ -\phi + \Gamma\, x + \pi &= 0 \\ Ax &= b \\ x + s &= e \\ X\psi &= \mu e \\ S\phi &= \mu e. \end{aligned} \tag{3.13}$$

These equations are much less nonlinear than (3.12), especially in the neighborhood of $x_j = 0$ or $s_j = 0$.

The primal-dual method is the application of Newton's method to (3.13). Thus, we get the Newton search direction $(\triangle x, \triangle s, \triangle\psi, \triangle\phi, \triangle\lambda, \triangle\pi)$ as the solution to the following set of equations:

$$\begin{bmatrix} \nabla^2 f(x) & \Gamma & -I & 0 & A^{\mathrm{T}} & I \\ \Gamma & 0 & 0 & -I & 0 & I \\ A & 0 & 0 & 0 & 0 & 0 \\ I & I & 0 & 0 & 0 & 0 \\ \Psi & 0 & X & 0 & 0 & 0 \\ 0 & \Phi & 0 & S & 0 & 0 \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle s \\ \triangle\psi \\ \triangle\phi \\ \triangle\lambda \\ \triangle\pi \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{bmatrix}, \tag{3.14}$$

where $\Psi = \mathrm{Diag}(\psi)$, $\Phi = \mathrm{Diag}(\phi)$, and

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{bmatrix} \equiv \begin{bmatrix} \psi - \Gamma\, s - \nabla f - A^{\mathrm{T}}\lambda - \pi \\ \phi - \Gamma\, x - \pi \\ b - Ax \\ e - x - s \\ \mu e - X\psi \\ \mu e - S\phi \end{bmatrix}.$$

This can be reduced to first solving the system

$$\begin{bmatrix} H_1 & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle \lambda \end{bmatrix} = \begin{bmatrix} u \\ r_3 \end{bmatrix}, \tag{3.15}$$

where

$$\begin{aligned} H_1 &= \nabla^2 f + X^{-1}\Psi + S^{-1}\Phi - 2\,\Gamma \\ u &= r_1 - r_2 - \Gamma\, r_4 + S^{-1}\Phi\, r_4 + X^{-1} r_5 - S^{-1} r_6. \end{aligned}$$

The rest of the Newton directions can then be obtained by the following formulas:

$$\begin{aligned} \triangle s &= r_4 - \triangle x \\ \triangle \psi &= X^{-1}[r_5 - \Psi \triangle x] \\ \triangle \phi &= S^{-1}[\Phi \triangle x + r_6 - \Phi\, r_4] \\ \triangle \pi &= r_2 + [S^{-1}\Phi - \Gamma]\triangle x + S^{-1}[r_6 - \Phi\, r_4]. \end{aligned}$$

To get the desired search direction, we can make use of the conjugate gradient algorithm on (3.15) to obtain a descent direction or negative curvature direction. A suitable combination of the descent direction and negative curvature directions will then be used. More precisely, we first apply the null-space approach described in Section 3.1. Letting Z be the null-space matrix of A and x_0 be any point such that $Ax_0 = r_3$, we find that $\triangle x = x_0 + Zy$ for some y. Substituting this into the top part of (3.15) and premultiplying both sides by Z^{T}, we get

$$Z^{\mathrm{T}} H_1 Z y = Z^{\mathrm{T}} u - Z^{\mathrm{T}} H_1 w, \tag{3.16}$$

where y could be determined using a conjugate gradient method. After the corresponding $\triangle x$ is obtained, $\triangle \lambda$ can then be obtained from

$$A^{\mathrm{T}} \triangle \lambda = u - H_1 \triangle x.$$

If $r_3 = 0$, then $x_0 = 0$ would be suitable and we have

$$Z^{\mathrm{T}} H_1 Z y = Z^{\mathrm{T}} u. \tag{3.17}$$

From the definitions of u and $r_1, \ldots, r_6$, we find that

$$\begin{aligned} Z^{\mathrm{T}} u &= Z^{\mathrm{T}}(-\nabla f + \mu X^{-1} e - \Gamma s - \mu S^{-1} e + \Gamma x + S^{-1} \Phi r_4) \\ &= Z^{\mathrm{T}}(-\nabla f + \mu X^{-1} e - \Gamma s - \mu S^{-1} e + \Gamma x) \end{aligned} \tag{3.18}$$

if we also have $r_4 = 0$, i.e., $x + s = e$.

Now, the null-space matrix for the linear equations in (P3.9) is $\bar{Z} = \begin{bmatrix} Z \\ -Z \end{bmatrix}$, where Z is the null-space matrix for A. The corresponding reduced Hessian system is

$$\bar{Z}^{\mathrm{T}} \nabla^2 F(x_l, s_l) \bar{Z} w = -\bar{Z}^{\mathrm{T}} \nabla F(x_l, s_l), \tag{3.19}$$

and on dropping the x_l and s_l notation, we find the right-hand side (rhs) is

$$\begin{aligned} -\bar{Z}^{\mathrm{T}} \nabla F &= -\begin{bmatrix} Z^{\mathrm{T}} & -Z^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \nabla_x F \\ \nabla_s F \end{bmatrix} \\ &= -Z^{\mathrm{T}} \nabla_x F + Z^{\mathrm{T}} \nabla_s F \\ &= Z^{\mathrm{T}}(-\nabla f + \mu X^{-1} e - \Gamma s - \mu S^{-1} e + \Gamma x). \end{aligned} \tag{3.20}$$

It follows that (3.18) and (3.20) have the same rhs and consequently, the direction obtained from the primal-dual procedure is assured of being a descent direction for $F(x, s)$. However, any direction of negative curvature obtained is with respect to H_1 and not H. But this is not really a difficulty if we have a direction of sufficient negative curvature for H_1, as seen in the following result.

Theorem 3.3. *Let the optimality conditions in equations (3.8) and (3.13) be satisfied by* $\chi^*(\mu) = (\tilde{x}^*(\mu), \tilde{\lambda}^*(\mu))$ *and* $\zeta^*(\mu) = (x^*(\mu), s^*(\mu), \psi^*(\mu), \phi^*(\mu), \lambda^*(\mu), \pi^*(\mu))$ *respectively. Also, let* $\{\mu_k\}_{k=1}^{\infty}$ *be a sequence of positive numbers such that* $\lim_{k \to \infty} \mu_k = 0$, *and* $\{\chi(\mu_k)\}_{k=1}^{\infty}$ *and* $\{\zeta(\mu_k)\}_{k=1}^{\infty}$ *be two sequences of iterates such that*

$$\lim_{k\to\infty} \|\chi(\mu_k) - \chi^*(\mu_k)\| \to 0 \tag{3.21}$$

$$\lim_{k\to\infty} \|\zeta(\mu_k) - \zeta^*(\mu_k)\| \to 0. \tag{3.22}$$

Suppose d *satisfies* $\|d\| = 1$ *and*

$$d^{\mathrm{T}} Z^{\mathrm{T}} H_1(\zeta(\mu)) Z d \le -\alpha \tag{3.23}$$

for some $\alpha > 0$ *and all* k*. Then there exist* β *and* $K > 0$ *such that for all* $k \ge K$*,*

$$d^{\mathrm{T}} Z^{\mathrm{T}} H(\chi(\mu_k)) Z d \le -\beta. \tag{3.24}$$

Proof. Since the optimality conditions in (3.8) and (3.13) are equivalent, we have $\tilde{x}^*(\mu) = x^*(\mu)$ and $\tilde{\lambda}^*(\mu) = \lambda^*(\mu)$. Note that by (3.22),

$$\lim_{k\to\infty} \frac{\psi_i(\mu_k)}{x_i(\mu_k)} = \lim_{k\to\infty} \frac{\psi_i^*(\mu_k)}{x_i^*(\mu_k)} = \lim_{k\to\infty} \frac{\mu_k}{(x_i^*(\mu_k))^2} = \lim_{k\to\infty} \frac{\mu_k}{(\tilde{x}_i^*(\mu_k))^2}. \tag{3.25}$$

If we let I_1 be the index set such that

$$\lim_{k\to\infty} \frac{\psi_i(\mu_k)}{x_i(\mu_k)} \to \infty \text{ for } i \in I_1, \tag{3.26}$$

then (3.25) implies that

$$\lim_{k\to\infty} \frac{\mu_k}{(\tilde{x}_i^*(\mu_k))^2} \to \infty \text{ for } i \in I_1. \tag{3.27}$$

Similarly, (3.22) implies that

$$\begin{aligned} \lim_{k\to\infty} \frac{\phi_i(\mu_k)}{s_i(\mu_k)} &= \lim_{k\to\infty} \frac{\phi_i^*(\mu_k)}{s_i^*(\mu_k)} = \lim_{k\to\infty} \frac{\mu_k}{(s_i^*(\mu_k))^2} = \lim_{k\to\infty} \frac{\mu_k}{(1 - x_i^*(\mu_k))^2} \\ &= \lim_{k\to\infty} \frac{\mu_k}{(1 - \tilde{x}_i^*(\mu_k))^2}. \end{aligned} \tag{3.28}$$

Therefore, if I_2 is the index set such that

$$\lim_{k\to\infty} \frac{\phi_i(\mu_k)}{s_i(\mu_k)} \to \infty \text{ for } i \in I_2, \tag{3.29}$$

then

$$\lim_{k\to\infty} \frac{\mu_k}{(1-\tilde{x}_i^*(\mu_k))^2} \to \infty \text{ for } i \in I_2. \tag{3.30}$$

Let $I = I_1 \cup I_2$ and $J = I^C$. Without loss of generality, assume the variables x are permuted such that $d = [d_I, d_J]$. Then

$$\begin{aligned} d^{\mathrm{T}} Z^{\mathrm{T}} H_1(\zeta(\mu_k)) Z d = (Zd)^{\mathrm{T}} [\nabla^2 f(x(\mu_k)) - 2\Gamma + (X(\mu_k))^{-1}\Psi(\mu_k) \\ + (S(\mu_k))^{-1}\Phi(\mu_k)] Zd \\ = (Zd)^{\mathrm{T}} \left(\nabla^2 f(x(\mu_k)) - 2\Gamma\right) Zd + (Zd)_I^{\mathrm{T}} D_I(\mu_k)(Zd)_I \\ + (Zd)_J^{\mathrm{T}} D_J(\mu_k)(Zd)_J, \end{aligned}$$

where $(X(\mu_k))^{-1}\Psi(\mu_k) + (S(\mu_k))^{-1}\Phi(\mu_k) = \begin{bmatrix} D_I(\mu_k) & 0 \\ 0 & D_J(\mu_k) \end{bmatrix}$. Since $D_I(\mu_k)$ is a diagonal matrix with diagonal elements converging to ∞ as $k \to \infty$, the term $(Zd)_I^{\mathrm{T}} D_I(\mu_k)(Zd)_I$ will also converge to ∞ if $(Zd)_I \neq 0$. By the continuity of $\nabla^2 f(x(\mu_k)) - 2\Gamma$ in the compact set $[0,1]^n$ and the definition of J, the expression $(Zd)^{\mathrm{T}} (\nabla^2 f(x(\mu_k)) - 2\Gamma) Zd + (Zd)_J^{\mathrm{T}} D_J (Zd)_J$ is bounded for $\|d\| = 1$. Since d must satisfy (3.23), this implies that $(Zd)_I = 0$. Note that $Zd \neq 0$ from (3.23). Let

$$\tilde{X}_H(\mu_k) = \begin{bmatrix} \tilde{D}_I(\mu_k) & 0 \\ 0 & \tilde{D}_J(\mu_k) \end{bmatrix}.$$

By (3.25) and (3.28), there exists $K_1 > 0$ such that for $k \geq K_1$,

$$\|D_J(\mu_k) - \tilde{D}_J(\mu_k)\| < \frac{\alpha}{4\|Zd\|^2}. \tag{3.31}$$

By (3.21) and (3.22) again, we find that

$$\lim_{k\to\infty} x_i(\mu_k) = \lim_{k\to\infty} x_i^*(\mu_k) = \lim_{k\to\infty} \tilde{x}_i^*(\mu_k) = \lim_{k\to\infty} \tilde{x}_i(\mu_k),$$

so that the finiteness of the limits implies that

$$\lim_{k\to\infty} (x_i(\mu_k) - \tilde{x}_i(\mu_k)) = 0.$$

Then by the continuity of $\nabla^2 f$ over $[0,1]^n$, there exists $K_2 > 0$ such that for all $k \geq K_2$,

$$\|\nabla^2 f(x(\mu_k)) - \nabla^2 f(\tilde{x}(\mu_k))\| \leq \frac{\alpha}{4\|Zd\|^2}. \tag{3.32}$$

Therefore, (3.31), (3.32) and $(Zd)_I = 0$ implies that for $k \geq K = \max\{K_1, K_2\}$,

$$\begin{aligned}
d^{\mathrm{T}} Z^{\mathrm{T}} H(\chi(\mu_k)) Zd &= (Zd)^{\mathrm{T}} \left(H(\chi(\mu_k)) - H_1(\zeta(\mu_k))\right) Zd + (Zd)^{\mathrm{T}} H_1(\zeta(\mu_k)) Zd \\
&\leq (Zd)^{\mathrm{T}} [\nabla^2 f(x(\mu_k)) - 2\Gamma + (X(\mu_k))^{-1}\Psi(\mu_k) + (S(\mu_k))^{-1}\Phi(\mu_k) \\
&\qquad - \nabla^2 f(\tilde{x}(\mu_k)) + 2\Gamma - \tilde{X}_H(\mu_k)] Zd - \alpha \\
&= (Zd)^{\mathrm{T}} \left(\nabla^2 f(x(\mu_k)) - \nabla^2 f(\tilde{x}(\mu_k))\right) Zd \\
&\qquad + (Zd)_I^{\mathrm{T}} (D_I(\mu_k) - \tilde{D}_I(\mu_k))(Zd)_I \\
&\qquad + (Zd)_J^{\mathrm{T}} (D_J(\mu_k) - \tilde{D}_J(\mu_k))(Zd)_J - \alpha \\
&\leq \|Zd\|^2 \|\nabla^2 f(x(\mu_k)) - \nabla^2 f(\tilde{x}(\mu_k))\| \\
&\qquad + \|Zd\|^2 \|D_J(\mu_k) - \tilde{D}_J(\mu_k)\| - \alpha \\
&\leq \frac{\alpha}{4} + \frac{\alpha}{4} - \alpha \\
&= -\frac{\alpha}{2}.
\end{aligned}$$

Thus, d satisfies (3.24) with $\beta = \dfrac{\alpha}{2}$. $\square$

In primal-dual methods, it is necessary to measure the progress of the iterates $(x_l, \lambda_l, s_l, \psi_l, \phi_l, \pi_l)$ towards satisfying both feasibility and optimality conditions by means of a merit function, M. For purposes of convergence analysis discussed in the next section, we define the merit function to be

$$M(x,s,\psi,\phi) = \max\left\{ \left\| \bar{Z}^{\mathrm{T}} \begin{bmatrix} \nabla f(x) + \Gamma\, s - \psi \\ \Gamma\, x - \phi \end{bmatrix} \right\|, \|X\psi - \mu e\|, \|S\phi - \mu e\| \right\}.$$

We can also define the merit function to be the objective function

$$M(x,s) = f(x) - \mu \sum_{j=1}^{n} [\ln x_j + \ln s_j] + \sum_{j=1}^{n} \gamma_j x_j s_j,$$

if the iterates are feasible. Otherwise, if we start the smoothing algorithm from any infeasible point, the merit function could be

$$\begin{aligned} M(x,s,\rho_1,\rho_2) = f(x) - \mu \sum_{j=1}^{n} [\ln x_j + \ln s_j] &+ \sum_{j=1}^{n} \gamma_j x_j s_j \\ &+ \rho_1 \|Ax - b\|_1 + \rho_2 \|x + s - e\|_1, \end{aligned}$$

where ρ_1 and ρ_2 are fixed positive numbers.

A summary of the primal-dual algorithm is given on the next page.

3.3 Convergence Analysis

We have mentioned earlier that the local minima of the barrier subproblems introduced by the smoothing algorithm will lead to the local minimum of the following problem:

$$\begin{array}{ll} \text{Minimize} & F(x,s) \triangleq f(x) + \sum_{j=1}^{n} \gamma_j x_j s_j \\ \text{subject to} & Ax = b \\ & x + s = e \\ & (x,s) \geq 0. \end{array} \tag{P3.10}$$

In addition, we will show that given iterates satisfying the first- and second-order optimality conditions for the barrier subproblems of (P3.10), we will have a limit point

Algorithm 3.2: Primal-Dual Logarithmic Barrier Smoothing

Set ϵ_F = tolerance for function evaluation,
ϵ_μ = tolerance for barrier/penalty value,
M = maximum norm for penalty matrix,
N = iteration limit for applying Newton's method,
θ_μ = reduction ratio for barrier parameter,
θ_γ = reduction ratio for penalty parameter,
μ_0 = initial barrier parameter,
Γ_0 = diagonal matrix with initial penalty parameters as diagonal entries,
r = any feasible starting point.
Set $\gamma = \gamma_0$,
$\mu = \mu_0$.
while $\|\Gamma\| < M$ or $\mu > \epsilon_\mu$
  Set $(x_0, \lambda_0, s_0, \psi_0, \phi_0, \pi_0) = r$.
  for $l = 0, 1, \ldots, N$
    if $M(x_l, s_l) < \epsilon_F \mu$
      Set $x_N = x_l, \lambda_N = \lambda_l, s_N = s_l, \psi_N = \psi_l, \phi_N = \phi_l, \pi_N = \pi_l, l = N$.
      Check if x_N is a direction of negative curvature.
    else
      Apply conjugate gradient algorithm to following equation
        $[Z^{\mathrm{T}} H_1(x_l, \lambda_l, s_l, \psi_l, \phi_l, \pi_l) Z] y = -Z^{\mathrm{T}} u(x_l, \lambda_l, s_l, \psi_l, \phi_l, \pi_l)$.
      Obtain $\triangle x, \triangle \lambda, \triangle s, \triangle \psi, \triangle \phi, \triangle \pi$.
      Perform a linesearch to determine α_l and set $x_{l+1} = x_l + \alpha_l \triangle x$.
      Update λ_{l+1}, s_{l+1}, ψ_{l+1}, ϕ_{l+1} and π_{l+1}.
    end if
  end for
  Set $r = (x_N, \lambda_N, s_N, \psi_N, \phi_N, \pi_N)$,
    $\mu = \theta_\mu \mu$,
    $\Gamma = \frac{1}{\theta_\gamma} \Gamma$.
end while

of these iterates that also satisfy the first- and second-order optimality conditions of (P3.10). In practice, we need to terminate an algorithm prior to satisfying the optimality conditions exactly. Consequently, we need to study the convergence of such a sequence of iterates.

We begin by considering the convergence properties of the iterates that do satisfy the optimality conditions of the barrier subproblems.

Lemma 3.2. *Let $\zeta(\mu) = (x(\mu), s(\mu), \psi(\mu), \phi(\mu), \lambda(\mu), \pi(\mu))$ be a vector satisfying the optimality conditions (3.13), and $\{\mu_k\}_{k=1}^{\infty}$ be a sequence of positive numbers such that $\lim_{k\to\infty} \mu_k = 0$. Also, assume that there exists a vector $(\bar{x}, \bar{s}, \bar{\lambda}, \bar{\pi})$ satisfying the following set of optimality conditions of problem (P3.10) with $(\bar{x}, \bar{s}) > 0$:*

$$\nabla f(x) - \psi + \Gamma\, s + A^{\mathrm{T}}\lambda + \pi = 0 \tag{3.33}$$

$$-\phi + \Gamma\, x + \pi = 0 \tag{3.34}$$

$$Ax = b \tag{3.35}$$

$$x + s = e \tag{3.36}$$

$$X\psi = 0 \tag{3.37}$$

$$S\phi = 0. \tag{3.38}$$

Then the sequence $\{\zeta(\mu_k)\}_{k=1}^{\infty}$ is bounded.

Proof. From (3.13), we have for each k,

$$\nabla f(x(\mu_k)) - \psi(\mu_k) + \Gamma\, s(\mu_k) + A^{\mathrm{T}}\lambda(\mu_k) + \pi(\mu_k) = 0 \tag{3.39}$$

$$-\phi(\mu_k) + \Gamma\, x(\mu_k) + \pi(\mu_k) = 0 \tag{3.40}$$

$$Ax(\mu_k) = b \tag{3.41}$$

$$x(\mu_k) + s(\mu_k) = e \tag{3.42}$$

$$X(\mu_k)\psi(\mu_k) = \mu_k e \tag{3.43}$$

$$S(\mu_k)\phi(\mu_k) = \mu_k e. \tag{3.44}$$

Since $x(\mu_k)$, $s(\mu_k) \in (0,1)^n$ for all k, we find that $\|(x(\mu_k), s(\mu_k))\| < K_1$ for some $K_1 > 0$.

From (3.35), (3.36), (3.41) and (3.42), we have

$$\begin{bmatrix} A & 0 \\ I & I \end{bmatrix} \begin{bmatrix} x(\mu_k) - \bar{x} \\ s(\mu_k) - \bar{s} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{3.45}$$

Also, from (3.33), (3.34), (3.39) and (3.40), we obtain

$$\begin{bmatrix} \nabla f(x(\mu_k)) - \nabla f(\bar{x}) + \Gamma(s(\mu_k) - \bar{s}) \\ \Gamma(x(\mu_k) - \bar{x}) \end{bmatrix} + \begin{bmatrix} A & 0 \\ I & I \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \lambda(\mu_k) - \bar{\lambda} \\ \pi(\mu_k) - \bar{\pi} \end{bmatrix} - \begin{bmatrix} \psi(\mu_k) - \bar{\psi} \\ \phi(\mu_k) - \bar{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{3.46}$$

Premultiplying both sides of (3.46) by $\begin{bmatrix} x(\mu_k) - \bar{x} \\ s(\mu_k) - \bar{s} \end{bmatrix}^{\mathrm{T}}$ and using (3.37), (3.38), (3.43)–(3.45), we get

$$\begin{aligned} (x(\mu_k) - \bar{x})^{\mathrm{T}}(\nabla f(x(\mu_k)) - \nabla f(\bar{x}) + \Gamma(s(\mu_k) - \bar{s})) + (s(\mu_k) - \bar{s})^{\mathrm{T}}\Gamma(x(\mu_k) - \bar{x}) \\ = (x(\mu_k) - \bar{x})^{\mathrm{T}}(\psi(\mu_k) - \bar{\psi}) + (s(\mu_k) - \bar{s})^{\mathrm{T}}(\phi(\mu_k) - \bar{\phi}) \\ = 2n\mu_k - \bar{x}^{\mathrm{T}}\psi(\mu_k) - x(\mu_k)\bar{\psi} - \bar{s}^{\mathrm{T}}\phi(\mu_k) - s(\mu_k)\bar{\phi}. \end{aligned}$$

This implies that for each j,

$$\begin{aligned} 0 \le \bar{x}_j\psi_j(\mu_k) &\le \bar{x}^{\mathrm{T}}\psi(\mu_k) + \bar{s}^{\mathrm{T}}\phi(\mu_k) \text{ (by nonnegativity of } \bar{x}, \bar{s}, \psi(\mu_k), \phi(\mu_k)) \\ &\le L_1, \end{aligned}$$

where $L_1 > 0$ is such that $L_1 \ge |2n\mu_k - (x(\mu_k) - \bar{x})^{\mathrm{T}}\psi(\mu_k) + (s(\mu_k) - \bar{s})^{\mathrm{T}}\phi(\mu_k)|$ for all k by continuity of ∇f over the compact set $[0,1]^n$ that contains the vectors $\bar{x}$, $x(\mu_k)$, $\bar{s}$, and $s(\mu_k)$. Thus, if $L_2 = \min_l \bar{x}_l > 0$, we have for all j

$$\psi_j(\mu_k) \le \frac{L_1}{L_2},$$

i.e., $\|\psi(\mu_k)\| < K_2$ for some $K_2 > 0$. Likewise, $\|\phi(\mu_k)\| < K_3$ for some $K_3 > 0$. From (3.40),

$$\pi(\mu_k) = \phi(\mu_k) - \Gamma\, x(\mu_k). \tag{3.47}$$

Since the rhs of (3.47) is bounded for all k, we also have $\|\pi(\mu_k)\| < K_4$ for some $K_4 > 0$. It remains to prove that $\lambda(\mu_k)$ is bounded for all k. Observe from (3.39) that

$$A^{\mathrm{T}}\lambda(\mu_k) = -\nabla f(x(\mu_k)) + \psi(\mu_k) - \Gamma\, s(\mu_k) - \pi(\mu_k),$$

so that

$$\begin{aligned}\|\lambda(\mu_k)\| &= \|(AA^{\mathrm{T}})^{-1}(-\nabla f(x(\mu_k)) + \psi(\mu_k) - \Gamma\, s(\mu_k) - \pi(\mu_k))\| \\ &\le \|(AA^{\mathrm{T}})^{-1}\|\| - \nabla f(x(\mu_k)) + \psi(\mu_k) - \Gamma\, s(\mu_k) - \pi(\mu_k)\|.\end{aligned}$$

Again, the boundedness of the rhs implies that $\|\lambda(\mu_k)\| < K_5$ for some $K_5 > 0$. □

Lemma 3.3. *Suppose the assumptions of Lemma 3.2 hold. Then there exists a limit point ζ^* of the sequence $\{\zeta(\mu_k)\}_{k=1}^{\infty}$ that satisfies the optimality conditions (3.33)–(3.38).*

Proof. The boundedness of the sequence $\{\zeta(\mu_k)\}_{k=1}^{\infty}$ from Lemma 3.2 implies that it has a limit point $\zeta^* = (x^*, s^*, \psi^*, \phi^*, \lambda^*, \mu^*)$, i.e., there is a subsequence $\mathcal{K}$ such that

$$\lim_{\substack{k\to\infty \\ k\in\mathcal{K}}} \zeta(\mu_k) = \zeta^*. \tag{3.48}$$

Taking limits as $k \to \infty$, $k \in \mathcal{K}$ on both sides of (3.39)–(3.44) and using (3.48) gives the desired result. □

Corollary 3.2. *Suppose the assumptions of Lemma 3.2 hold. If $\zeta(\mu_k)$ satisfies both the first- and second-order optimality conditions of problem (P3.9), then there exists a limit point ζ^* satisfying the first- and second-order optimality conditions of problem (P3.10).*

Proof. From Lemma 3.3, it suffices to show that the minimum eigenvalue of the reduced Hessian with respect to ζ^* is nonnegative. But this follows from the continuity of the reduced Hessian, the nonnegativity of the minimum eigenvalue of the reduced Hessian with respect to $\zeta(\mu_k)$, and (3.48). □

Now we consider the case when the set of iterates only satisfy the optimality conditions of the barrier subproblems approximately.

Lemma 3.4. *Suppose* $\{\bar{\zeta}(\mu_k)\}_{k=1}^{\infty}$ *is a set of iterates generated by Algorithm 3.2. Then as* $k \to \infty$, $\bar{\zeta}(\mu_k)$ *satisfies the conditions*

$$\nabla f(\bar{x}(\mu_k)) - \bar{\psi}(\mu_k) + \Gamma\, \bar{s}(\mu_k) + A^{\mathrm{T}}\bar{\lambda}(\mu_k) + \bar{\pi}(\mu_k) = O(\mu_k) \tag{3.49}$$

$$-\bar{\phi}(\mu_k) + \Gamma\, \bar{x}(\mu_k) + \bar{\pi}(\mu_k) = O(\mu_k) \tag{3.50}$$

$$A\bar{x}(\mu_k) = b \tag{3.51}$$

$$\bar{x}(\mu_k) + \bar{s}(\mu_k) = e \tag{3.52}$$

$$\bar{X}(\mu_k)\bar{\psi}(\mu_k) = O(\mu_k)e \tag{3.53}$$

$$\bar{S}(\mu_k)\bar{\phi}(\mu_k) = O(\mu_k)e. \tag{3.54}$$

Proof. Clearly, (3.51) and (3.52) hold because we are starting with a feasible iterate. From the termination criteria of Algorithm 3.2, we have

$$\bar{Z}^{\mathrm{T}} \begin{bmatrix} \nabla f(\bar{x}(\mu_k) + \Gamma\, \bar{s}(\mu_k) - \bar{\psi} \\ \Gamma\, \bar{x}(\mu_k) - \bar{\phi} \end{bmatrix} = O(\mu_k)e \tag{3.55}$$

$$\bar{\psi}(\mu_k) - \mu_k \bar{X}(\mu_k)^{-1}e = O(\mu_k)e \tag{3.56}$$

$$\bar{\phi}(\mu_k) - \mu_k \bar{S}(\mu_k)^{-1}e = O(\mu_k)e, \tag{3.57}$$

where $\bar{Z}$ is the null-space matrix of $\begin{bmatrix} A & 0 \\ I & I \end{bmatrix}$. Then

$$\begin{aligned}
&\bar{Z}^{\mathrm{T}} \begin{bmatrix} \nabla f(\bar{x}(\mu_k)) - \bar{\psi}(\mu_k) + \Gamma\, \bar{s}(\mu_k) + A^{\mathrm{T}}\bar{\lambda}(\mu_k) + \bar{\pi}(\mu_k) \\ -\bar{\phi}(\mu_k) + \Gamma\, \bar{x}(\mu_k) + \bar{\pi}(\mu_k) \end{bmatrix} \\
&= \bar{Z}^{\mathrm{T}} \left(\begin{bmatrix} \nabla f(\bar{x}(\mu_k)) - \bar{\psi}(\mu_k) + \Gamma\, \bar{s}(\mu_k) \\ -\bar{\phi}(\mu_k) + \Gamma\, \bar{x}(\mu_k) \end{bmatrix} + \begin{bmatrix} A & 0 \\ I & I \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \bar{\lambda}(\mu_k) \\ \bar{\pi}(\mu_k) \end{bmatrix} \right) \\
&= \bar{Z}^{\mathrm{T}} \begin{bmatrix} \nabla f(\bar{x}(\mu_k) + \Gamma\, \bar{s}(\mu_k) - \bar{\psi} \\ \Gamma\, \bar{x}(\mu_k) - \bar{\phi} \end{bmatrix} = O(\mu_k) \text{ (by (3.55))}.
\end{aligned}$$

This implies (3.49) and (3.50). On the other hand, multiplying both sides of (3.56) by $X(\mu_k)$, we obtain

$$\begin{aligned}\bar{X}(\mu_k)\bar{\psi}(\mu_k) - \mu_k e &= O(\mu_k)X(\mu_k)e \\ &= O(\mu_k)e,\end{aligned}$$

since $x(\mu_k) \in [0,1]^n$. This gives us (3.53), and we can similarly obtain (3.54) from (3.57). □

Corollary 3.3. *Suppose $\{\bar{\zeta}(\mu_k)\}_{k=1}^{\infty}$ is a set of iterates generated by Algorithm 3.2 with the minimum eigenvalue of the reduced Hessian with respect to $\bar{\zeta}(\mu_k)$ being nonnegative for all k. Let $\bar{\zeta}$ be a limit point of $\{\bar{\zeta}(\mu_k)\}_{k=1}^{\infty}$. Then $\bar{\zeta}$ satisfies the optimality conditions (3.33)–(3.38), and the minimum eigenvalue of the reduced Hessian with respect to $\bar{\zeta}$ is nonnegative.*

Proof. The first part follows from Lemma 3.4 by taking the limits of both sides of (3.49)–(3.54) with respect to the relevant subsequence $\mathcal{K}$ of $\{\bar{\zeta}(\mu_k)\}_{k=1}^{\infty}$. Thus,

$$\lim_{\substack{k\to\infty \\ k\in\mathcal{K}}} \|\bar{\zeta}(\mu_k) - \bar{\zeta}\| = 0. \tag{3.58}$$

Let the null-space matrix of A be denoted by Z. The reduced Hessian with respect to $\bar{\zeta}(\mu_k)$ is then given by

$$\begin{aligned}&Z^{\mathrm{T}}(\nabla^2 f(\bar{x}(\mu_k)) + \bar{X}(\mu_k)^{-1}\bar{\Psi}(\mu_k) + \bar{S}(\mu_k)^{-1}\bar{\Phi}(\mu_k) - 2\Gamma)Z \\ &= Z^{\mathrm{T}}(\nabla^2 f(\bar{x}(\mu_k)) + O(\mu_k)I - 2\Gamma)Z \text{ (using (3.54) and (3.55))} \\ &= Z^{\mathrm{T}}(\nabla^2 f(\bar{x}(\mu_k)) - 2\Gamma)Z + O(\mu_k)I \\ &\to Z^{\mathrm{T}}(\nabla^2 f(\bar{x}) - 2\Gamma)Z,\end{aligned}$$

as $k \to \infty$, $k \in \mathcal{K}$, using (3.58). Since the minimum eigenvalue of the reduced Hessian with respect to $\bar{\zeta}(\mu_k)$ is nonnegative for all k, this implies that the minimum eigenvalue of $Z^{\mathrm{T}}(\nabla^2 f(\bar{x}) - 2\Gamma)Z$ is also nonnegative, by continuity. □

Corollary 3.3 implies that any of the limit points of the set of iterates generated by Algorithm 3.2 will satisfy both the first- and second-order optimality conditions under suitable assumptions on the termination criteria of the algorithm. We could have weakened the assumption in Corollary 3.3 so that the minimum eigenvalue of the reduced Hessian is only required to be $\geq -\theta\mu_k$ for some constant $\theta > 0$ and still get the same result. However, this is not necessary because the positivity of the eigenvalues of the reduced Hessian of the barrier terms will result in the minimum eigenvalue of the entire reduced Hessian matrix being positive.

Also, as $\bar{\zeta}$ may not be unique, different convergent subsequences of the iterates $\{\bar{\zeta}(\mu_k)\}_{k=1}^{\infty}$ may lead to different $\bar{\zeta}$ satisfying the optimality conditions. This agrees with the observation that the iterates of $\{\bar{\zeta}(\mu_k)\}_{k=1}^{\infty}$ only satisfy the optimality conditions of the barrier subproblems approximately and may potentially produce different trajectories.

Chapter 4

Analysis of Linear Systems with Large Diagonal Elements

In the process of obtaining the iterates of the smoothing algorithms, we must solve reduced Hessian systems to determine the direction of descent for each linesearch. The computation involved in solving such systems of equations can be so significant that it warrants a more detailed analysis. Though these systems are linear, the diagonal elements could have varying orders of magnitude and perhaps cause ill-conditioning of the system.

Before analyzing such systems and their perturbation effects, we review briefly the methods used for solving a general square linear system $Ax = b$. Basically, these can be divided into direct and iterative methods. The direct methods include those that perform operations to change the entries of A or to factorize A into a product of matrices, with the resulting modified linear system(s) being easier to solve, and would lead to an exact solution of $Ax = b$. An example is the LU factorization method, where A is factorized into the product of a lower triangular matrix L and an upper triangular matrix U, with the diagonal of either L or U comprised of ones only. The solution of $Ax = b$ then involves solving the simpler systems $Ly = b$ and $Ux = y$. If A is known to be positive definite, it is possible to obtain $A = LU$ with $U = L^{\mathrm{T}}$, and the algorithms to determine the nonzero elements of L are called Cholesky factorization methods.

Iterative methods on the other hand produce a sequence of vectors that approximate the actual solution to $Ax = b$. Usually, the entries of A are unchanged, except with preconditioning (discussed below) or the splitting of A into an appropriate sum of matrices. Also, the operations involved in an iterative method may only include matrix-vector multiplications. This makes iterative methods more attractive than direct methods when A is large, especially if an exact solution to $Ax = b$ is not critical. A widely used iterative method is the conjugate gradient algorithm. It can be described as a class of methods that generate a sequence of mutually conjugate vectors with respect to A, i.e., if this sequence of vectors is denoted by $\{r_n\}_{n=1}^{N}$ for some positive integer N, then $r_i^{\mathrm{T}} A r_j = 0$ for $i \neq j$.

The performance of iterative methods can be improved by preconditioning. For example, if A is symmetric, we would seek a positive definite matrix C such that $C \approx A$ in some sense, where systems involving C can be solved more easily. The iterative method is (conceptually) applied to the "better behaved" system $C^{-1/2}AC^{-1/2}y = C^{-1/2}b$, and x is recovered from $C^{1/2}x = y$.

4.1 Linear Systems with Large Diagonal Elements

Consider a square system of equations $Ax = b$ in which A is symmetric and has some large diagonal elements. This arises for example when we are solving the reduced Hessian system with variables approaching their bounds. In Section 4.2, we discuss how to deal with the reduced Hessian system in further detail. For the time being, we focus on the sensitivity analysis of symmetric linear systems with large diagonal elements and how to compute a solution x to these systems.

4.1.1 Sensitivity Analysis

In general, the accuracy to which $Ax = b$, can be solved deteriorates as the condition number of A increases. For a perturbation $\triangle b$ in b, the perturbation to the solution is bounded according to

$$\frac{\|\triangle x\|}{\|x\|} \leq \kappa(A)\frac{\|\triangle b\|}{\|b\|}, \tag{4.1}$$

where $\kappa(A)$ is the condition number of A (see [GV96]). Likewise, for a perturbation $\triangle A$ in A, we have

$$\frac{\|\triangle x\|}{\|x\|} \leq \kappa(A)\frac{\|\triangle A\|}{\|A\|}. \tag{4.2}$$

A characterization of this analysis is the assumption that the elements of $\triangle A$ are similar, so that a bound in terms of $\|\triangle A\|$ is satisfactory. Consequently, when $\kappa(A)$ is large, the bound for the relative perturbation in x will also be large. A feature of this perturbation analysis is that the perturbation in an individual element of x depends on $\|x\|$ and not on the magnitude of the individual element. Consequently, when the vector x has some elements that are extremely small, their relative perturbations may be extremely large. Note that the sensitivity analysis yields upper bounds. It can be shown that $\triangle b$ and $\triangle A$ exist for which the bounds are tight. However, for some $\triangle b$ and $\triangle A$, the bounds may be unduly pessimistic.

When the ill-conditioning in A is due solely to some large diagonal elements, a more refined analysis is possible. Consider splitting A as a sum of a diagonal matrix D_A and a matrix M_A whose diagonal elements are zero, i.e., $A = D_A + M_A$. Now we would consider the perturbed system as

$$(D_A + M_A + \triangle D_A + \triangle M_A)(x + \triangle x) = b + \triangle b \tag{4.3}$$

and assume that $\triangle M_A$, the perturbation in the off-diagonal elements of A, is small compared to $\|M_A\|$.

What we require is an analysis to derive similar results to (4.1) and (4.2), but in terms of $\|\triangle M_A\|$ and $\|\triangle D_A\|$. A way of achieving this is by scaling the rows and columns of A.

Let us assume that the rows and columns of A are reordered so that A has nonincreasing diagonal elements, i.e., $a_{i,i} \geq a_{i+1,i+1}$ for each i. We can then partition these diagonal elements into two vectors, $\bar{d}$ and $\tilde{d}$, where $\bar{d}$ includes the large diagonals, i.e., $|\bar{d}_j| \gg 1$ for each j. Let $\bar{D} = \mathrm{Diag}(\bar{d})$. Then the matrix

$$D = \begin{bmatrix} \bar{D} & 0 \\ 0 & I \end{bmatrix}$$

is a suitable preconditioner for the matrix A. Define

$$B \equiv D^{-1/2} A D^{-1/2} = \begin{bmatrix} B_1 & C \\ C^{\mathrm{T}} & B_2 \end{bmatrix}.$$

Since $\|\bar{D}\|$ is large, the submatrix B_1 will be "close" to an identity matrix because it has unit diagonal elements and its off-diagonal elements are of order $1/\sqrt{\bar{d}_i \bar{d}_j}$. Thus, we can write B_1 as $I + E$, where $\|E\| \ll 1$. The submatrix C would also have small elements as they would be of order $1/\sqrt{\bar{d}_i}$.

Thus, an alternative definition of x is $x = D^{-1/2} y$, where y is defined by

$$By = b_D, \quad b_D = D^{-\frac{1}{2}} b, \quad B = \begin{bmatrix} I + E & C \\ C^{\mathrm{T}} & B_2 \end{bmatrix}. \tag{4.4}$$

It follows from the structure of B that it is well-conditioned if B_2 is well-conditioned, even if the large diagonals cause A to appear ill-conditioned.

We can now obtain a new sensitivity analysis of (4.3) by analyzing (4.4). Considering the equation $(B + \triangle B)(y + \triangle y) = b_D + \triangle b_D$ and comparing with equation (4.3) gives

$$\triangle B = D^{-1/2} (\triangle D_A + \triangle M_A) D^{-1/2} \tag{4.5}$$

$$\triangle b_D = D^{-1/2} \triangle b. \tag{4.6}$$

After canceling the term $By = b_D$ and ignoring the small term $\triangle B \triangle y$, we obtain $B \triangle y + \triangle B y \approx \triangle b_D$, i.e.,

$$\begin{bmatrix} B_1 & C \\ C^{\mathrm{T}} & B_2 \end{bmatrix} \begin{bmatrix} \triangle \bar{y} \\ \triangle \tilde{y} \end{bmatrix} + \begin{bmatrix} \triangle B_1 & \triangle C \\ \triangle C^{\mathrm{T}} & \triangle B_2 \end{bmatrix} \begin{bmatrix} \bar{y} \\ \tilde{y} \end{bmatrix} \approx \begin{bmatrix} \triangle \bar{b}_D \\ \triangle \tilde{b}_D \end{bmatrix},$$

where appropriate partitions are introduced for vectors y and b_D (as well as other vectors of interest in the subsequent discussion).

The above equations can be rewritten as

$$\begin{aligned}(I+E)\triangle\bar{y} + C\triangle\tilde{y} + \triangle B_1\bar{y} + \triangle C\tilde{y} &\approx \triangle\bar{b}_D \\ C^{\mathrm{T}}\triangle\bar{y} + B_2\triangle\tilde{y} + \triangle C^{\mathrm{T}}\bar{y} + \triangle B_2\tilde{y} &\approx \triangle\tilde{b}_D.\end{aligned}$$

Let $\epsilon = \frac{1}{\|D\|}$; this is a small positive quantity because of the large diagonal elements of D. Since $\bar{b}_D = \bar{D}^{-1/2}\bar{b}$, $\|\bar{b}_D\| \leq \epsilon^{1/2}\|\bar{b}\|$ is also small and $\triangle\bar{b}_D$ even smaller. Also, let $\triangle D = \delta\|D\|$ and $\|\triangle M\| = \theta$, where $\delta, \theta > 0$. Then (4.5) implies that $\|\triangle B\| \leq \|D^{-1/2}\|(\|\triangle D_A\| + \|\triangle M_A\|)\|D^{-1/2}\| \leq \delta + \theta\epsilon$, which is also a small quantity. From $B\triangle y + \triangle By \approx \triangle b_D$, we can conclude that the entries in $\triangle\bar{y}$ must be small too. This means that terms like $E\triangle\bar{y}$, $\triangle B_1\bar{y}$, $C^{\mathrm{T}}\triangle\bar{y}$ and $\triangle C^{\mathrm{T}}\bar{y}$ are small and may be ignored to give the following system:

$$\begin{aligned}\triangle\bar{y} + C\triangle\tilde{y} + \triangle C\tilde{y} &\approx \triangle\bar{b}_D \\ B_2\triangle\tilde{y} + \triangle B_2\tilde{y} &\approx \triangle\tilde{b}_D.\end{aligned}$$

Assuming that B_2 is nonsingular, we may solve for $\triangle\bar{y}$ and $\triangle\tilde{y}$ to obtain:

$$\triangle\bar{y} \approx \triangle\bar{b}_D - \triangle C\tilde{y} - C\triangle\tilde{y} \tag{4.7}$$

$$\triangle\tilde{y} \approx B_2^{-1}(\triangle\tilde{b}_D - \triangle B_2\tilde{y}). \tag{4.8}$$

From (4.8), we have

$$\|\triangle\tilde{y}\| \leq \gamma_1\|B_2^{-1}\|(\|\triangle\tilde{b}_D\| + \|\triangle B_2\|\|\tilde{y}\|),$$

where $\gamma_1 \approx 1$.

Since $\|B_2^{-1}\| = \dfrac{\kappa(B_2)}{\|B_2\|}$ and $\dfrac{1}{\|\tilde{y}\|} \leq \gamma_2\dfrac{\|B_2\|}{\|\tilde{b}_D\|}$ (because $\tilde{b}_D = C^{\mathrm{T}}\bar{y} + B_2\tilde{y} \approx B_2\tilde{y}$), where $\gamma_2 \approx 1$, we get

$$\frac{\|\triangle\tilde{y}\|}{\|\tilde{y}\|} \leq \gamma_1\kappa(B_2)\left(\gamma_2\frac{\|\triangle\tilde{b}_D\|}{\|\tilde{b}_D\|} + \frac{\|\triangle B_2\|}{\|B_2\|}\right). \tag{4.9}$$

Taking note that $\tilde{x} = \tilde{y}$ and $\triangle\tilde{x} = \triangle\tilde{y}$, we conclude that

$$\frac{\|\triangle\tilde{x}\|}{\|\tilde{x}\|} \leq \gamma_1\kappa(B_2)\left(\gamma_2\frac{\|\triangle\tilde{b}_D\|}{\|\tilde{b}_D\|} + \frac{\|\triangle B_2\|}{\|B_2\|}\right), \tag{4.10}$$

which is independent of $\kappa(A)$ and is a much sharper bound than that given by general analysis.

From (4.7), we know that for each i,

$$\triangle\bar{y}_i \approx \triangle(\bar{b}_D)_i - \triangle C_i\tilde{y} - C_i\triangle\tilde{y},$$

where C_i is the ith row of C. Thus,

$$|\triangle\bar{y}_i| \leq \gamma_3(|\triangle(\bar{b}_D)_i| + \|\triangle C_i\|\|\tilde{y}\| + \|C_i\|\|\triangle\tilde{y}\|), \tag{4.11}$$

where $\gamma_3 \approx 1$.

Now, it is sufficient to consider the case $(\bar{b}_D)_i \neq 0$ for each i, since this is usually what we begin with. In fact, we will see later that this case holds for the systems of interest. Even if the magnitude of $(\bar{b}_D)_i$ is very small, we can always deflate the system with appropriate substitutions. Thus, in the subsequent analysis, we shall always assume that $(\bar{b}_D)_i \neq 0$ for each i.

If $\|C\|$ is small enough such that $2\|C_i\|\|\tilde{y}\| \leq |(\bar{b}_D)_i|$ for each i, i.e., $\|C_i\|\|\tilde{y}\| \leq |(\bar{b}_D)_i| - \|C_i\|\|\tilde{y}\|$, then

$$\frac{1}{|(\bar{b}_D)_i| - \|C_i\tilde{y}\|} \leq \frac{1}{|(\bar{b}_D)_i| - \|C_i\|\|\tilde{y}\|} \leq \frac{1}{\|C_i\|\|\tilde{y}\|}. \tag{4.12}$$

Also, $2\|C_i\tilde{y}\| \leq 2\|C_i\|\|\tilde{y}\| \leq |(\bar{b}_D)_i|$ gives $|(\bar{b}_D)_i| \leq 2(|(\bar{b}_D)_i| - \|C_i\tilde{y}\|)$, or equivalently,

$$\frac{1}{|(\bar{b}_D)_i| - \|C_i\tilde{y}\|} \leq \frac{2}{|(\bar{b}_D)_i|}. \tag{4.13}$$

Since $\bar{b}_D = (I+E)\bar{y} + C\tilde{y} \approx \bar{y} + C\tilde{y}$, we have $|\bar{y}_i| \geq \gamma_4(||(\bar{b}_D)_i| - \|C_i\tilde{y}\||) = \gamma_4(|(\bar{b}_D)_i| -$

$\|C_i\tilde{y}\|$), where $\gamma_4 \approx 1$, so that (4.12) and (4.13) implies

$$\frac{1}{|\bar{y}_i|} \leq \min\left\{\frac{1}{\|C_i\|\|\tilde{y}\|}, \frac{2}{\gamma_4|(\bar{b}_D)_i|}\right\}. \tag{4.14}$$

Thus, for $\|C\|$ small enough, we have from (4.11) and (4.14) that

$$\left|\frac{\triangle\bar{y}_i}{\bar{y}_i}\right| \leq 2\gamma_5\left|\frac{\triangle(\bar{b}_D)_i}{(\bar{b}_D)_i}\right| + \gamma_3\frac{\|\triangle C_i\|}{\|C_i\|} + \gamma_3\frac{\|\triangle\tilde{y}\|}{\|\tilde{y}\|}, \tag{4.15}$$

where $\gamma_5 = \frac{\gamma_3}{\gamma_4} \approx 1$. Using (4.9), we get, for each i in which $(\bar{b}_D)_i \neq 0$,

$$\left|\frac{\triangle\bar{y}_i}{\bar{y}_i}\right| \leq 2\gamma_5\left|\frac{\triangle(\bar{b}_D)_i}{(\bar{b}_D)_i}\right| + \gamma_3\frac{\|\triangle C_i\|}{\|C_i\|} + \gamma_6\kappa(B_2)\left(\gamma_2\frac{\|\triangle\tilde{b}_D\|}{\|\tilde{b}_D\|} + \frac{\|\triangle B_2\|}{\|B_2\|}\right), \tag{4.16}$$

where $\gamma_6 = \gamma_1\gamma_3 \approx 1$. From the equation $D^{1/2}x = y$, we have $\bar{D}^{1/2}\bar{x} = \bar{y}$, i.e., $\sqrt{\bar{d}_i}\bar{x}_i = \bar{y}_i$ for each i. A perturbed system for the ith equation is

$$\sqrt{\bar{d}_i + \triangle\bar{d}_i}\,(\bar{x}_i + \triangle\bar{x}_i) = \bar{y}_i + \triangle\bar{y}_i,$$

which on dividing by $\sqrt{\bar{d}_i}$ gives

$$\sqrt{1 + \frac{\triangle\bar{d}_i}{\bar{d}_i}}\,(\bar{x}_i + \triangle\bar{x}_i) = \bar{x}_i + \frac{\triangle\bar{y}_i}{\sqrt{\bar{d}_i}}.$$

Since we would expect $\left|\frac{\triangle\bar{d}_i}{\bar{d}_i}\right|$ to be much smaller than 1, we can approximate $\sqrt{1 + \frac{\triangle\bar{d}_i}{\bar{d}_i}}$ by $1 + \frac{\triangle\bar{d}_i}{2\bar{d}_i}$, so that the perturbed system becomes

$$\left(1 + \frac{\triangle\bar{d}_i}{2\bar{d}_i}\right)(\bar{x}_i + \triangle\bar{x}_i) \approx \bar{x}_i + \frac{\triangle\bar{y}_i}{\sqrt{\bar{d}_i}}.$$

Canceling out the common term $\bar{x}_i$ and ignoring the small quantity $\triangle\bar{d}_i\triangle\bar{x}_i$, we get

$$\triangle\bar{x}_i \approx -\frac{1}{2}\frac{\triangle\bar{d}_i}{\bar{d}_i}\bar{x}_i + \frac{\triangle\bar{y}_i}{\sqrt{\bar{d}_i}}.$$

Therefore,

$$\left|\frac{\triangle\bar{x}_i}{\bar{x}_i}\right| \leq \frac{\gamma_7}{2}\left|\frac{\triangle\bar{d}_i}{\bar{d}_i}\right| + \gamma_7\left|\frac{\triangle\bar{y}_i}{\bar{y}_i}\right|.$$

Using (4.16), we conclude that for each i,

$$\left|\frac{\triangle\bar{x}_i}{\bar{x}_i}\right| \leq \frac{\gamma_7}{2}\left|\frac{\triangle\bar{d}_i}{\bar{d}_i}\right| + 2\gamma_8\left|\frac{\triangle(\bar{b}_D)_i}{(\bar{b}_D)_i}\right| + \gamma_9\frac{\|\triangle C_i\|}{\|C_i\|} + \gamma_{10}\kappa(B_2)\left(\gamma_2\frac{\|\triangle\tilde{b}_D\|}{\|\tilde{b}_D\|} + \frac{\|\triangle B_2\|}{\|B_2\|}\right), \tag{4.17}$$

where $\gamma_8 = \gamma_5\gamma_7 \approx 1$, $\gamma_9 = \gamma_3\gamma_7 \approx 1$ and $\gamma_{10} = \gamma_6\gamma_7 \approx 1$.

As mentioned previously, the first term in the rhs of the inequality in (4.17) is due to perturbation in $\bar{d}_i$ and is not significant when compared to the other terms. Also, from (4.10) and (4.17), we see that the relative perturbation of x now depends on the condition number of B_2.

4.1.2 Computing x

Once y is computed, we can easily compute x from the formula $x = D^{-1/2}y$. In fact, by the definition of D, we get

$$\tilde{x}_i = \tilde{y}_i \;\text{ and }\; \bar{x}_i = \bar{y}_i/\sqrt{\bar{d}_i}$$

for each i.

The analyses done in the previous section can then be used to provide an estimate of the perturbation that may arise in both $\tilde{x}$ and $\bar{x}$ following perturbation in the relevant submatrices of A.

4.2 Reduced Hessian Systems

We may now consider the case when the matrix A discussed in the previous sections is the reduced Hessian. In fact, we need to solve the reduced Hessian system

$$(Z^{\mathrm{T}}HZ)u = -Z^{\mathrm{T}}g, \tag{4.18}$$

where H is the sum of the Hessian of the objective function $\nabla^2 F$ and the diagonal term D arising from the Hessian of the barrier terms, and Z is a full rank matrix whose columns span the null space of A.

A consequence of a variable being close to a bound (and this is inevitable) is that the corresponding diagonal element of H is large. Unfortunately, while H may be ill-conditioned in this way, that is not true of $Z^{\mathrm{T}}HZ$, which may be ill-conditioned but is likely to have large off-diagonal elements. It is not easy to compare the condition number of H with that of $Z^{\mathrm{T}}HZ$. For example, if H is singular and has rank $n-1$, $Z^{\mathrm{T}}HZ$ may have full rank and be well-conditioned. But if H has $n-m$ or less large diagonal elements, then $Z^{\mathrm{T}}HZ$ is likely to be ill-conditioned with condition number similar to that of H.

Example 4.1. Consider the Hessian matrix $H = \begin{bmatrix} 10^6 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and the linear constraint matrix $A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$. A null-space matrix of A is given by $Z = \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Then $Z^{\mathrm{T}}HZ = \begin{bmatrix} 10^6+1 & 10^6 \\ 10^6 & 10^6+1 \end{bmatrix}$ and $\kappa(Z^{\mathrm{T}}HZ) \approx 2 \times 10^6$, which is of the same order as $\kappa(H) = 10^6$. If $H = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ with Z unchanged, then $Z^{\mathrm{T}}HZ = I$.

In the next section, we show that if H has large diagonal elements, Z may be chosen such that the only large elements of $Z^{\mathrm{T}}HZ$ are on the diagonal.

4.2.1 Transformation to a Linear System with Large Diagonal Elements

As discussed in Section 4.1 for the matrix A, we also split the reduced Hessian $H_Z \equiv Z^{\mathrm{T}}HZ$ into the sum of a diagonal matrix and another matrix with zero diagonal elements. After performing a permutation of variables, we can write

$$H_Z = D_Z + M_Z,$$

where $\mathrm{diag}(M) = 0$ and $D_{i,i} \geq D_{i+1,i+1}$ for each i. We can further partition D_Z into two diagonal matrices D_1 and D_2, i.e.,

$$D_Z = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix},$$

where the elements of D_1 are large and those of D_2 are not. Because of the barrier function in the objective, this partition tends to be reflective of which variables are close to their bounds. However, there is no necessity to know or keep track of this partition throughout the algorithm.

When the size of the reduced Hessian is less than the number of large elements of the Hessian, the reduced Hessian is in general not ill-conditioned. Indeed it may be diagonally dominant. Consequently, it is still worthwhile to apply a diagonal preconditioner, such as

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & I \end{bmatrix}$$

discussed extensively in the previous section.

If we used the above preconditioning matrices, the sensitivity analysis in Section 4.1.1 would apply to the reduced Hessian. Here, we find that for perturbations $\triangle Z$ and $\triangle H$ in Z and H and ignoring second-order terms, we have

$$\begin{aligned} \triangle D_Z + \triangle M_Z &= 2Z^{\mathrm{T}}H\triangle Z + Z^{\mathrm{T}}\triangle HZ \\ \triangle b &= \triangle Z^{\mathrm{T}}g + Z^{\mathrm{T}}\triangle g. \end{aligned}$$

For cases where Z is of a special structure and can be determined exactly, $\triangle Z$ can be set to 0. Also, we can estimate the values of $\triangle g$ and $\triangle H$ based on the information obtained from the derivatives of the objective function.

4.2.2 Permutation of Variables

An issue that remains unaddressed is the effect of permuting variables on the reduced system. To analyze the effect, we can apply the variable reduction technique, i.e., we first partition the matrix A into $[B \;\; N]$, where B is nonsingular. A natural form of Z is $\begin{bmatrix} -B^{-1}N \\ I \end{bmatrix}$. For example, in the case of an optimization problem with the "assignment" constraints

$$\sum_{j=1}^{n} x_{ij} = 1$$

for each i, A is usually of the form $[e_1, \dots, e_1, e_2, \dots, e_2, \dots, e_m, \dots, e_m]$, where e_i is the ith column of I_m. We can then choose B to be the identity matrix for this problem with "assignment" constraints by picking e_i for $i = 1, \dots, m$ from the relevant columns of A. Then

$$Z = \begin{bmatrix} -I^{-1}N \\ I \end{bmatrix} = \begin{bmatrix} -e & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & -e \\ I & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & I \end{bmatrix},$$

which, as we shall see, differs from the Z we use in Chapter 6 only by a permutation.

Consider permuting some columns of A so that we obtain the new matrix $\tilde{A} = [\tilde{B} \;\; \tilde{N}] = AP$ for some permutation matrix P. This permutation changes the Hessian H to $\tilde{H} = P^{\mathrm{T}}HP$. Letting $\tilde{Z} = \begin{bmatrix} -\tilde{B}^{-1}\tilde{N} \\ I \end{bmatrix}$ be the new null-space matrix, we find that $A(P\tilde{Z}) = (AP)\tilde{Z} = \tilde{A}\tilde{Z} = 0$, i.e., the columns of $P\tilde{Z}$ are in the null space of A. Since the columns of Z form a basis of the null space of A, this means that $P\tilde{Z} = ZQ$ for some nonsingular matrix Q. Note that Q can be obtained by using the formula

$Q = IQ = ([0 \;\; I]Z)Q = [0 \;\; I](ZQ) = [0 \;\; I]P\tilde{Z}$. Moreover,

$$\begin{aligned} Q^{\mathrm{T}}Z^{\mathrm{T}}HZQ &= (ZQ)^{\mathrm{T}}H(ZQ) \\ &= (P\tilde{Z})^{\mathrm{T}}HP\tilde{Z} \\ &= \tilde{Z}^{\mathrm{T}}P^{\mathrm{T}}HP\tilde{Z} \\ &= \tilde{Z}^{\mathrm{T}}\tilde{H}\tilde{Z}. \end{aligned}$$

Pre-multiplying the original reduced system $Z^{\mathrm{T}}HZy = -Z^{\mathrm{T}}g$ by Q^{T}, we find that $Q^{\mathrm{T}}Z^{\mathrm{T}}HZy = -Q^{\mathrm{T}}Z^{\mathrm{T}}g = (ZQ)^{\mathrm{T}}g = -\tilde{Z}^{\mathrm{T}}P^{\mathrm{T}}g$, and the above identity leads us to the new reduced system $\tilde{Z}^{\mathrm{T}}\tilde{H}\tilde{Z}\tilde{y} = -\tilde{Z}^{\mathrm{T}}\tilde{g}$, where $y = Q\tilde{y}$ and $g = P\tilde{g}$.

4.3 Computing the Solution of $Bx = b_D$

As discussed previously, we wish to obtain a solution of $Ax = b$ by first computing a solution to the equation $By = b_D$, i.e.,

$$\begin{bmatrix} B_1 & C \\ C^{\mathrm{T}} & B_2 \end{bmatrix} \begin{bmatrix} \bar{y} \\ \tilde{y} \end{bmatrix} = \begin{bmatrix} \bar{b}_D \\ \tilde{b}_D \end{bmatrix}. \tag{4.19}$$

To solve (4.19), we can apply either direct methods such as the modified Cholesky algorithms (see e.g., [GM74, FGM95]), or iterative methods such as the conjugate gradient method. Here, the special structure of the matrices involved may make iterative methods more attractive and we discuss them in greater detail in the following sections.

4.3.1 Conjugate Gradient Method

Conjugate gradient (CG) methods tend to be effective in solving large-scale systems if they are well-conditioned. Thus, it would be good to use D in the previous section to precondition the system $Ax = b$. We can certainly compute the individual entries of B and then apply the CG method directly to the equation $By = b_D$, but in general, a preconditioner D would be applied as in [GV96, page 354] using products with A

and solves with D (not $D^{1/2}$).

Certainly, there are some properties of $\bar{x}$ and $\bar{y}$ that make them distinct from $\tilde{x}$ and $\tilde{y}$ (here, the partition of any vector z into $\begin{bmatrix} \bar{z} \\ \tilde{z} \end{bmatrix}$ is as described in Section 4.1.1). For example, we know that $\bar{y} \approx 0$, and we get $\bar{x}$ with even smaller entries than $\bar{y}$, since $\bar{x} = \bar{D}^{-1/2}\bar{y}$ and the diagonal elements in $\bar{D}$ are large. Thus, it makes sense to split the solving of (4.19) into two, i.e., one involving $\bar{y}$ and the other involving $\tilde{y}$, and then apply algorithms such as the CG method to both separately. For example, the bottom part of (4.19) can be written as $B_2\tilde{y} \approx \tilde{b}_D$ because $C^{\mathrm{T}}\bar{y} \approx 0$. Thus, the CG method can be applied to obtain a solution $\tilde{y}$ that is substituted into the top part of (4.19) to give $B_1\bar{y} = \bar{b}_D - C\tilde{y}$. The CG method can then be applied again to this system to obtain an approximate $\bar{y}$. The fact that $\bar{y} \approx 0$ implies it is unnecessary to solve for $\bar{y}$ to any great accuracy. Different criteria may have to be applied to terminate CG on the separate systems.

4.3.2 The Schur Complement Method

The splitting into two systems could also be done by using a Schur complement approach with the above partition. Here, we reduce system (4.19) to the smaller subsystem

$$S\tilde{y} = b_S,$$

where $S = B_2 - C^{\mathrm{T}}B_1^{-1}C$ is the Schur complement of B with respect to B_1, and $b_S = \tilde{b}_D - C^{\mathrm{T}}B_1^{-1}\bar{b}_D$. After solving for $\tilde{y}$, we can obtain $\bar{y}$ by solving

$$B_1\bar{y} = \bar{b} - C\tilde{y}.$$

The subsystem $S\tilde{y} = b_S$ may be easier to solve as S could be less ill-conditioned than B and we can apply the conjugate gradient algorithm to this subsystem. Also, in the situation that $B_1 = I + E$ where $\|E\| \ll 1$, we can approximate B_1^{-1} by $I - E$ or $I - E + \frac{1}{2}E^2$, and then apply iterative refinement procedures to correct the errors. This is especially true for the reduced Hessian matrix when the majority of its diagonal elements are large in magnitude, resulting in B_2 being a smaller submatrix than B_1, i.e., $S\tilde{y} = b_S$ is then a smaller system to solve than the reduced Hessian system, and yet B_1 is sufficiently close to the identity matrix to approximate its inverse with $I - E$.

4.3.3 Block Gauss-Seidel Method

Another approach to splitting the system is to use the block Gauss-Seidel method. We would first write the matrix B_2 as $M_1 + M_2$, where M_1 is diagonal and diag(M_2) = 0. Also, we have discussed earlier that B_1 can be written as $I + E$. The block Gauss-Seidel method would then entail solving the following equations iteratively:

$$\begin{aligned} \bar{x}^{(k)} &= \bar{b}_D - E\bar{x}^{(k-1)} - C\tilde{x}^{(k-1)} \\ M_1\tilde{x}^{(k)} &= \tilde{b}_D - C^{\mathrm{T}}\bar{x}^{(k)} - M_2\tilde{x}^{(k-1)}, \end{aligned}$$

where $\bar{x}^{(0)} = \bar{b}_D$ and $\tilde{x}^{(0)} = \tilde{b}_D$.

In the event that $\|E\|$ is very small, especially when we are in the later major iterations of the reduced Hessian system, we can even set E to be the zero matrix in the above algorithm.

Chapter 5

Algorithm Implementation and Details

In this chapter, we discuss certain issues encountered during implementation of the primal logarithmic barrier smoothing algorithm. These include the details of obtaining the initial iterate for the smoothing algorithm, the search direction, and the steplength used for generating new iterates.

5.1 Initial Iterate for Smoothing Algorithm

In a logarithmic barrier algorithm, it is essential that an interior feasible point be used as the initial iterate. However, not every interior feasible point is a good initial iterate. If the initial penalty parameter γ_0 is zero (or small as it usually is) and the initial barrier parameter μ_0 is large, there is no danger in picking any interior feasible point to be the initial iterate. Nevertheless, it is inefficient to have the initial iterate close to any of the constraint boundaries because the minimizer of the initial barrier subproblem is likely to be far from the constraint boundary, and extra work would be needed to converge to this minimizer. Thus, we need to choose an initial iterate that is compatible with the penalty and barrier parameters, and preferably, an interior point that is "neutral" to all the constraints in the feasible region.

As an example, if we have the feasible region $\mathcal{S}$ without any linear constraints

$Ax = b$, we would expect the "neutral" interior point to be $\frac{1}{2}e$ because such a point will be equally far from the boundaries $x_j = 0$ or $x_j = 1$. However, if we have a linear constraint $e^{\mathrm{T}}x = 1$, then the "neutral point" ought to be $\frac{1}{n}e$ for reasons we now describe.

5.1.1 Analytic Center

A possible choice of such a "neutral" point is one whereby the product of all the slack variables for the constraints $\{x : 0 \leq x \leq e\}$ is maximized, subject to feasibility constraints:

$$\begin{array}{ll} \text{Maximize} & \prod_{j=1}^{n} x_j(1 - x_j) \\ \text{subject to} & Ax = b \\ & 0 < x < e. \end{array} \tag{P5.1}$$

This point is known in the literature as the analytic center of the feasible region $\mathcal{S}$ (see e.g., [Ye97, Chapter 2]).

Problem (P5.1) can be converted into the following equivalent minimization problem by a logarithmic transformation of the objective function, where the constraints $0 < x < e$ are implicitly assumed:

$$\begin{array}{ll} \text{Minimize} & -\sum_{j=1}^{n} [\ln x_j + \ln(1 - x_j)] \\ \text{subject to} & Ax = b. \end{array} \tag{P5.2}$$

Note that if A has full rank, then problem (P5.2) has a unique optimal solution, by strict convexity of the objective function and the convexity of the feasible region. In fact, this unique optimal solution, or equivalently, the analytic center of $\mathcal{S}$, can be obtained by solving the optimality conditions

$$\begin{array}{ll} A^{\mathrm{T}}\lambda & = x_g \\ Ax & = b, \end{array} \tag{5.1}$$

where $(x_g)_i = \frac{1}{x_i} - \frac{1}{1-x_i}$ for all i and λ is the Lagrange multiplier of the constraint $Ax = b$.

Example 5.1. Consider problem (P5.2) when there are no linear equality constraints. Then it is clear that the optimal solution of (P5.2) is $\frac{1}{2}e$. On the other hand, suppose $A = e^{\mathrm{T}}$ and $b = 1$. It can be easily verified that the vector $\frac{1}{n}e$ satisfies optimality conditions (5.1) for this matrix A, and hence is the unique optimal solution to (P5.2).

5.1.2 Relationship with Trajectory of Smoothing Algorithm

The previous section indicates that the analytic center of the feasible region $\mathcal{S}$ appears to be a reasonable point to be the initial iterate of the smoothing algorithm. There are certainly other candidates for the initial iterate, such as the center of the ellipsoid that can be inscribed in the feasible region $\mathcal{S}$ with the maximum volume. However, we now show the relationship of the analytic center to the trajectory of the smoothing algorithm with respect to the smoothing parameter μ.

Theorem 5.1. *Suppose $\{x : Ax = b,\ 0 < x < e\} \neq \emptyset$, and $x^*(\mu),\ x^*$ are the solutions to problems (P3.3) and (P5.2) respectively. Then $\lim\limits_{\mu\to\infty} x^*(\mu) = x^*$.*

Proof. Let $\{\mu_k\}_{k=1}^{\infty}$ be any sequence such that $\lim\limits_{k\to\infty} \mu_k = \infty$, and let $\bar{x}$ be a limit point of the sequence $\{x^*(\mu_k)\}_{k=1}^{\infty}$. There must be at least one such limit point because $x^*(\mu)$ is in the compact set $[0,1]^n$ for all $\mu > 0$.

Define $\Phi(x) = -\sum\limits_{j=1}^{n}[\ln x_j + \ln(1-x_j)]$. Observe that $x^*(\mu)$ is the solution to the problem

$$\begin{array}{ll} \text{Minimize} & \frac{1}{1+\mu}f(x) + \frac{\mu}{1+\mu}\Phi(\mu) \\ \text{subject to} & Ax = b, \end{array} \tag{P5.3}$$

whose objective function differs from that of (P3.3) by the multiplicative constant $\frac{1}{1+\mu}$. Thus

$$\frac{1}{1+\mu}f(x^*(\mu)) + \frac{\mu}{1+\mu}\Phi(x^*(\mu)) \leq \frac{1}{1+\mu}f(x^*) + \frac{\mu}{1+\mu}\Phi(x^*). \tag{5.2}$$

Since f is continuous on $[0,1]^n$, there exists a positive constant $L > 0$ such that $|f(x)| \leq L$ for all $x \in [0,1]^n$. Let $\epsilon > 0$ and define $M > \max\{\frac{L}{\epsilon} - 1, 1\}$. Then for all k such that $\mu_k > M$, we have $\frac{L}{1+\mu_k} < \epsilon$, and hence

$$\begin{aligned}
-\epsilon + \frac{\mu_k}{1+\mu_k}\Phi(x^*(\mu_k)) &< -\frac{L}{1+\mu_k} + \frac{\mu_k}{1+\mu_k}\Phi(x^*(\mu_k)) \\
&\leq \frac{1}{1+\mu_k} f(x^*(\mu_k)) + \frac{\mu_k}{1+\mu_k}\Phi(x^*(\mu_k)) \\
&\leq \frac{1}{1+\mu_k} f(x^*) + \frac{\mu_k}{1+\mu_k}\Phi(x^*) \text{ (from (5.2))} \\
&\leq \frac{L}{1+\mu_k} + \frac{\mu_k}{1+\mu_k}\Phi(x^*) \\
&< \epsilon + \frac{\mu_k}{1+\mu_k}\Phi(x^*).
\end{aligned}$$

Taking limits as $k \to \infty$, we have

$$-\epsilon + \lim_{k\to\infty} \Phi(x^*(\mu_k)) \leq \epsilon + \Phi(x^*).$$

Since ϵ is an arbitrary positive number, we conclude that $\lim_{k\to\infty} \Phi(x^*(\mu_k)) \leq \Phi(x^*)$. Now, $\Phi(x^*) \leq \Phi(x^*(\mu))$ for all $\mu > 0$ by optimality of x^* in (P5.2). This implies that $\Phi(x^*) \leq \lim_{k\to\infty} \Phi(x^*(\mu_k))$, and thus

$$\lim_{k\to\infty} \Phi(x^*(\mu_k)) = \Phi(x^*). \tag{5.3}$$

Since $x^* \in (0,1)^n$, this implies that $\lim_{k\to\infty} \Phi(x^*(\mu_k))$ must be bounded. Therefore, by continuity of Φ on $(0,1)^n$ and (5.3),

$$\Phi(\bar{x}) = \Phi(x^*).$$

Thus, $\bar{x} = x^*$ since x^* is the unique minimizer to (P5.2). This shows that any limit point of $x^*(\mu_k)$ is equal to x^*, so that $\lim_{k\to\infty} x^*(\mu_k) = x^*$, and we can conclude that $\lim_{\mu\to\infty} x^*(\mu) = x^*$. □

We see that the analytic center of the feasible region is an ideal starting point for the smoothing algorithm because it allows the iterates to be on the desired trajectory $\{x^*(\mu)\}$ at an early stage of the algorithm. Finding the exact analytic center of the feasible region $\mathcal{S}$ can be expensive and there are many efforts to deal with this issue (see e.g., [GTP98], [Ye97, Chapter 3]). However, it is not crucial that we obtain the exact analytic center of $\mathcal{S}$ and it suffices to start the smoothing algorithm with a large barrier parameter μ and to approximate the analytic center from the iterate obtained in the minor iterations of determining a vector satisfying the optimality conditions for that value of μ. Moreover, for certain types of feasible region $\mathcal{S}$, the analytical center is known, as illustrated in Example 5.1.

5.2 Using the Conjugate Gradient Method

Some of the broad issues of applying the CG method to the reduced Hessian system have been discussed in earlier chapters. We are interested here in the precise algorithms used, the termination criteria involved, and how to obtain the search directions.

The CG method used in the smoothing algorithm is shown on the next page. It is different from standard CG methods in the sense that it can handle systems that are indefinite, and has different termination criteria. It involves an adaptation of the CG methods used in a truncated Newton method [DS83].

5.2.1 Computing Z and Associated Operations

In Section 4.2.2, we mentioned how to obtain the null-space matrix Z for the full-rank matrix $A \in \mathbb{R}^{m\times n}$ using the variable-reduction technique. Here, we discuss how we perform such a partitioning of A into $[B \;\; N]$, where B is nonsingular. It is worth mentioning that a QR factorization could also be used to obtain Z. For example, if we obtain $A^{\mathrm{T}} = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$ using Householder transformations, where Q is an orthogonal matrix and R is a nonsingular upper triangular matrix, then $AQ = (Q^{\mathrm{T}}A^{\mathrm{T}})^{\mathrm{T}} = [R^{\mathrm{T}} \; 0]$ and the last $n-m$ columns of Q would be a null-space matrix for A.

Algorithm 5.1: CG method for $Z^{\mathrm{T}}HZu = -Z^{\mathrm{T}}g$.

Set ϵ = tolerance for indicating nonpositive curvature,
δ = tolerance for residual error,
N = iteration limit,
$x_0 = 0$,
$r_0 = -Z^{\mathrm{T}}g$,
$p_0 = r_0$.
for $l = 0, 1, \ldots, N$
  if $p_l^{\mathrm{T}} Z^{\mathrm{T}} H Z p_l \le \epsilon \|p_l\|^2$
    if $l = 0$
      Set $d_1 = p_0$, $d_2 = 0$.
    else
      Set $d_1 = x_l$, $d_2 = p_l$.
    endif
  elseif $\|r_l\| \le \delta \|Z^{\mathrm{T}}g\|$
    Set $l = N$, $d_1 = x_l$, $d_2 = 0$.
  else
    Set $\alpha_l = \frac{\|r_l\|^2}{p_l^{\mathrm{T}} Z^{\mathrm{T}} H Z p_l}$,
      $x_{l+1} = x_l + \alpha_l p_l$,
      $r_{l+1} = r_l - \alpha_l Z^{\mathrm{T}} H Z p_l$,
      $\beta_l = \frac{\|r_{l+1}\|^2}{\|r_l\|^2}$,
      $p_{l+1} = r_{l+1} + \beta_l p_l$.
  endif
end for
Output: d_1, d_2

When forming the partition $[B \quad N]$, we need to choose a partition of the Hessian matrix H based on its diagonal d_H. For example, if d_H has elements with approximately the same order of magnitude, then we can pick any linearly independent subset of columns of A to form B, with the remaining columns of A for forming N. In the situation when d_H has elements with varying orders of magnitude, we can partition the index set of d_H into two sets D_1 and D_2 as in Section 4.2.1, such that $i \in D_1$ would imply that $(d_H)_i$ is large, while $i \in D_2$ would imply that $(d_H)_i$ is small. In the case that $|D_1| < |D_2|$, we can pick any largest set of indices from D_2 such that the corresponding columns of A are linearly independent. If there is an insufficient number of columns chosen to form B, we can pick the rest from D_1 such that we have the smallest magnitudes of $(d_H)_i$ and the corresponding columns of A are linearly independent. The rest of the unpicked columns will then form N. Likewise when $|D_1| \geq |D_2|$, we will pick any largest linearly independent set of columns with indices from D_1 to form B, using indices of D_2 with the largest magnitudes of $(d_H)_i$ if necessary, and the rest of the unpicked columns will form N.

With the appropriate partition for A, it may not be necessary to compute Z for matrix-vector multiplications of the type $Z^{\mathrm{T}}u$ or Zu. For example, $Z^{\mathrm{T}}u$ can be obtained as follows:

$$Z^{\mathrm{T}}u = [-N^{\mathrm{T}}B^{-\mathrm{T}} \quad I] \begin{bmatrix} u_B \\ u_N \end{bmatrix} = u_N - N^{\mathrm{T}}w,$$

where $B^{\mathrm{T}}w = u_B$. An LU factorization of B can usually exploit any special structure that B might have. Even if Z is computed via QR factorization, it is also possible to perform matrix-vector multiplications involving Z efficiently, without forming Z explicitly.

5.2.2 Obtaining the Search Direction

In Algorithm 5.1, after we obtain d_1 and d_2, the search direction can be obtained as a suitable linear combination of the two directions Zd_1 and Zd_2. There are many ways of performing this linear combination. The search direction that is used for the smoothing algorithm is

$$d = Zd_1 + Zd_2. \tag{5.4}$$

However, we would also like to improve the quality of this search direction, especially the direction of negative curvature Zd_2, because we know from [DS83] that d_1 satisfies conditions of sufficient descent: $g^{\mathrm{T}} Z d_1 \leq -K_1 \|Z^{\mathrm{T}} g\|^2$ and $\|d_1\| \leq K_2 \|Z^{\mathrm{T}} g\|$, where K_1, $K_2 > 0$. If we find in Algorithm 5.1 that there is a p_l such that $p_l^{\mathrm{T}} Z^{\mathrm{T}} H Z p_l \leq \epsilon \|p_l\|^2$, we would like to enhance the direction of $d_2 = p_l$ so that it is a good direction of negative curvature, i.e., we would like a direction of negative curvature $\bar{d}_2$ that satisfies

$$\bar{d}_2^{\mathrm{T}} Z^{\mathrm{T}} H Z \bar{d}_2 \leq \theta \lambda_{\min}(Z^{\mathrm{T}} H Z), \tag{5.5}$$

where $\theta \in (0, 1]$ and $\|\bar{d}_2\| = 1$. Algorithms that take a poor direction of negative curvature d_2 to obtain $\bar{d}_2$ satisfying (5.4) are discussed at length in [Bom99]. Two such adapted algorithms are shown next, namely the modified Lanczos algorithm and the coordinate search. The former algorithm makes use of a tridiagonal matrix close to $Z^{\mathrm{T}} H Z$ in order to get a good approximation to the eigenvector corresponding to $\lambda_{\min}(Z^{\mathrm{T}} H Z)$, while the latter is based on the Rayleigh criterion, i.e., minimizing the Rayleigh quotient:

$$\underset{x \neq 0}{\text{Minimize}} \; \frac{x^{\mathrm{T}} Z^{\mathrm{T}} H Z x}{x^{\mathrm{T}} x}. \tag{P5.4}$$

Algorithm 5.2: Enhancing given direction of negative curvature d_2 of $Z^{\mathrm{T}} H Z$ by modified Lanczos method.

Set $x_0 = d_2$,
$q_0 = 0$,
$\beta_0 = \|x_0\|^2$.
for $l = 1, 2, \ldots, n$
Set $q_l = x_{l-1}/\beta_{l-1}$,
$x_l = Z^{\mathrm{T}} H Z q_l - \beta_{l-1} q_{l-1}$,
$\alpha_l = q_l^{\mathrm{T}} x_l$,
$x_l = x_l - \alpha_l q_l$,
$\beta_l = \|x_l\|$.
end for

Algorithm 5.3: Enhancing given direction of negative curvature d_2 of $A = Z^{\mathrm{T}}HZ$ by coordinate search.

Set $x = d_2$,
$\mu = x^{\mathrm{T}}Ax$,
$\beta = \|x\|^2$.
for $l = n, n-1, \ldots, 2, 1, 2, \ldots, n-2, n-1, \ldots$
Set $\gamma = x^{\mathrm{T}}A_{\bullet,l}$.
Solve $(A_{ll}x_l - \gamma)\alpha^2 + (A_{ll}\beta - \mu)\alpha + (\gamma\beta - \mu x_l) = 0$ to obtain α_1, α_2.

$$
\begin{aligned}
\text{Set } \mu_i &= \mu + 2\alpha_i x_l + \alpha_i^2 A_{ll},\ i = 1, 2,\\
\beta_i &= \beta + 2\alpha_i x_l + \alpha_i^2,\ i = 1, 2,\\
I &= \arg\min\ \{\mu_i/\beta_i \mid i = 1, 2\},\\
\mu &= \mu + 2\alpha_I x_l + \alpha_I^2 A_{ll},\\
\beta &= \beta + 2\alpha_I x_l + \alpha_I^2,\\
x_l &= x_l + \alpha_I.
\end{aligned}
$$

end for

5.3 Linesearch

Here we shall assume that the search direction d is given by (5.5). It is then essential to find a good steplength α to determine the next iterate $x + \alpha d$, with respect to the objective function $F(x) = f(x) + \mu\Phi(x)$. There are certain conditions that α must satisfy in order to be a reasonable steplength. The foremost quantity required is the maximum steplength achievable in the feasible region, $\alpha_{\max}$. This is because a unit steplength from the iterate x may make the next iterate $x + \alpha d \notin \mathcal{S}$, where $\mathcal{S} = \{x : Ax = b,\ 0 < x < e\}$ is the feasible region.

It is also necessary to ensure that certain sufficient decrease conditions are satisfied by the steplength, such as the following set of conditions (see [FM93]):

$$
\begin{aligned}
&F(x + \alpha d) \le F(x) + \mu\alpha\nabla F(x + \alpha d)^{\mathrm{T}}d \qquad \text{and}\\
&|\nabla F(x + \alpha d)^{\mathrm{T}}d| \le \eta|\nabla F(x)^{\mathrm{T}}d| \quad \text{or} \quad \alpha = \alpha_{\max},
\end{aligned}
\tag{5.6}
$$

if $d^{\mathrm{T}}\nabla^2 F(x)d \geq 0$, and

$$\begin{array}{ll} F(x+\alpha d) \leq F(x) + \mu\left(\alpha \nabla F(x+\alpha d)^{\mathrm{T}} d + \frac{1}{2}\alpha^2 d^{\mathrm{T}}\nabla^2 F(x) d\right) & \text{and} \\ |\nabla F(x+\alpha d)^{\mathrm{T}} d| \leq \eta |\nabla F(x)^{\mathrm{T}} d + \alpha d^{\mathrm{T}} \nabla^2 F(x) d| \ \text{ or } \ \alpha = \alpha_{\max}, & \end{array} \tag{5.7}$$

if $d^{\mathrm{T}}\nabla^2 F(x)d < 0$, where $\mu \in (0, \frac{1}{2})$ and $\eta \in [\mu, 1)$.

There are steplength procedures to find an α that satisfies the above conditions (see [GMSW79, MS79]). There are also special procedures, such as interpolation methods (see [MW94]) that can deal with the barrier terms in the objective or merit function used in the linesearch more effectively. However, we wish to determine the quality of the solutions obtained by the smoothing algorithm, independent of the linesearch procedures. Thus, only a simple backtracking algorithm is used for the smoothing algorithm and this amounts to obtaining an α that satisfies (5.6) only. This has proved adequate for conducting numerical tests on the algorithm.

5.4 Parameter Initialization and Update

The initialization and update of the parameters used in the smoothing algorithm, especially the barrier parameter μ, can influence the quality of the solution obtained. We have already seen that the analytic center is an ideal iterate to begin with and in the event of computational difficulty of obtaining the exact analytic center, the initialization of μ to a large value would produce an iterate that has a good chance of being approximately the analytic center. Also, it is helpful to reduce μ gradually, as we would like the iterates to follow a trajectory from the starting point to a good local or even global minimizer of the original problem.

To illustrate better, consider the problem

$$\begin{array}{rl} \text{Minimize} & -(x_1-1)^2 - (x_2-1)^2 - 0.1(x_1+2x_2-2) \\ \text{subject to} & x_j \in \{0,2\}, j = 1,2, \end{array} \tag{P5.5}$$

which, after the introduction of the smoothing function, becomes the transformed problem on the next page:

Algorithm 5.4: Backtracking algorithm for determining steplength.

Set x = current iterate of smoothing algorithm,
d = search direction for x
N = iteration limit for backtracking
θ_1 = fraction of steplength not exceeding boundary
θ_2 = reduction ratio in backtracking.
for $k = 1, 2, \dots, n$
if $d_k > 0$
Set $\beta_k = \frac{1-x_k}{d_k}$.
elseif $d_k < 0$
Set $\beta_k = \frac{-x_k}{d_k}$.
else
Set $\beta_k = \infty$.
end if
end for
Set $\alpha = \theta_1 \min\{\min_k \beta_k, 1\}$.
for $l = 1, 2, \dots, N$
if $F(x + \alpha d) > F(x) + \mu\alpha\nabla F(x + \alpha d)^{\mathrm{T}} d$
Set $\alpha = \theta_2\alpha$.
end if
end for

$$\begin{aligned} \text{Minimize} \quad & -(x_1 - 1)^2 - (x_2 - 1)^2 - 0.1(x_1 + 2x_2 - 2) \\ & -\mu(\ln x_1 + \ln(2 - x_1) + \ln x_2 + \ln(2 - x_2)). \end{aligned} \tag{P5.6}$$

It can easily be verified that the global (and hence local) minimum to (P5.5) is given by $(x_1^*, x_2^*) = (2, 2)$. However, there are also three other local minima given by (0,0), (0,2) and (2,0).

Figure 5.1 shows the contour graph of the objective function in (P5.5), as well as different trajectories arising from the smoothing algorithm. It can be seen that initializing μ with a large value in this case leads to a trajectory of iterates converging to the global minimum. However, if the initial μ had been a smaller value, there is a risk that a trajectory leading to a local (but not global) minimum would be obtained. Also, a large reduction of μ could also cause the path of iterates to switch from one trajectory to another.

We see that it helps to be conservative in the choice of the initial parameters as well as the updating of the parameters. Though more computational time may be needed, we have a better chance of obtaining a trajectory that leads to a good quality solution to the original problem. Typical ranges of the parameters used in the smoothing algorithm are summarized below.

Table 5.1: Typical ranges of parameters of smoothing algorithm.

Parameter	Range
Initial barrier parameter, μ_0	$10 - 1000$
Final barrier parameter, μ_N	$10^{-4} - 1$
Barrier parameter reduction, θ_μ	$0.1 - 0.999$
Initial penalty parameter, γ_0	$0 - 10$
Final penalty parameter, γ_N	$10 - 10^5$
Penalty parameter increment, θ_γ	$0.1 - 0.9$

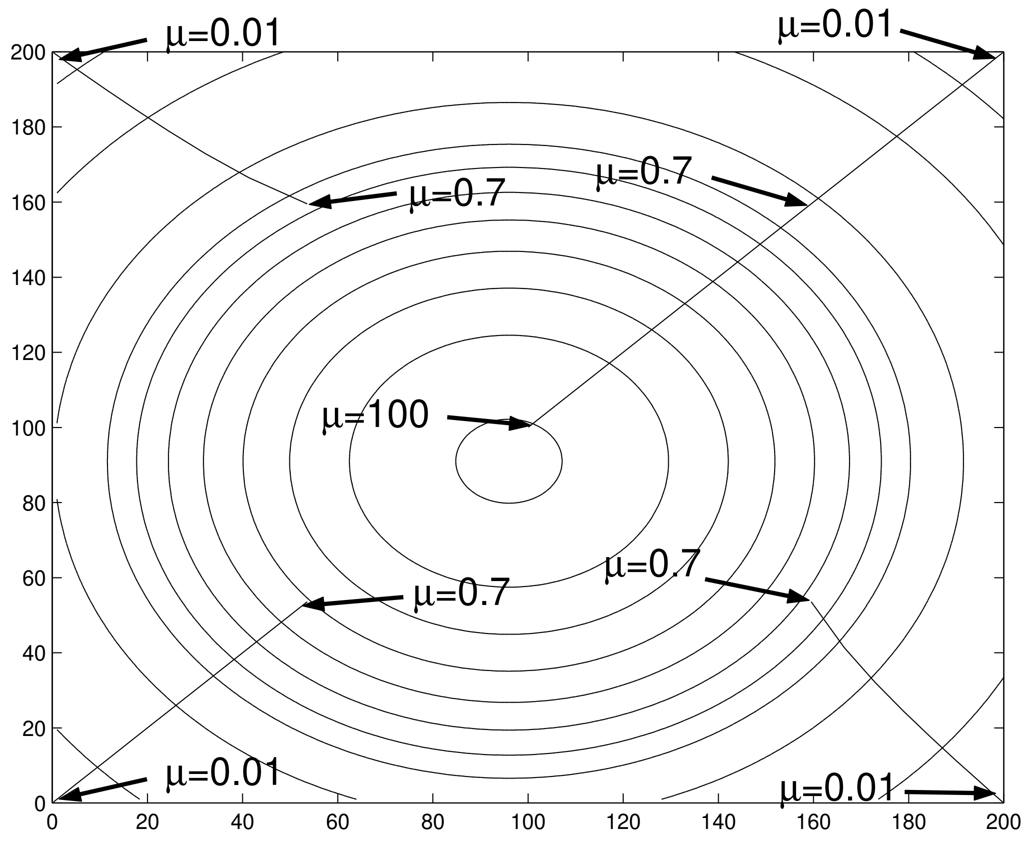

Figure 5.1: Trajectories taken by iterates of smoothing algorithm.

Chapter 6

Applications

While we can demonstrate convergence of the smoothing algorithm to an integer solution, we have yet to show whether it is a good integer solution. To do this, we study the performance of the algorithm on problems whose optimal or best known solution is available. We also solve a number of real problems and compare the smoothing algorithm to the solution found by alternative methods. Many such problems are known in the literature to be $\mathcal{NP}$-hard or $\mathcal{NP}$-complete as discussed in Section 1.2.2.

One of the simpler classes of discrete optimization problems with nonlinear objective function is that of the unconstrained binary quadratic problem. Extensive numerical results and performance comparisons for various algorithms in this class are presented in Section 6.1. In fact, since this class of problems is sufficiently simple without complications arising from the objective function or the constraints, these numerical results could serve as a basis for deciding whether certain implementation procedures are favorable or not to the smoothing algorithm.

A nonlinear discrete problem that has linear constraints is the assignment of frequencies in a cellular network. In Section 6.2, we formulate the problem mathematically and give numerical results from applying the algorithm to this problem using real-world data. In Section 6.3, we look at another type of assignment problem that has been studied extensively, known as the quadratic assignment problem (see [PRW94, Cel98]). Nonlinear discrete optimization problems also arise in other applications such as medical decision analysis. In Section 6.4, we look into one such problem of finding the optimal number of biopsy needles that should be performed on a patient diagnosed with cancer in order to maximize the detection of cancer.

Table 6.1 shows the platform and software that were used to perform the numerical computations. Also, we have used MATLAB[1] to implement the smoothing algorithms for all the applications discussed in this chapter.

Table 6.1: Platform and software used for numerical comparison.

Platform	1	2	3
Processor	Pentium II (400 MHz)	Sun Ultra	SGI
Operating System	Windows 95	Solaris 8	IRIX 6.5
Memory Size	128 MB	256 MB	1.5 GB
Software Used	Matlab 5.3	Matlab 6.1	GAMS 20.5, Matlab 6.0
Precision	2.22×10^{-16}	2.22×10^{-16}	2.22×10^{-16}

6.1 Unconstrained Binary Quadratic Programming

The unconstrained binary quadratic problem (BQP) is to minimize a quadratic objective function subject to the variables being binary:

$$\begin{array}{rl} \text{Minimize} & x^{\mathrm{T}}Px + c^{\mathrm{T}}x \\ \text{subject to} & x \in \{0,1\}^n, \end{array} \tag{P6.1}$$

where $P \in \mathbb{R}^{n\times n}$ and $c \in \mathbb{R}^n$. By noting that $x^{\mathrm{T}}Mx = x^{\mathrm{T}}M^{\mathrm{T}}x$ and $c^{\mathrm{T}}x = x^{\mathrm{T}}Cx$ for any feasible x and any $M \in \mathbb{R}^{n\times n}$, where $C = \mathrm{Diag}(c)$, it suffices to consider the following "simpler" problem

$$\begin{array}{rl} \text{Minimize} & x^{\mathrm{T}}Qx \\ \text{subject to} & x \in \{0,1\}^n, \end{array} \tag{P6.2}$$

where Q is a symmetric matrix.

[1] www.mathworks.com

6.1.1 Examples

It is clear that any unconstrained quadratic programming problem with purely integer variables from a bounded set can be transformed into (P6.1) by the reformulation techniques discussed in Section 1.3.2. This would then include many classes of problems, including least-squares problems with bounded integer variables:

$$\underset{x \in D}{\text{Minimize}} \, \|s - Ax\|_2, \tag{P6.3}$$

where $s \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, and D is a bounded subset of $\mathbb{Z}^n$.

An example of BQP arising out of real-world applications is the multiuser detection problem in synchronous CDMA (Code Division Multi-Access) communication systems as described in [LPWH01]. In short, it was necessary to obtain an estimate of $x \in \{-1, 1\}^m$ in

$$y = RWx + r,$$

where x is a vector of bits transmitted by m active users, R is a symmetric normalized signature correlation matrix with unit diagonal elements, W is a diagonal matrix with diagonal elements being signal amplitudes of the corresponding users, and r is Gaussian noise with zero mean and known covariance matrix. The maximum likelihood estimate of x can then be obtained by solving

$$\begin{array}{ll} \text{Minimize} & x^{\mathrm{T}} W R^2 W x - 2y^{\mathrm{T}} R W x \\ \text{subject to} & x \in \{-1, 1\}^m, \end{array}$$

which can be transformed into (P6.2). Other examples include machine scheduling [AKA94] and molecular conformation [PR94], as well as graph problems such as those determining maximum cuts [BMZ00] and maximum cliques [PR92].

6.1.2 Numerical Results

One of the popular test sets for this class of problems is from the OR-library maintained by J. E. Beasley (http://mscmga.ms.ic.ac.uk/info.html). The results of some heuristic algorithms applied to his data sets are reported in [Bea98] and [GARK00].

The entries of matrix Q are integers uniformly drawn from $[-100, 100]$, with density 10%. As the test problems were formulated as maximization problems, for purposes of comparison of objective values in this section, we now maximize the objective.

The smoothing algorithm was run on all 60 problems from Beasley's test set ranging from 50 to 2500 binary variables on Platform 3 of Table 6.1. The parameters of the smoothing algorithm used are the initial barrier parameter, $\mu_0 = 100$, the initial penalty parameter, $\gamma_0 = 1$, the ratio of reduction of barrier parameters, $\theta_\mu = 0.5$, the ratio of increment of penalty parameters, $\theta_\gamma = 2$, the major iteration limit, $N = 50$, and all tolerance levels $\epsilon = 0.01$. The initial iterate used for all the test problems is the analytic center of $[0, 1]^n$, i.e., $\frac{1}{2}e$. The objective values obtained by the smoothing algorithm with these parameter settings are shown in Tables 6.2 to 6.7.

Table 6.2: Comparison of numerical output of algorithms applied to BQP test problems with 50 binary variables based on objective values (maximization).

Problem Number	BARON	CPLEX	DICOPT	SBB	Smoothing Algorithm
1	2160	2160	1646	1646	2160
2	3658	3658	3268*	3328	3658
3	4778	4778	4548	4548	4778
4	3472	3472	3280*	3330	3472
5	4152	4152	3886	3886	4152
6	3842	3842	3368*	3656	3842
7	4588	4588	4352	4352	4578 (0.22%)
8	4222	4222	3836*	3836	4222
9	3862	3862	3294*	3282	3862
10	3496	3496	3170*	3212	3496

In the tables (including those in subsequent sections), we used the symbol * to denote solver error or objective/solution vector cannot be obtained. The values obtained in [Bea98] or [GARK00] have not been included as we suspected some flaws in the data. Nevertheless, the accuracy of the data sets does not affect the testing of our algorithms as we were able to apply the nonlinear mixed-integer programming solvers of GAMS on the data sets as a basis of comparison. These solvers include the

Table 6.3: Comparison of numerical output of algorithms applied to BQP test problems with 100 binary variables based on objective values (maximization).

Problem Number	BARON	CPLEX	DICOPT	SBB	Smoothing Algorithm
1	7910	7910	7180	7180	7800 (1.39%)
2	11178	11178	10858*	11038	11112 (0.59%)
3	12956	12956	12790*	12898	12902 (0.42%)
4	10606	10606	10370	10370	10602 (0.04%)
5	8996	8996	8640*	8640	8992 (0.04%)
6	*	10486	9696*	9888	10364 (1.16%)
7	10030	10030	9650*	9672	9920 (1.10%)
8	11380	11380	10948	10948	11380
9	11340	11340	9776*	9776	11340
10	12438	12438	11550*	11924	12364 (0.59%)

Table 6.4: Comparison of numerical output of algorithms applied to BQP test problems with 250 binary variables based on objective values (maximization).

Problem Number	DICOPT	SBB	Smoothing Algorithm
1	45122*	45230	45744
2	43568	43568	44818
3	49052*	49222	49620
4	41110*	41150	41534
5	47670*	47670	48044
6	40364*	40364	41204
7	46210	46210	46644
8	33956*	34098	35612
9	48004*	48012	48442
10	39248*	39332	40546

Table 6.5: Comparison of numerical output of algorithms applied to BQP test problems with 500 binary variables based on objective values (maximization).

Problem Number	DICOPT	SBB	Smoothing Algorithm
1	109712*	112984	115602
2	125352*	125370	128378
3	127730	127730	131084
4	124410*	124976	129358
5	122560*	122708	124736
6	120690*	120566	121406
7	117820*	118662	122350
8	118606*	118606	123196
9	116614	116614	121218
10	124404*	127302	130684

Table 6.6: Comparison of numerical output of algorithms applied to BQP test problems with 1000 binary variables based on objective values (maximization).

Problem Number	DICOPT	SBB	Smoothing Algorithm
1	364866	364866	371132
2	345642*	348176	354332
3	364062*	364222	369562
4	368028*	368184	369880
5	343414	343414	352252
6	354238*	354654	360248
7	365634*	365634	369756
8	344482*	344562	350040
9	341366*	342470	348646
10	345684*	346158	350338

Table 6.7: Comparison of numerical output of algorithms applied to BQP test problems with 2500 binary variables based on objective values (maximization).

Problem Number	DICOPT	SBB	Smoothing Algorithm
1	1494902*	1499132	1513318
2	1450470*	1451398	1468492
3	1399406*	1401190	1408836
4	1493204*	1496206	1503950
5	1479610*	1479760	1491090
6	1447998	1447998	1464820
7	1466628*	1466674	1471098
8	1470002*	1470382	1482778
9	1450380*	1460478	1480266
10	1472306*	1472796	1475248

DICOPT solver, which is based on extensions of the outer-approximation algorithm for the equality relaxation strategy and solving a series of nonlinear programming and mixed-integer linear programming problems; and the SBB solver, which is based on a combination of the standard branch-and-bound method for the mixed-integer linear programming problems and standard nonlinear programming solvers. Also, as described in [Bea98], (P6.2) can be reformulated as a binary linear program (at the cost of squaring the number of variables) as follows:

$$\begin{array}{lll} \text{Maximize} & q^{\mathrm{T}}y & \\ \text{subject to} & y_{ij} \leq x_i, & i,j = 1,2,\ldots,n \\ & y_{ij} \leq x_j, & i,j = 1,2,\ldots,n \\ & y_{ij} \geq x_i + x_j - 1, & i,j = 1,2,\ldots,n \\ & x \in \{0,1\}^n & \\ & y \in \{0,1\}^{n^2}, & \end{array}$$

where $q = \mathrm{vec}(Q)$. This allows us to use the Mixed-Integer Solver in CPLEX as another source of comparison. For all the testing using GAMS as the interface (including the subsequent sections), we set the maximum iteration limit to be 100,000 and the maximum iteration time to be 100,000 seconds. In addition, we used BARON, the global optimization solver based on the branch-and-reduce method discussed in Section 1.3.3.2. Both CPLEX and BARON could be used only on problems with 100 or fewer variables.

CPLEX found the optimal solution for all 20 problems on which it could be run. The smoothing algorithm successfully found 11 of these optima, with another 6 results having less than 1% error in the optimal objective value. The error in remaining 3 results were also close, with the worst one having an error of 1.39% in the optimal objective value. All the error percentages for the results obtained by the smoothing algorithm are shown bracketed in Tables 6.2 and 6.3. On the other hand, the DICOPT solver obtained results with error percentages ranging from 4.81% to 23.8% for the 50-variable test problems and from 1.28% to 13.79% for the 100-variable test problems, while the SBB solver obtained results with error percentages ranging from 4.09% to 23.8% for the 50-variable test problems and from 0.44% to 13.79% for the 100-variable test problems. The DICOPT and SBB solvers obtained the same objective values for many of the test problems, especially those in which no solver error was encountered in using DICOPT. This is because both solvers use the same nonlinear programming solvers to solve the underlying relaxation problems.

After comparing Tables 6.2 to 6.7, we see that the smoothing algorithm produced better solutions than the DICOPT/SBB solvers even though we cannot verify whether these solutions are globally optimal or good quality local optimal solutions for Tables 6.4 to 6.7. However, it is also necessary to compare the performance in terms of the computational time involved in generating the solution vectors. First, we did a comparison of the smoothing algorithm with and without preconditioning to see how the efficiency and computational time are affected by preconditioning. In this case, the preconditioner introduced is the diagonal matrix arising from the Hessian of the logarithmic barrier term, i.e., $P = \mu X_H$, where X_H is defined in Chapter 3. Thus, the reduced Hessian system we are solving without preconditioning at iterate x is

$$(2Q - 2\Gamma + \mu X_H)u = -2Qx - \Gamma(e - 2x) + \mu X_g e,$$

while the preconditioned reduced Hessian system at iterate x is

$$(2P^{-\frac{1}{2}}QP^{-\frac{1}{2}} - 2P^{-1}\Gamma + I)v = P^{-\frac{1}{2}}(-2Qx - \Gamma\,(e - 2x) + \mu X_g e),$$

with $u = P^{-\frac{1}{2}}v$.

Table 6.8 shows the comparison of the computational effort required by the smoothing algorithm in terms of the number of CG iterations and the backtracking line-search iterations for using the reduced Hessian system and the preconditioned reduced Hessian system. It can be seen that preconditioning reduces the computational effort significantly on these test problems, and this is especially clear from the graph in Figure 6.1 that plots the number of CG iterations required versus the number of variables in the BQP test problems.

Table 6.8: Comparison of average number of iterations required to solve BQP test problems by smoothing algorithm with and without preconditioning.

Number of variables	Smoothing algorithm without preconditioning		Smoothing algorithm with preconditioning	
	Conj. Grad.	Backtrack	Conj. Grad.	Backtrack
50	659	128	192	121
100	997	152	257	143
250	1664	174	302	152
500	2474	190	355	166
1000	3893	213	394	171
2500	6927	293	524	200

Table 6.9 shows a comparison of the average computation time taken by DICOPT, SBB and the smoothing algorithm with and without preconditioning for the test problems with the same number of variables. Although the algorithms were run on the same platform, it should be noted that DICOPT and SBB are implemented in a commercial solver interface package while the smoothing algorithm is implemented in the higher level MATLAB language. The results in Table 6.9 illustrate that fewer iterations are required with preconditioning than without, and the cost of the preconditioning is minimal. Thus, preconditioning works well for this class of problems.

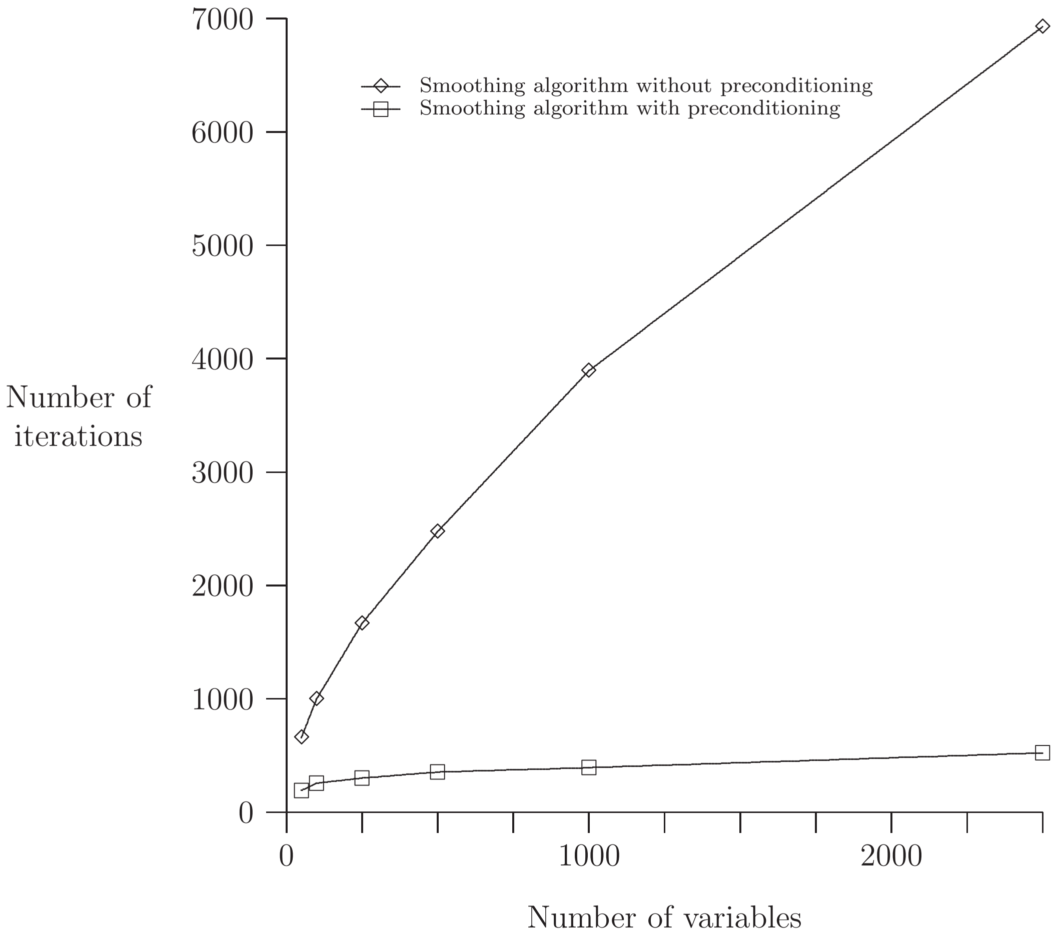

Figure 6.1: Graph of number of CG iterations required by the smoothing algorithm for solving BQP test problems.

In general, preconditioning may work well with the smoothing algorithm on other classes of nonlinear discrete optimization problem, but the impact would vary according to the nature of the problem, such as the objective function and the linear constraints involved in the problem. For the numerical results in the subsequent sections, we will only be considering the smoothing algorithm without preconditioning in order to assess the performance of the smoothing algorithm without any potential acceleration techniques.

Table 6.9: Comparison of average computation time (seconds) required to solve BQP test problems.

Number of variables	DICOPT	SBB	Smoothing algorithm without preconditioning	Smoothing algorithm with preconditioning
50	0.1	0.1	2.3	1.7
100	0.2	0.3	3.5	2.5
250	1.5	1.7	8.3	5.5
500	10.7	12.0	29.5	21.8
1000	85.2	89.5	217.4	108.8
2500	1494.6	1524.0	2850.9	997.2

To compare better the increase in the computational effort as the number of variables increase, we plotted in Figure 6.2 the average computation times taken by the three algorithms for solving the BQP test problems. It can be seen that when problem size increases, the increase in computation time for the smoothing algorithm with preconditioning is smaller than for the other two algorithms, at least for 2500 variables or less. This illustrates that compared to other algorithms, the smoothing algorithm may be increasingly efficient when the number of variables grows.

6.2 Frequency Assignment Problem

Consider the assignment of frequencies in a cellular network in which there are T transmitters and F frequencies. In practice, a transmitter uses several frequencies but a block of frequencies may be considered as one frequency.

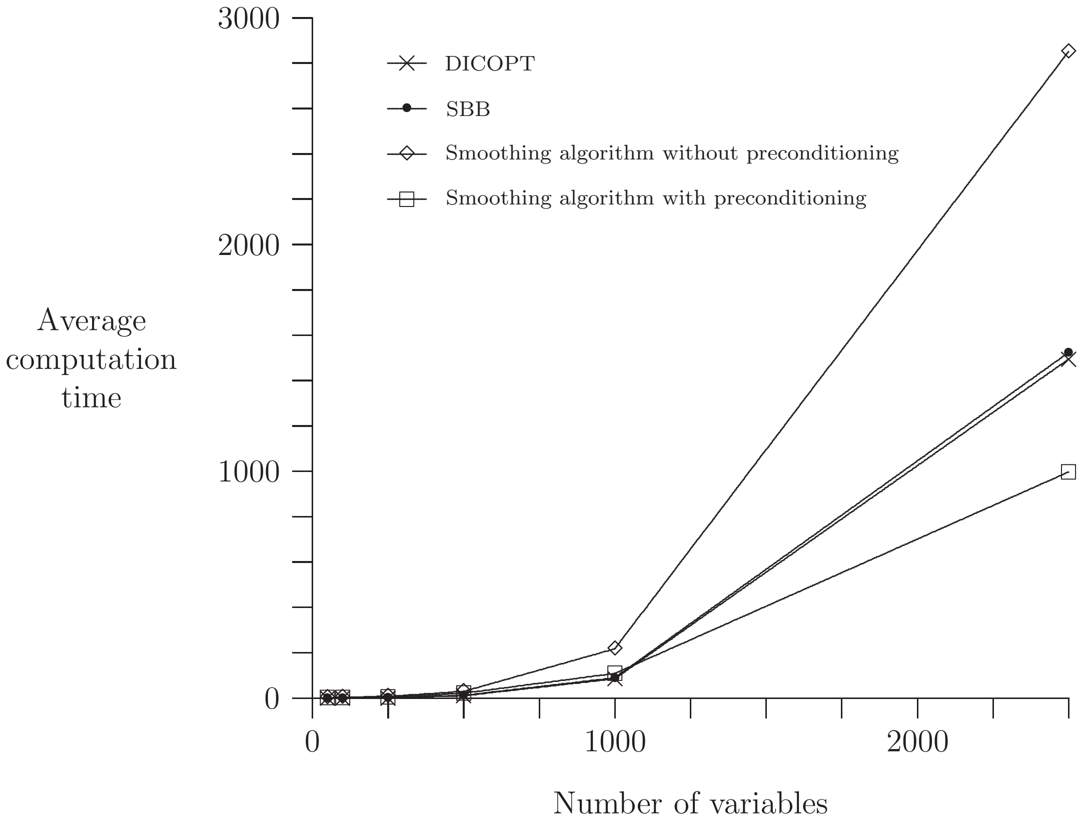

Figure 6.2: Graph of average computation time required by the algorithms for solving BQP test problems.

When $F < T$, it is necessary for two or more transmitters to use the same frequency and as a consequence, noise occurs at a receiver. Thus, there is a need to assign frequencies to these transmitters in such a way that the amount of noise is minimized. Typically, $F \ll T$, and so the number of possible assignments is astronomically large; an exhaustive search, even a highly selective one, is not possible. Moreover, for real problems, there is no assignment for which noise can be eliminated.

One approach to solving this problem is to take first an assignment, say T_1 uses F_3, T_2 uses F_7, etc. For each such assignment, the noise may be computed, and then another different assignment can be examined to see if the noise is reduced. The total number of choices using this approach is $(T-F)^F$ since there must be F transmitters assigned to F distinct frequencies to minimize the contribution of noise. This problem has been the focus of much research, and many algorithms have been proposed. For recent surveys of such algorithms, see [Dor01, DM02].

6.2.1 Formulation of the Problem in Binary Variables

Let $\mathcal{T}$ be the index set of transmitters and $\mathcal{F}$ be the index set of frequencies that can be assigned. Let S be the matrix (s_{ij}), where $i, j \in \mathcal{T}$ and s_{ij} is a measure of the amount of interference between transmitters i and j. The diagonal elements of S are zero.

Let X be the matrix (x_{ik}), where $i \in \mathcal{T}, k \in \mathcal{F}$ and x_{ik} is a number between 0 and 1 representing the proportion of frequency k used by transmitter i. In other words, we assume that all transmitters transmit on all frequencies, and our unknowns are the proportions used by a given transmitter. Since the unknowns are proportions, we have constraints

$$\sum_{k \in \mathcal{F}} x_{ik} = 1$$

for all $i \in \mathcal{T}$.

For a transmitter i, the amount of interference caused by assigning it frequency k is given by $\sum\limits_{j \in \mathcal{T}} s_{ij} x_{jk}$. Considering all the transmitters and all the frequencies, the total amount of interference is given by

$$\sum_{i\in\mathcal{T}}\sum_{k\in\mathcal{F}} x_{ik}\left(\sum_{j\in\mathcal{T}} s_{ij}x_{jk}\right) = \sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{T}}\sum_{k\in\mathcal{F}} s_{ij}x_{ik}x_{jk}.$$

The problem can then be written as

$$\begin{array}{ll} \text{Minimize} & \displaystyle\sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{T}}\sum_{k\in\mathcal{F}} s_{ij}x_{ik}x_{jk} \\ \text{subject to} & \displaystyle\sum_{k\in\mathcal{F}} x_{ik} = 1 \text{ for all } i \in \mathcal{T} \\ & x_{ik} = 0 \text{ or } 1 \text{ for all } i \in \mathcal{T},\ k \in \mathcal{F}. \end{array}$$

6.2.2 Transformation of the Problem

Letting $m = |\mathcal{T}|$ and $n = |\mathcal{F}|$, we transform the problem to

$$\begin{array}{ll} \text{Minimize} & \displaystyle\sum_{i=1}^{m}\sum_{k=1}^{n}\left[\sum_{j=1}^{m} s_{ij}x_{ik}x_{jk} - \mu\left[\ln x_{ik} + \ln(1-x_{ik})\right] + \gamma x_{ik}(1-x_{ik})\right] \\ \text{subject to} & \displaystyle\sum_{k=1}^{n} x_{ik} = 1 \text{ for all } i = 1,\ldots,m. \end{array}$$

The constraints $\sum_{k=1}^{n} x_{ik} = 1$ (for all i) can be written in the matrix form $Ax = e$, where

$$\begin{aligned} A &= [e_1,\ldots,e_1,e_2,\ldots,e_2,\ldots,e_m,\ldots,e_m] \text{ is an } (m\times mn) \text{ matrix,} \\ x &= \mathrm{vec}(X^{\mathrm{T}}),\ \text{and} \\ e_i &= i\text{th column of the } (m\times m) \text{ identity matrix } I_m. \end{aligned}$$

6.2.3 Reduced Hessian Matrix

The simple form of A enables us to define and obtain a sparse and structured form of the null-space matrix, namely the $(mn \times m(n-1))$ matrix

$$Z = \begin{bmatrix} -e^{\mathrm{T}} & 0 & \dots & 0 \\ I_{n-1} & 0_{n-1} & \dots & 0_{n-1} \\ 0 & -e^{\mathrm{T}} & \dots & 0 \\ 0_{n-1} & I_{n-1} & \dots & 0_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -e^{\mathrm{T}} \\ 0 & 0 & \dots & I_{n-1} \end{bmatrix}.$$

This allows us to perform matrix-vector products $Z^{\mathrm{T}}HZv$ efficiently when solving the reduced Hessian system (3.11) by CG methods.

6.2.4 Numerical Results

We are grateful to Magnus Almgren of the LM Ericsson Telephone Co. for providing data collected from the Dallas region of Texas. The problem they provided has 137 stations and the density of the interference matrix S is 53%. We considered the assignment of 5, 8, 10, 12, 15, 18 and 20 frequencies to these stations.

The parameters of the smoothing algorithm used are the initial barrier parameter, $\mu_0 = 100$, the initial penalty parameter, $\gamma_0 = 0$, the ratio of reduction of barrier parameters, $\theta_\mu = 0.95$, the major iteration limit, $N = 50$, and all tolerance levels $\epsilon = 0.001$. The initial iterate used for all the test problems is $\frac{1}{m}e$, where m is the number of frequencies available to be assigned. The objective values obtained by running the smoothing algorithm on Platform 3 of Table 6.1 with these parameter settings are shown in Table 6.10.

In all the tests, almost all the variables $x_{i,k}$ were sufficiently close to 0 or 1 at termination, even though the penalty term (3.3) was not introduced. The variables that were not 0 or 1 at termination indicate possible alternative solutions obtained by rounding up or down and maintaining feasibility. In fact, the objective value obtained by such rounding procedures is usually invariant.

Our results on the objective values obtained after rounding are compared to three other methods, namely the modified de Werra-Gay algorithm [DG94], the DICOPT and SBB solvers, and Dorstenstein's local recoloring algorithm [Dor01], running on Platforms 2, 1 and 2 of Table 6.1 respectively. Both the de Werra-Gay and Dorstenstein algorithms are specifically designed for the frequency assignment problem. In fact, Dorstenstein's results for the frequency assignment problems are far better than those obtained by other algorithms for this problem. In the comparison shown in Table 6.10, we find that the smoothing algorithm is getting solutions with objective values close to those of Dorstenstein's algorithm while outperforming the other two methods.

Table 6.10: Comparison of algorithms applied to the frequency assignment problem with Ericsson's data set based on objective values (minimization).

Number of frequencies n	Modified de Werra-Gay Algorithm	DICOPT	SBB	Dorstenstein's Local Recoloring Algorithm	Smoothing Algorithm
5	8708.4	24322.4*	6707.7	6279.0	6318.5
8	4006.6	2985.4	2985.4	2534.9	2715.5
10	2527.8	1970.7	1970.7	1581.4	1656.2
12	1775.1	1266.0	1266.0	1047.3	1083.6
15	991.0	745.9	745.9	588.5	683.8
18	630.2	499.4	499.4	346.6	383.1
20	437.5	344.9	344.9	238.4	329.1

We have also included in Table 6.11 a comparison of the computational time taken by the DICOPT and SBB solvers and the smoothing algorithm. The large computational time of the smoothing algorithm as compared to that of both the DICOPT and SBB solvers stems from a conservative setting of the parameters involved in the algorithm and from the MATLAB implementation. Still, the computational time taken by the algorithm is not unreasonable in view of the large number of variables involved.

Table 6.11: Comparison of algorithms applied to the frequency assignment problem with Ericsson's data set based on computation time (seconds).

Number of frequencies n	DICOPT	SBB	Smoothing Algorithm
5	182*	18	2460
8	22	23	5218
10	25	24	8218
12	32	32	13975
15	44	41	32203
18	61	53	30970
20	56	60	27887

6.3 Quadratic Assignment Problem

In some facility location problems, it is necessary to place exactly L facilities within L areas in such a way as to minimize the total cost of interaction between any two facilities. To describe this in greater detail, let A denote the interaction matrix among facilities, i.e., A_{ij} is the amount of interaction between facilities i and j, and let B denote the distance matrix, i.e., B_{kl} is the distance between zones k and l, where $i, j, k, l \in \{1, 2, \ldots, L\}$. Also, let C be the facility setup cost matrix, i.e., C_{ik} is the setup cost for facility i to be placed in zone k, where $i, k \in \{1, 2, \ldots, L\}$. Then if $\sigma \in S_L$ is a permutation such that $\sigma(i)$ denotes the zone for facility i to be placed in, where $i \in \{1, 2, \ldots, L\}$, then the quadratic assignment problem (QAP) involves finding an optimal permutation σ such that the following objective function measuring the total cost of interaction is minimized:

$$\sum_{i=1}^{L} \sum_{j=1}^{L} A_{ij} B_{\sigma(i)\sigma(j)} + \sum_{i=1}^{L} C_{i\sigma(i)}.$$

Here, the size of the QAP is defined to be L.

There are also applications other than facility location problem that can be formulated as a QAP, e.g., the backboard wiring problem in electronics [Ste61], as well as problems in human factors engineering such as the design of keyboards [LA93].

6.3.1 Equivalent Formulation

It is possible to find for the QAP an equivalent formulation that is in the form of (P3.1). Let

$$x_{ik} = \begin{cases} 1, & \text{if facility } i \text{ is placed in zone } k \\ 0, & \text{otherwise.} \end{cases}$$

Then an equivalent formulation would be

$$\begin{array}{ll} \text{Minimize} & \sum_{i=1}^{L}\sum_{j=1}^{L}\sum_{k=1}^{L}\sum_{l=1}^{L} A_{ij}B_{kl}x_{ik}x_{jl} + \sum_{i=1}^{L}\sum_{k=1}^{L} C_{ik}x_{ik} \\ \text{subject to} & \sum_{k=1}^{L} x_{ik} = 1, \quad \text{for all } i \\ & \sum_{i=1}^{L} x_{ik} = 1, \quad \text{for all } k \\ & x_{ik} = 0 \text{ or } 1, \quad \text{for all } i,\, k. \end{array} \tag{P6.4}$$

Letting $X = (x_{ij})$ and introducing the Kronecker product and vec notation, the objective function in (P6.4) can be simplified to

$$\text{vec}(X)^{\mathrm{T}}(A \otimes B)\text{vec}(X) + \text{vec}(C)^{\mathrm{T}}\text{vec}(X).$$

Thus, it will be easy to perform arithmetic operations involving the gradient and Hessian of the objective function using the above expression.

6.3.2 Numerical Results

The QAP test problems that were used for comparing the performance of the smoothing algorithm are the Nugent facility location problems [NVR68] and the Steinberg backboard wiring problem (using both Manhattan and squared Euclidean distances) mentioned earlier, and the numerical data was obtained from the QAPLIB (http://www.opt.math.tu-graz.ac.at/qaplib/). The former set of test problems is probably the most frequently used for comparison purposes. The difficulty of solving the QAP problems is illustrated by the fact that the Nugent problem with 30 variables was solved to optimality only in June 2000, while the Steinberg problem with 36 variables was solved to optimality in October 2001 [BA01].

The parameters of the smoothing algorithm used are the initial barrier parameter, $\mu_0 = 100$, the initial penalty parameter, $\gamma_0 = 0.01$, the ratio of reduction of barrier parameters, $\theta_\mu = 0.7$, the ratio of increment of penalty parameters, $\theta_\gamma = 1.43$, the major iteration limit, $N = 50$, and all tolerance levels $\epsilon = 0.01$. The initial iterate used for all the test problems is $\frac{1}{L}e$, where L is the size of the QAP. The objective values obtained by the smoothing algorithm running on Platform 3 of Table 6.1 with these parameter settings are shown in Table 6.12.

Since the test problems have proven optimal solutions, we are able to compare the quality of the solutions obtained by the smoothing algorithm. The nature of this combinatorial problem makes it possible to include simple local search algorithms to further improve the solutions obtained by the algorithm. For this purpose, we used the 2-OPT local search algorithm, which means that given a permutation $\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_L]$, we check if other permutations $\tau = [\tau_1, \tau_2, \ldots, \tau_L]$ would produce a solution with lower objective value, where

$$\begin{aligned}
\tau_i &= \sigma_i, \ i \neq p, q \\
\tau_p &= \sigma_q \\
\tau_q &= \sigma_p,
\end{aligned}$$

for some p, q. In addition, we include in Table 6.12 the objective values obtained from the DICOPT solver running on Platform 1 of Table 6.1 and the SBB solver running on Platform 3 of Table 6.1.

From Table 6.12, we see that the smoothing algorithm obtains solutions with objective values close to optimal, and that these solutions can sometimes be further improved by local search algorithms. The error percentage of the objective value obtained by the smoothing algorithm with 2-OPT are shown bracketed and ranges from 0.07% to 1.82%. Though the SBB solver obtains marginally better solutions for the smaller problems than the smoothing algorithm, it (as well as the DICOPT

Table 6.12: Comparison of algorithms applied to QAP test problems based on objective values (minimization).

Test Problem	Optimal Objective Value	DICOPT	SBB	Smoothing Algorithm	Smoothing Algorithm with 2-OPT
Nugent-12	578	660	578	590	586 (1.38%)
Nugent-15	1150	1198	1152	1160	1160 (0.87%)
Nugent-20	2570	2634	2588	2578	2574 (0.16%)
Nugent-30	6124	*	*	6128	6128 (0.07%)
Steinberg-36a	9526	10228*	10524	9680	9622 (1.01%)
Steinberg-36b	15852	*	18258	16492	16140 (1.82%)

Table 6.13: Comparison of algorithms applied to QAP test problems based on computation time (seconds).

Test Problem	DICOPT	SBB	Smoothing Algorithm
Nugent-12	3	20	55
Nugent-15	12	101	120
Nugent-20	54	556	624
Nugent-30	*	*	7813
Steinberg-36a	821*	4564	20230
Steinberg-36b	*	1112	29410

solver) is unable to handle the larger problems. Thus, the smoothing algorithm has the advantage of obtaining a solution very close to the best solution and yet not being seriously impacted by the size of the problem. We also show in Table 6.13 the computation time taken by the smoothing algorithm to obtain these solutions.

6.4 Medical Decision-Making

In the previous sections, we looked at classical problems in the literature that have been the focus of research efforts for an extended period of time. There are certainly many other recent applications in industry that can be formulated as nonlinear discrete optimization problems in which the smoothing algorithm can be applied to yield good solutions. We consider one such problem in medical decision-making.

Consider the problem of determining optimal positions for certain medical procedures to be administered in order to maximize the detection or treatment of cancer in a patient. An example of such a problem is described in [SZ01], where it is required to determine optimal locations for biopsy needles to be placed in the prostate of a patient, such that the probability of detecting cancer is maximized. In short, let

$$x_k = \begin{cases} 1, & \text{if a biopsy needle is administered in zone } k \\ 0, & \text{otherwise,} \end{cases}$$

where $k \in \{1, 2, \ldots, K\}$ and K is the total number of prostate zones to be considered. Also, suppose $Q = ee^{\mathrm{T}} - P$, where P is the probability matrix for cancer detection, i.e., P_{ik} is the estimated probability that a biopsy needle administered in zone k will detect cancer in patient i, where $k \in \{1, 2, \ldots, K\}$, $i \in \{1, 2, \ldots, T\}$, and T is the total number of patients. Then the problem can be formulated and simplified to

$$\begin{array}{ll} \text{Minimize} & \sum_{i=1}^{T} \prod_{k=1}^{K} Q_{ik}^{x_k} \\ \text{subject to} & \sum_{k=1}^{K} x_k = L \\ & Ax = b \\ & x_k = 0 \text{ or } 1, \quad \text{for all } k, \end{array} \tag{P6.5}$$

where L is the total number of biopsy needles to be administered to a patient. The constraint $Ax = b$ is used to take into account other constraints such as the left-right symmetry constraints that are not included in (P6.5).

In [SZ01], the nonconvex problem (P6.5) is transformed to a convex programming problem before applying Generalized Benders Decomposition (GBD) to solve it. For purposes of comparison with the performance of the smoothing algorithm applied to problems with the objective function not necessarily convex, we have applied the algorithm directly to (P6.5). The data was obtained from Professor A. Sofer and the problem has 48 binary variables with the Q matrix being 278 by 48.

The parameters of the smoothing algorithm used are the initial barrier parameter, $\mu_0 = 100$, the initial penalty parameter, $\gamma_0 = 0.1$, the ratio of reduction of barrier parameters, $\theta_\mu = 0.75$, the ratio of increment of penalty parameters, $\theta_\gamma = 1.33$, the major iteration limit, $N = 20$, and all tolerance levels $\epsilon = 0.1$. The initial iterate used for all the test problems is $\frac{L}{K}e$, where L and K are defined above. The objective values obtained by the smoothing algorithm running on Platform 1 of Table 6.1 with these parameter settings are shown in Table 6.14.

Table 6.14 also shows the objective values obtained by the GBD algorithm running on Platform 2 of Table 6.1, as well as those obtained from the DICOPT and SBB solvers running on Platform 1 of Table 6.1. The iteration limit and tolerance level for the GBD algorithm were 10^6 and 10^{-12} respectively. As the GBD and the smoothing algorithms are written in the same programming language, namely MATLAB, it is meaningful to compare their computation times as in Table 6.15. It can be seen from both tables that the smoothing algorithm is able to generate good quality solutions within reasonable time.

Table 6.14: Comparison of algorithms applied to prostate cancer detection problem based on objective values (minimization).

Number of Needles	GBD Technique	DICOPT	SBB	Smoothing Algorithm
6	52.3	67.8	52.3	52.3
8	40.8	45.5	40.8	40.8
10	34.1	39.0	34.1	34.2
12	28.6	31.8	28.6	29.7
14	24.9	28.3	24.9	25.2
16	21.8	24.3	22.2	22.4

Table 6.15: Comparison of algorithms applied to prostate cancer detection problem based on computation time (seconds).

Number of Needles	GBD Technique	DICOPT	SBB	Smoothing Algorithm
6	14	3	27	83
8	67	6	27	81
10	322	5	43	76
12	597	5	25	86
14	906	11	48	86
16	983	4	39	70

Chapter 7

Extension to More General Problems and Conclusions

Though this thesis is focused on smoothing algorithms for nonlinear optimization with linear equality constraints and pure bounded integer variables, it is possible to extend the algorithm to handle more general optimization problems. This is discussed in Section 7.1. Conclusions and further work to be done on these smoothing algorithms are given in Section 7.2.

7.1 More General Problems

There are many nonlinear discrete optimization problems that are not restricted to the format of problem (P3.1). For example, we may need to have both integer and continuous variables present in the formulation of a problem, i.e., a mixed-integer programming problem with nonlinear objective function. We illustrate next how the smoothing algorithms could be adapted to such problems, and some of the results in the earlier chapters could then be carried over for these problems. However, we do not concern ourselves with issues such as checking whether the feasible region is non-empty or not, and how to obtain an initial iterate for these general problems. Also, we assume that the integer variables are bounded, so that we essentially only need to consider problems whose discrete variables are binary variables.

7.1.1 Adding Continuous Variables

Consider the following nonlinear optimization problem in which there are some variables that are not necessarily discrete:

$$\begin{array}{ll} \text{Minimize} & f(x,y) \\ \text{subject to} & A_1 x + A_2 y = b \\ & x \in \{0,1\}^p,\ y \in \mathbb{R}^q. \end{array} \tag{P8.1}$$

We have partitioned the variables in such a way that x denotes the discrete variables, while y denotes the continuous variables. Since our approach involves converting discrete variables to continuous variables, adding continuous variables to the problem is relatively straightforward.

We can perform a continuous relaxation of problem (P8.1) to obtain

$$\begin{array}{ll} \text{Minimize} & f(x,y) \\ \text{subject to} & A_1 x + A_2 y = b \\ & 0 \le x \le e,\ y \in \mathbb{R}^q, \end{array} \tag{P8.2}$$

and the appropriate smoothing function to be added would again be $\Phi(x)$ of (3.2). The resulting problem after smoothing is then

$$\begin{array}{ll} \text{Minimize} & f(x,y) + \mu\Phi(x) \\ \text{subject to} & A_1 x + A_2 y = b, \end{array} \tag{P8.3}$$

where we have omitted the implicit constraints $(x,y) \in (0,1)^p \times \mathbb{R}^q$. We can also add an optional penalty term to force the binary variables to their bounds, so that the problem becomes

$$\begin{array}{ll} \text{Minimize} & f(x,y) + \mu\Phi(x) + \gamma \sum_{j\in J} x_j(1-x_j) \\ \text{subject to} & A_1 x + A_2 y = b. \end{array} \tag{P8.3}$$

Now we can apply the smoothing algorithms in Chapter 3 to problem (P8.3), with slight modification. For example, the first-order optimality conditions (3.14) are now given by

$$\begin{aligned} \nabla_x f - \mu X_g e + \Gamma\,(e - 2x) + A_1^{\mathrm{T}}\lambda &= 0 \\ \nabla_y f + A_2^{\mathrm{T}}\lambda &= 0 \\ A_1 x + A_2 y &= b, \end{aligned} \tag{7.1}$$

where $X_g = \mathrm{Diag}(x_g)$ with $(x_g)_i = \frac{1}{x_i} - \frac{1}{1-x_i}$, $i = 1, \ldots, p$, $\Gamma = \gamma \mathrm{Diag}(\chi_J)$ with χ_J being a column vector such that the ith component of χ_J is 1 if $i \in J$ and 0 otherwise, and λ is the Lagrange multiplier of the constraint $A_1 x + A_2 y = b$. The corresponding Newton system is

$$\begin{bmatrix} \nabla^2_{xx} f + \mu X_H - 2\,\Gamma & \nabla^2_{xy} f & A_1^{\mathrm{T}} \\ \nabla^2_{xy} f & \nabla^2_{yy} f & A_2^{\mathrm{T}} \\ A_1 & A_2 & 0 \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle y \\ \triangle \lambda \end{bmatrix} = \begin{bmatrix} -\nabla_x f + \mu X_g e - \Gamma\,(e - 2x) - A_1^{\mathrm{T}}\lambda \\ -\nabla_y f - A_2^{\mathrm{T}}\lambda \\ b - A_1 x - A_2 y \end{bmatrix}, \tag{7.2}$$

where X_H= $\mathrm{Diag}(x_H)$ and $(x_H)_i = \frac{1}{x_i^2} + \frac{1}{(1-x_i)^2}$, $i = 1, \ldots, p$. This system can be simplified to a reduced Hessian system similar to (3.17), to which the CG method can be applied.

7.1.2 Adding Linear Inequality Constraints

Consider a nonlinear optimization problem with binary variables and a mix of both linear equality and inequality constraints:

$$\begin{aligned} \text{Minimize} \quad & f(x) \\ \text{subject to} \quad & A_1 x = b_1 \\ & A_2 x \le b_2 \\ & x \in \{0,1\}^p. \end{aligned} \tag{P8.4}$$

We can include slack variables y for the inequality constraints, so that we only need to deal with linear equality constraints:

$$\begin{array}{rl} \text{Minimize} & f(x) \\ \text{subject to} & A_1 x \qquad = b_1 \\ & A_2 x + y = b_2 \\ & 0 \le x \le e, \;\; y \ge 0, \end{array} \tag{P8.5}$$

The continuous relaxation of problem (P8.5) would be

$$\begin{array}{rl} \text{Minimize} & f(x) \\ \text{subject to} & A_1 x \qquad = b_1 \\ & A_2 x + y = b_2 \\ & 0 \le x \le e, \;\; y \ge 0. \end{array} \tag{P8.6}$$

In this case, the smoothing function to be added would be a modification of that in (3.2) to take into account the bounds on y, i.e., the new smoothing function is

$$\Phi(x, y) = -\sum_{j=1}^{p} (\log x_j + \log(1 - x_j)) + \sum_{j=1}^{q} \log y_j.$$

The resulting "smoothed" problem is

$$\begin{array}{rl} \text{Minimize} & f(x) + \mu \Phi(x, y) \\ \text{subject to} & A_1 x \qquad = b_1 \\ & A_2 x + y = b_2, \end{array} \tag{P8.7}$$

where we have omitted the implicit constraints $(x, y) \in (0, 1)^p \times (0, \infty)^q$. Again, an optional penalty term can be added to force the binary variables to their bounds, so that the actual problem we are solving is

$$\begin{array}{rl} \text{Minimize} & \mathcal{F}(x, y) \equiv f(x) + \mu \Phi(x, y) + \gamma \sum_{j \in J} x_j (1 - x_j) \\ \text{subject to} & A_1 x \qquad = b_1 \\ & A_2 x + y = b_2. \end{array} \tag{P8.8}$$

The first-order optimality conditions of (P8.8) are given by

$$\begin{aligned} \nabla_x \mathcal{F}(x,y) + A_1^{\mathrm{T}}\lambda_1 + A_2^{\mathrm{T}}\lambda_2 &= 0 \\ -\mu Y_g e + \lambda_2 &= 0 \\ A_1 x &= b_1 \\ A_2 x + y &= b_2, \end{aligned} \tag{7.3}$$

where $X_g = \mathrm{Diag}(x_g)$ with $(x_g)_i = \frac{1}{x_i} - \frac{1}{1-x_i}$, $i = 1, \ldots, p$, $Y_g = \mathrm{Diag}(y_g)$ with $(y_g)_i = \frac{1}{y_i}$, $i = 1 \ldots, q$, $\Gamma = \gamma \mathrm{Diag}(\chi_J)$ with χ_J being a column vector such that the ith component of χ_J is 1 if $i \in J$ and 0 otherwise, and λ_1, λ_2 are the Lagrange multipliers of the constraints $A_1 x = b_1$, $A_2 x + y = b$ respectively. The corresponding Newton system is

$$\begin{bmatrix} \nabla_{xx}^2 f(x) + \mu X_H - 2\,\Gamma & 0 & A_1^{\mathrm{T}} & A_2^{\mathrm{T}} \\ 0 & \mu Y_H & 0 & I \\ A_1 & 0 & 0 & 0 \\ A_2 & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle y \\ \triangle \lambda_1 \\ \triangle \lambda_2 \end{bmatrix} = \begin{bmatrix} -\nabla_x \mathcal{F}(x,y) - A_1^{\mathrm{T}}\lambda_1 - A_2^{\mathrm{T}}\lambda_2 \\ \mu Y_g e - \lambda_2 \\ b_1 - A_1 x \\ b_2 - A_2 x - y \end{bmatrix}, \tag{7.4}$$

where X_H= $\mathrm{Diag}(x_H)$ with $(x_H)_i = \frac{1}{x_i^2} + \frac{1}{(1-x_i)^2}$, $i = 1, \ldots, p$, and Y_H= $\mathrm{Diag}(y_H)$ with $(y_H)_i = \frac{1}{y_i^2}$, $i = 1, \ldots, q$. This system can be simplified to a reduced Hessian system similar to (3.17) before the CG method is applied.

7.2 Conclusions

In summary, we have developed a continuation approach to nonlinear optimization problems with discrete variables by transforming such problems into one that allows global or nonlinear optimization techniques for continuous variables to be applied. As these optimization techniques improve over time, the ability of our algorithm to obtain good quality or even global solutions to the nonlinear discrete optimization problem will also improve. Such a continuation approach offers the advantage of reducing the number of local minima encountered in the solution process. Also, the size of the original nonlinear discrete problem does not appear to affect the computational burden of the algorithms exponentially, unlike existing methods such as branch-and-bound.

We have also tested the performance of our proposed algorithms on different classes of nonlinear discrete problems. The numerical results show that most of the binary variables in the test problems were very close to 0 or 1 at termination of the algorithm. Also, good-quality solutions were being obtained in a reasonable amount of computational time.

Though the continuation approach we proposed provides the framework of a general-purpose algorithm to deal with large classes of nonlinear discrete optimization problems, it is also possible to tailor the approach to specific types of problems by presetting the parameters of the smoothing algorithm, computing the null-space matrix in advance, and exploiting the structure of the problem to improve efficiency in dealing with the reformulated continuous problem. Moreover, the algorithm has the flexibility to be used in conjunction with other exact solution techniques to produce better solutions to the original problem.

Besides running more types of test problems with the smoothing algorithm, other future work will include methods to improve the efficiency of the algorithm, such as better linesearch and preconditioning procedures. Also, the algorithm could be adapted to consider the smaller problems arising out of fixing binary variables that have converged rapidly to their bounds of 0 and 1. It would also be helpful to look into adaptive updating techniques for the parameters involved in the smoothing algorithms, so that alternative solutions can be examined. We could also consider introducing extra linear constraints to prevent convergence to the undesired local minima. Lastly, there are other smoothing functions that could be considered in this continuation approach, such as other types of barrier functions used in interior-point methods.

In conclusion, this thesis illustrates what the continuation approach can achieve in the realm of nonlinear discrete optimization. It is hoped that the approach will open the door to new and efficient techniques to handle many difficult nonlinear discrete optimization problems in the real world.

Bibliography

[AG90] E.L. Allgower and K. Georg (1990), *Numerical Continuation Methods: An Introduction*, Springer-Verlag, New York.

[AKA94] B. Alidaee, G. Kochenberger and A. Ahmadian (1994), 0-1 quadratic programming approach for the optimal solution of two scheduling problems, *International Journal of Systems Science*, 25, 401–408.

[BA01] N.W. Brixius and K.M. Anstreicher (2001), The Steinberg wiring problem, Working Paper, University of Iowa.

[Bea98] J.E. Beasley (1998), Heuristic algorithms for the unconstrained binary quadratic programming problem, Working Paper, Imperial College, http://mscmga.ms.ic.ac.uk/jeb/bqp.pdf

[Ber95] D.P. Bertsekas (1995), *Nonlinear Programming*, Athena Scientific, Belmont, Massachusetts.

[BMZ00] S. Burer, R. Monteiro and Y. Zhang (2000), Rank-two relaxation heuristics for max-cut and other binary quadratic programs, Technical Report TR00-33, Department of Computational and Applied Mathematics, Rice University.

[Bom99] E.G. Boman (1999), *Infeasibility and Negative Curvature in Optimization*, Ph.D. Thesis, Scientific Computing and Computational Mathematics Program, Stanford University, Stanford.

[Bor88] M. Borchardt (1988), An exact penalty approach for solving a class of minimization problems with boolean variables, *Optimization*, 19(6), 829–838.

[BSS93] M.S. Bazaraa, H.D. Sherali and C.M. Shetty (1993), *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons, New York.

[BW01] G. Belvaux and L.A. Wolsey (2001), Modeling practical lot-sizing problems as mixed-integer programs, *Management Science*, 47(7), 993–1007.

[CMY78] S.N. Chow, J. Mallet-Paret and J.A. Yorke (1978), Finding zeros of maps: homotopy methods that are constructive with probability one, *Mathematics of Computation*, 32, 887–899.

[Cel98] E. Cela (1998), *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer Academic Publishers, Dordrecht.

[CNW02] M. Crary, L.K. Nozick and L.R. Whitaker (2002), Sizing the US destroyer fleet, *European Journal of Operational Research*, 136(3), 680–695.

[CSD01] J.-F. Cordeau, F. Soumis and J. Desrosiers (2001), Simultaneous assignment of locomotives and cars to passengers trains, *Operations Research*, 49(4), 531–548.

[Dav53] D.F. Davidenko (1953), On a new method of numerical solution of systems of nonlinear equations, *Dokl. Akad. Nauk SSSR*, 88, 601–602.

[Del00] A. Del Gatto (2000), *A Subspace Method Based on a Differential Equation Approach to Solve Unconstrained Optimization Problems*, Ph.D. Thesis, Management Science and Engineering Department, Stanford University, Stanford.

[DG86] M.A. Duran and I.E. Grossmann (1986), An outer approximation algorithm for a class of mixed-integer nonlinear programs, *Mathematical Programming*, 36, 307–339.

[DG94] D. de Werra and Y. Gay (1994), Chromatic scheduling and frequency assignment, *Discrete Applied Mathematics*, 49, 165–174.

[DM02] T. Dorstenstein and W. Murray (2002), Constructive and exchange algorithms for the frequency assignment problem, *Journal of Combinatorial Optimization*, to appear.

[Dor01] T. Dorstenstein (2001), *Constructive and Exchange Algorithms for the Frequency Assignment Problem*, Ph.D. Thesis, Management Science and Engineering Department, Stanford University, Stanford, http://www.geocities.com/dorstenstein/research.html

[DS83] R.S. Dembo and T. Steihaug (1983), Truncated-Newton algorithms for large-scale unconstrained optimization, *Mathematical Programming*, 26, 190–212.

[FM68] A.V. Fiacco and G.P. McCormick (1968), *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley & Sons, New York and Toronto.

[FM93] A. Forsgren and W. Murray (1993), Newton methods for large-scale linear equality-constrained minimization, *SIAM Journal on Matrix Analysis and Applications*, 14(2), 560–587.

[FGM95] A. Forsgren, P. E. Gill, and W. Murray (1995). Computing modified Newton directions using a partial Cholesky factorization, *SIAM Journal on Scientific Computing*, 16, 139–150.

[GARK00] F. Glover, B. Alidaee, C. Rego, and G. Kochenberger (2000). One-pass heuristics for large scale unconstrained binary quadratic problems, Research Report HCES-09-00, Hearin Center for Enterprise Science, http://hces.bus.olemiss.edu/reports/hces0900.pdf

[Geo72] A.M. Geoffrion (1972), Generalized Benders decomposition, *Journal of Optimization Theory and Applications*, 10:4, 237–260.

[GH89] R. Ge and C. Huang (1989), A continuous approach to nonlinear integer programming, *Applied Mathematics and Computation*, 34, 39–60.

[GM74] P.E. Gill and W. Murray (1974), Newton-type methods for unconstrained and linearly constrained optimization, *Mathematical Programming*, 7, 311–350.

[GMSW79] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright (1979), Two steplength algorithms for numerical optimization, Report SOL 79-25, Management Science and Engineering Department, Stanford University, Stanford.

[GMW81] P.E. Gill, W. Murray, and M. Wright (1981), *Practical Optimization*, Academic Press, London.

[Gom58] R.E. Gomory (1958), Outline of an algorithm for integer solutions to linear programs, *Bulletin of the American Mathematical Society*, 64, 275–278.

[GR85] O.K. Gupta and A. Ravindran (1985), Branch and bound experiments in convex nonlinear integer programming, *Management Science*, 31, 1533–1546.

[GTP98] M.D. González-Lima, R.A. Tapia, and F.A. Potra (1998). On effectively computing the analytic center of the solution set by primal-dual interior-point methods, *SIAM Journal on Optimization*, 8, 1–25.

[GV96] G.H. Golub and C.F. Van Loan (1996), *Matrix Computation*, The John Hopkins University Press, Baltimore and London.

[HRS00] R.G. Haight, C.S. Revelle, and S.A. Snyder (2000), An integer optimization approach to a probabilistic reserve site selection problem, *Operations Research*, 48(5), 697–708.

[Kan97] C. Kanzow (1997), A new approach to continuation methods for complementarity problems with uniform P-functions, *Operations Research Letters*, 20, 85–92.

[KMM90] M. Kojima, S. Mizuno, and T. Noma (1990), Limiting behavior of trajectories generated by a continuation method for monotone complementarity problems, *Mathematics of Operations Research*, 15, 662–675.

[KMM93] M. Kojima, N. Megiddo, and S. Mizuno (1993), A general framework of continuation methods for complementarity problems, *Mathematics of Operations Research*, 18, 945–963.

[KMN91] M. Kojima, N. Megiddo, and T. Noma (1991), Homotopy continuation methods for nonlinear complementarity problems, *Mathematics of Operations Research*, 16, 754–774.

[KRR91] N. Karmarkar, M.G.C. Resende, and K.G. Ramakrishnan (1991), An interior point algorithm to solve computationally difficult set covering problems, *Mathematical Programming*, 52, 597–618.

[LA93] L.W. Light and P.G. Anderson (1993), Designing better keyboards via simulated annealing, *AI Expert*, 8(9), 20–27.

[LB66] E.L. Lawler and M.D. Bell (1966), A method for solving discrete optimization problems, *Operations Research*, 14, 1098–1112.

[LPWH01] J. Luo, K. Pattipati, P. Willett, and F. Hasegawa (2001), Near-optimal multiuser detection in synchronous CDMA using probabilistic data association, *IEEE Communications Letters*, 5(9), 361–363.

[LW66] E.L. Lawler and D.E. Wood (1966), Branch-and-bound methods: A survey, *Operations Research*, 14, 699–719.

[Man69] O.L. Mangasarian (1969), *Nonlinear Programming*, McGraw-Hill, New York.

[Man76] O.L. Mangasarian (1976), Equivalence of the complementarity problem to a system of nonlinear equations, *SIAM Journal on Applied Mathematics*, 31, 89–92.

[Mit70] L.G. Mitten (1970), Branch-and-bound methods: General formulation and properties, *Operations Research*, 18, 24–34.

[Mit99] J.E. Mitchell (1999), Branch-and-cut algorithms for combinatorial optimization problems, *Handbook of Applied Optimization*, Oxford University Press, 2001 (to appear).

[MS79] J.J. Moré and D.C. Sorensen (1979), On the use of directions of negative curvature in a modified Newton method, *Mathematical Programming*, 16, 1–20.

[MW94] W. Murray and M.H. Wright (1994). Line search procedures for the logarithmic barrier function, *SIAM Journal on Optimization*, 4, 229–246.

[MW97] J.J. Moré and Z. Wu (1997). Global continuation for distance geometry problems, *SIAM Journal on Optimization*, 7, 814–836.

[NS95] S.G. Nash and A. Sofer (1995), *Linear and Nonlinear Programming*, McGraw-Hill, New York.

[NS96] S.G. Nash and A. Sofer (1996), Preconditioning reduced matrices, *SIAM Journal on Matrix Analysis and Applications*, 17, 47–68.

[NVR68] C.E. Nugent, T.E. Vollman, and J. Ruml (1968), An experimental comparison of techniques for the assignment of facilities to locations, *Operations Research*, 16, 150–173.

[NW99a] G.L. Nemhauser and L.A. Wolsey (1999), *Integer and Combinatorial Optimization*, John Wiley & Sons, New York.

[NW99b] J. Nocedal and S.J. Wright (1999), *Numerical Optimization*, Springer-Verlag, New York.

[PR88] R.G. Parker and R.L. Rardin (1988), *Discrete Optimization*, Academic Press, New York.

[PR92] P.M. Pardalos and G.P. Rodgers (1992), A branch and bound algorithm for maximum clique problem, *Computers and Operations Research*, 19, 363–375.

[PRW94] P.M. Pardalos, F. Rendl, and H. Wolkowicz (1994), The quadratic assignment problem: a survey of recent developments, in *Quadratic Assignment and Related Problems*, 16, 1–42, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Providence, RI.

[PR94] A.T. Phillips and J.B. Rosen (1994), A quadratic assignment formulation of the molecular conformation problem, *Journal of Global Optimization*, 4, 229–241.

[PS82] C.H. Papadimitriou and K. Steiglitz (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

[RS01] S. Rajagopalan and J.M. Swaminathan (2001), A coordinated production planning model with capacity expansion and inventory management, *Management Science*, 47(11), 1562–1580.

[Sah00] N.V. Sahinidis (2000), BARON Global Optimization Software User Manual, *http://archimedes.scs.uiuc.edu/baron/manuse.pdf*

[Sch98] A. Schrijver (1998), *Theory of Linear and Integer Programming*, John Wiley & Sons, New York.

[Sch01] M.A. Schulze (2001), *Active contours (snakes): A demonstration using Java*, http://www.markschulze.net/snakes/

[Sha89] L. Sha (1989), *A Macrocell Placement Algorithm Using Mathematical Programming Techniques*, Ph.D. Thesis, Electrical Engineering Department, Stanford University, Stanford.

[Shi00] T. Shiina (2000), Integer programming model and exact solution for concentrator location problem, *Journal of the Operations Research Society of Japan*, 43(2), 291–305.

[Ste61] L. Steinberg (1961), The backboard wiring problem: A placement algorithm, *SIAM Review*, 3, 37–50.

[SS92] A.C. Sun and W.D. Seider (1992), Homotopy-continuation algorithm for global optimization, in *Recent Advances in Global Optimization (editors:C.A. Floudas, P.M. Pardalos)*, Princeton University Press, Princeton, New Jersey.

[SZ01] A. Sofer and J. Zeng (2001), Optimized needle biopsy strategies for prostate cancer detection, submitted for publication.

[TM99] A. Tajima and S. Misono (1999), Using a set packing formulation to solve airline seat allocation/reallocation problems, *Journal of the Operations Research Society of Japan*, 42(1), 32–44.

[TS99] M. Tawarmalani and N.V. Sahinidis (1999), Global optimization of mixed-integer nonlinear programs: A theoretical and computational study, *Mathematical Programming*, submitted.

[Van01] F. Vanderbeck (2001), A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem, *Management Science*, 47(6), 864–879.

[VD68] A.F. Veinott, Jr. and G.B. Dantzig (1968), Integral extreme points, *SIAM Review*, 10(3), 371–372.

[VD02] V. Verter and A. Dasci (2002), The plant location and flexible technology acquisition problem, *European Journal of Operational Research*, 136(2), 366–382.

[Was73] E. Wasserstrom (1973), Numerical solutions by the continuation method, *SIAM Review*, 15(1), 89–119.

[Wat00] L.T. Watson (2000), Theory of globally convergent probability-one homotopies for nonlinear programming, *SIAM Journal on Optimization*, 11(3), 761–780.

[Wri97] S.J. Wright (1997), *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia.

[YC02] S. Yan, J.-C. Chang (2002), Airline cockpit crew scheduling, *European Journal of Operational Research*, 136(3), 501–511.

[Ye97] Y. Ye (1997), *Interior Point Algorithms: Theory and Analysis*, John Wiley & Sons, New York.

[ZGZ99] L.-S. Zhang, F. Gao, and W.-X. Zhu (1999), Nonlinear integer programming and global optimization, *Journal of Computational Mathematics*, 17:2, 179–190.

[ZL01] Y.-B. Zhao and D. Li (2001), On a new homotopy continuation trajectory for nonlinear complementarity problems, *Mathematics of Operations Research*, 26(1), 119–146.