PATH-FOLLOWING METHODS FOR LINEAR PROGRAMMING

A thesis submitted to the post-graduate School, Ahmadu Bello University, Zaria, Nigeria, in partial fulfilment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY (MATHEMATICS)

BY

MUHAMMAD MASUDUR RAHMAN QUABILI

Department of Mathematics

Faculty of Science

Ahmadu Bello University, Zaria, Nigeria

APRIL, 1999

D E D I C A T I O N

Dedicated to:

My respected father: Late Professor Muhammad Abdul Ghafur.

and my beloved mother: Late Syeda Hamida Akhtar Khatun.

DECLARATION

I, Muhammad Masudur Rahman Quabili, declare that this thesis is entirely my own work and not a published or an unpublished work of another author. No part of this dissertation has been submitted for a degree in any other University. Information derived from published or unpublished work of others has been duly acknowledged in the text.

Signature Masudur R. Quabili

(MUHAMMAD MASUDUR RAHMAN QUABILI)

B.A. (Rajshahi), M.A. (Dhaka)

April 6, 1999.

CERTIFICATION

This thesis entitled "PATH-FOLLOWING METHODS FOR LINEAR PROGRAMMING" by Muhammad Masudur Rahman Quabili meets the regulations governing the award of the degree of Doctor of Philosophy of Ahmadu Bello University, Zaria, and is approved for its contribution to knowledge and literary presentation.

.....
Dr. J.N. Singh
Chairman
Supervisory Committee

Date: 4/5/1999

.....
Dr. U.M. Makarfi
Member,
Supervisory Committee

Date: 23/7/99

DEPARTMENT OF MATHEMATICS
28 JUL 1999
A.B.U. ZARIA

.....
Dr. U.M. Makarfi
Head,
Department of Mathematics

Date: 23/7/99

.....
Professor S.B. Ojo
Dean,
Post graduate school

Date: 06/08/99

ACKNOWLEDGEMENT

Thanks to Almighty Allah (the most merciful, the beneficent), who created me as a human being, and gave me capability to carry out my research work successfully.

I do express my profound gratitude to my supervisor, Dr. J.N. Singh for his constant guidance, continuous help in analysing papers, suggesting topics, and also valuable advice in different dimensions to extend this research work.

I would also like to extend my gratitude to the Head of the Department of Mathematics, Dr U.M. Makarfi who permitted me to do my Ph.D. programme, for the cooperation and encouragement extended to me in the course of this study.

I am grateful to Professor M.A. Rashid for his scholarly and valuable advice regarding this research work and other academic matters.

I tender my warm thanks to all academic and non academic colleagues in the Department of Mathematics for their moral, academic support and encouragement given to me.

I thank Mr. Isah Abdulkarim who has joyfully typed this dissertation in his best possible way.

Finally, I want to express my immense and sincere gratitude to my wife, Aziza Khatun Quabili and my children Shabbir, Zakir and Shakura for all their love, encouragement, patience, understanding, moral support and cooperation.

M.R. Quabili

ABSTRACT

In this thesis a unified treatment of various Path - following methods for linear programming is presented.

A modification based on "Specifiic Most Violated Constraints" in Pivot and Probe Algorithm (PAPA) is introduced which ensures early termination of the algorithm.

While comparing the Ellipsoid method with Karmarkar's algorirthm it has been shown that the hearts of these two algorithms depend on the solution of weighted least squares subproblems and these subproblems are closely related. Some results related to the volume reduction inequality for the ellipsoid method and convergence inequality for Karmarkar's algorithm have also been obtained.

Various types of parameterization of central trajectory and proximity measures have been critically examined. As a result some inequalities which ensures faster convergence in Primal - Dual interior point path - following methods have been obtained.

TABLE OF CONTENTS

Tittle Page.....i

Dedication.....ii

Declaration.....iii

Certification.....iv

Acknowledgement.....v

Abstract.....vi

Table of content.....vii

CHAPTER 0. INTRODUCTION.....1

CHAPTER 1. LINEAR PROGRAMMING AND SIMPLEX METHOD

1.1 LINEAR PROGRAMMING.....6

1.1.1 Optimization problem.....6

1.1.2 Programming problems.....6

1.1.3 Linear programming.....7

1.1.4 Mathematical formulation of linear programming problem.....8

1.2 SOME DEFINITIONS.....9

1.3 THE SIMPLEX METHOD.....12

1.3.1 Introduction.....12

1.3.2 Some Notations.....12

1.3.3 Some important Theorems.....13

1.3.4 Revised simplex method.....15

1.4 DUALITY IN LINEAR PROGRAMMING.....15

1.4.1 Introduction.....15

KASU

1.4.2 Dual Simplex Algorithm 22

1.4.3 Primal-Dual Algorithm 24

1.5 PIVOT AND PROBE ALGORITHM (PAPA).....25

1.5.1 Mathematical information of

the optimal table for relaxed LP.....25

1.5.2 A proposed new Rule for selecting

the specific most violated constraint.....28

1.5.3 Implementation of the Rule to

solve a Numerical problem 34

1.6 COMPUTATIONAL COMPLEXITY AND

THE SIMPLEX ALGORITHM 37

CHAPTER 2. ELLIPSOID METHOD

2.1 INTRODUCTION 41

2.2 SOME DEFINITIONS 42

2.3 THE ELLIPSOID ALGORITHM 44

2.4 MODIFICATIONS OF THE ELLIPSOID

ALGORITHM 47

2.4.1 Deep cuts 47

2.4.2 Surrogate cuts 48

2.5 POLYNOMIAL SOLVABILITY 49

2.6 APPLICATION OF ELLIPSOID METHOD TO

LINEAR PROGRAMMING 50

2.6.1 Simultaneous solution of

Primal and Dual in R^{m+n} 50

2.6.2 Bisection method 52

2.6.3 Sliding objective function method 53

2.7 RELATIONS TO THE SIMPLEX METHOD 54

2.8 CONCLUDING REMARKS 55

CHAPTER 3. KARMARKAR'S ALGORITHM

3.1 INTRODUCTION 57

3.2 GENERAL PROJECTIVE TRANSFORMATION 57

3.2.1 Projective Transformation T (a,a_o).....58

3.3 CONVERSION OF A LINEAR PROGRAMMING PROBLEM TO CANONICAL FORM 59

3.3.1 Optimization over a polytope 63

3.3.2 Optimization over sphere 64

3.3.3 Bounds on the objective function 64

3.3.4 Invariant potential functions 65

3.4 THE ALGORITHM 65

3.5 THEOREMS 67

3.6 EFFICIENCY OF THE ALGORITHM 68

3.6.1 Convergence in Karmarkar's Algorithm.....68

3.6.2 Rate of convergence of Kamarkar's Algorithm.....68

3.7 COMPLEXITY OF THE ALGORITHM 69

3.8 TERMINATION OF ITERATIONS 71

3.9 COMPARISON OF THE ELLIPSOID METHOD AND KAMARKAR'S ALGORITHM 73

3.10 CONCLUDING REMARKS 83

CHAPTER 4. PATH FOLLOWING METHODS

4.1 INTRODUCTION 85

4.2 GEOMETRICAL ASPECTS OF THE CENTRAL PATH 86

4.3 TOOLS FOR GENERAL INTERIOR POINT METHODS 90

4.3.1 The linear programming problem 90

4.3.2 The scaling-steepest Descent algorithm.....94

4.3.3 Barrier method 101

4.3.4 Analytic center 104

4.3.5 Auxiliary functions 105

4.3.6 Guessing a Dual slack and a lower bound.....109

4.4 THE CENTRAL PATH 110

4.4.1 Primal-dual properties 111

4.4.2 Parametrerization of the central path.....112

4.4.3 Complexity of the conceptual path-following Algorithm 120

4.5 NEARLY CENTRAL POINTS 125

4.5.1 Proximity 126

4.5.2 Treatment of proximity 129
4.6 CENTRALIZATION.....134
4.6.1 Finding the Analytic center.....135
4.6.2 Centralization in primal-dual Parameterization 136
4.6.3 Centralization in primal Parameterization 137
4.6.4 Centralization in Dual Parameterization 138
4.7 IMPLEMENTABLE PENALTY FUNCTION AND CENTER METHODS.....138
4.7.1 Penalty function methods.....139
4.7.2 Methods of centes 143
4.8 PRIMAL-DUAL METHODS 144
4.8.1 Scaling.....145
4.8.2 Primal-dual central points 146
4.8.3 Nearly central points.....147
4.8.4 Centering 148
4.8.5 Some new convergence results for primal-dual interior point path-following algorithms.....151
4.8.6 Newton-Raphson Direction.....158
4.8.7 Proximity to the central path.....161
4.9 POTENTIAL REDUCTION METHODS 162
4.9.1 Primal only methods 164
4.9.2 Ye's Approach 166
4.9.3 Primal-Dual Algorithms 170

4.10 INITIALISATION AND TERMINATION.....172

4.11 CONCLUDING REMARKS

AND OPEN QUESTIONS 174

REFERENCES.....177

APPENDIX.... A computer program to implement modified pivot and probe algorithm.....187

CHAPTER 0

INTRODUCTION

We consider the standard linear programming problem

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } Ax=b \qquad (P) \\ &x \geq 0 \end{aligned}$$

Where $c \epsilon \mathbb{R}^n$, $b \epsilon \mathbb{R}^m$, A is a full-rank m x n matrix, n>m.

We assume that the feasible region.

$$S = \{x \in \mathbb{R}^n | Ax=b, x \geq 0\}$$

is bounded and has a nonempty relative interior given by

$$S^o = \{x \in \mathbb{R}^n | Ax=b, x > 0\}$$

The linear programming problem was first solved by Dantzig [17] in 1951. The Simplex method developed by him is still the most widely used algorithm, and it will possibly remain so in future. Although the Simplex Method is efficient and elegant, it does not posses a property that became more and more charming in the last three decades: polynomial complexity. A problem devised by Klee and Minty [51] forced the Simplex Method to execute a number of arithmetical operations that grew exponentially with the number of variables of the problem, attaching to the method an exponential worst-case complexity.

The question on whether a polynomial algorithm for the linear programming problem exists was answered in 1978 by Khachiyan [59,60]. He applied the Ellipsoidal method of Shor [103] and Yudin and

Nemirovskii [55] to the Linear Programming Problem and proved a polynomial bound on the number of arithmetical operations needed to find an optimal solution. The bound, $0(n^4L)$, depends on a number L, the length of the input, the total number of bits used in the description of the problem data, which is somewhat frustrating. The existence of a method with a complexity bound based only on the number of variables and constraints, is still a difficult open problem. The method raised on enormous enthusiasm, and had a great impact on the theory of complexity, but unfortunately the practical implementation have been irremediably inefficient.

Comprehensive studies of these two methods are discussed in Dantzig [18],Schrijver [99], Goldfarb and Todd [38]. Complexity issues are discussed in Shamir [102], Megiddo [77,78], Bland, Goldfarb and Todd [13], Borgwardt [14] and Tardos [116].

In 1984, Karmarkar [58] published his algorithm, which not only had a polynomial complexity bound of $0(n^{3.5}L)$ operations, lower than Khachiyan's [59], but was announced as more efficient than the Simplex Method. There was initially much discussion about this claim, but now it is clear that well-coded versions of the new methodology are very efficient, specially when the problem size increases above some thousands of variables. Karmarkar's Algorithm is essentially different from the Simplex Method in that it evolves through the relative interior of the feasible set, instead of following a sequence of vertices as does the Simplex Method. Karmarkar's Algorithm has a flavour of nonlinear

programming, in contrast with the combinatorial gait of the Simplex Method.

Karmarkar's Algorithm in its original form needed a special formulation of the linear programming problem, and relied on the knowledge of the value of an optimal solution, or a process for generating efficient lower bounds for it. Soon standard-form variants were developed by Anstericher [4], Gay [33], Gonzaga [41], Steger [115], and Ye and Kojima [130], and an efficient method for generating lower bounds for the optimal cost was devised by Todd and Burrell [119]. Another approach for finding lower bounds was developed by Anstreicher [3].

A thorough simplification of Karmarkar's algorithm reproduces the algorithm due to Dikin [20,21], which now received the name of "Affine-Scaling". Karmarkar's algorithm, its variants and implementations, are discussed in Godfarb and Todd [38].

Our concern starts from the fact that Karmarkar's algorithm performs well by avoiding the boundary of the feasible set. And it does this with the help of a classical resource, first used in optimization by Frisch [27] in 1955: the Logarithmic Barrier Function:

$$x \in \mathbb{R}^n,\ x > 0 \ \dashrightarrow\ p(x) = -\sum_{i=1}^{n} \log x_i$$

This function grows indefinitely near the boundary of the feasible set S, and can be used as a penalty attached to those points. Combining p(.) With the objective makes points near the boundary expensive, and forces

any minimization algorithm to avoid them.

A question is then naturally raised: how far from the boundary should one stay? A successful answer was given though the definition of the analytic center of a polytope by Sonnevend [114], the unique point that minimizes the barrier function. A well-behaved curve is formed by the analytic centers of all the constants - cost slices of the feasible set in (P): the central path. This is the subject of this work. This path is a region with some very attractive primal-dual properties, and provides an answer to our question: try to stay near the central path. Renegar did so in 1986 [95], and obtained the First PATH-FOLLOWING ALGORITHM, with a complexity lower than that of Karmarkar's Method in terms of number of iterations ($0(\sqrt{n}L)$ against $0(nL)$). Renegar's approach was based on Huard's method of centers [52].

Soon afterwards Vaidya [122], refining Renegar's results, and Gonzaga [42], following a penalty function aproach, described algorithms with an overall complexity of $0(n^3L)$ arithmetical operations, limit that is still standing. Simultaneously, Kojima, Mizuno, and Yoshise [65] developed a Primal-Dual Path-Following Method, which was soon to be reduced to that low complexity by the same authors [64] and by Monteiro and Adler [85]. A fourth approach based on Karmarkar's potential function appeared later, first in Ye [128] and then in Freund [25], and in Gonzaga [47].

Only proven complexity results were cited in the brief historical account above. An amazing fact has been found out by Anstreicher [7]:

the classical barrier function method (SUMT) developed for nonlinear programming by Fiacco and McCormick [23], exactly as implemented in 1968, solves linear and quadratic programs in $0(\sqrt{n}L \log L)$ iterations.

The field of Interior Point Methods has been extremely active in the last few years. More than two thousand papers were written, developing the four approaches for linear programming, extending them to convex quadratic programming, to linear complementarity problems and to convex nonlinear programming. Path-following Methods, which started as short-steps algorithms with nice theoretical properties, evolved into practical large-steps methods. The purpose of this thesis is to describe a unified treatment of central path algorithms, and to show how one arrives naturally at the four approaches, Methods of Centers, Penalty Function Methods, Potential Reduction Methods, and Primal-Dual Methods. We shall see that the good properties of points near the central path are intimately associated to nice primal-dual properties at these point and this will provide the unifying concepts for the whole theory. And at the end we shall be able to abandon the central path and work directly with primal-dual properties while keeping all the nice properties of path-following methods.

Proofs will be given only for some results and we shall restrain ourselves to linear programming.

The contents are arranged in accordance with the sequence of the discussion.

CHAPTER 1

LINEAR PROGRAMMING AND THE SIMPLEX METHOD

1.1 Linear Programming

1.1.1 Optimization Problem

The fundamental purpose of the study of optimization problem is to arrive at the best possible decision in the existing set of circumstances. Optimization problems arise in Operations Research, Economics, Aerospace, Physics, Control Theory, Chemical Engineering etc. The ultimate goal of such decisions is to either minimize the effort required to do a job or maximize the desired benefit. As effort required or benefit desired in any practical situation can be expressed as a function of certain decision variables, the problems ultimately reduce to seeking maximization or minimization of such functions subject to certain constraints. They are classified as optimization problems.

1.1.2 Programming Problems

A subclass of optimization problems may be referred as to what is known as programming problems. Broadly speaking, programming problems deal with determining optional allocations of limited resources to meet given objectives. More specifically they deal with situations where a number of resources such as men, materials, machines and land are available and they are combined to yield some targeted products. There are, however, certain restrictions which need to be put on all or some of these broad categories; for example on the amount of resources

available, the quality and quantity of the products etc. Even with these restrictions there will exist many feasible allocations. Out of permissible allocations of resources, it is desired to specify one or more which would maximize or minimize some numerically identifiable quantity such as profit, loss etc.

1.1.3 Linear Programming

Linear programming problems constitute a subclass of such optimization problems in which the function to be optimized and the constraints are all linear. These days, linear programming happens to be one of the most frequently used decision making tools in industry, administration and various other spheres of life. An extensive bibliography of linear programming applications can be found in Linear Programming methods and application by S.I. Gass [32,pp. 469-529]. The increasing capacity of computer permitting the designing of larger and larger linear programming models had led to the rapid expansion in the size of LP models and to significant practical improvements in solution methods. This is one of the reasons that a good deal of scientific effort has been devoted to developing fast and robust algorithms in recent past.

It is interesting to note that despite its wide applicability to every day real life problems, linear programming was unknown prior to 1947. It is doubtless that the seminal potential of LP technique could be seen reflected in the work of couple of mathematicians, for example, Fourier in 1823 [23a] and de la Valle Poussin in 1911 [94a]. But these works were taken as isolated cases only. They were soon forgotten. In 1939,

L.V. Kantrovich [57] published an extensive monograph entitled "Mathematical methods in organisation and planning of production". Thus Kantrovich should be credited with being the first to recognise certain broad class of production problems having well-defined mathematical structures. He believed that they were amenable to practical numerical evaluation i.e., could be numerically solved. Unfortunately, Kantrovich's earlier efforts did not receive much attention in the beginning. His earlier work remained unknown both in USSR and the rest of the world for nearly 20 years. It was only some major developments had taken place in mathematical programming technique that Kantrovich paper [57] became known sometimes in 1959.

1.1.4 Mathematical formulation of Linear Programming Problem

Maximize (minimize) $Z(x) = c_1x_1 + c_2x_2 + \ldots + c_rx_r$

subject to constraints:

$a_{11}x_1+a_{12}x_2 + a_{1r}x_r(\leq,=,\geq)\ b_1$

$a_{21}x_1+a_{22}x_2 + a_{2r}x_r(\leq,=,\geq)\ b_2$

.

.

.

$a_{r1}x_1+a_{r2}x_2 + a_{1r}x_r(\leq,=,\geq)\ b_r$

$x_1, x_2, \ldots, x_r \geq 0.$

On introducing slack and surplus variables (which converts inequality into equality constraints) to above linear programming problem, we get the standard form in matrix notations as follows:

$$\text{Minimize } Z(x) = c^Tx$$

$$\text{Subject to } Ax = b \qquad (P)$$

$$x \geq 0$$

where A is an mxn matrix, c and x are n-dimensional vectors i.e, c, $x \in \mathbb{R}^n$, b is an m-dimensional vector i.e, $b \in \mathbb{R}^m$ and r+m=n.

In the year 1947, Gerogre B. Dantzig formulated this problem and proposed the simplex method to solve it. Since then various modifications in the simplex method have been proposed. We propose to discuss this method and various modifications in this chapter. Sethi and Thompson [100] has also proposed a variant of the simplex method known as Pivot and Probe Algorithm (PAPA) for solving linear programming problems.

We propose some modifications in this variant which ensures early termination of the algorithm. Some computational results indicating early termination due to this modification have also been discussed.

Before harping on the problem related to obtaining solution to linear programming problem (P.), we give some definitions and introduce some notations.

1.2 SOME DEFINITIONS

1. **Requirement Vector**: If we consider the linear programming problem (P) in the standard form as:

   minimize $c^T x$

   subject to $Ax = b$

   $x \geq 0$,

   Then the vector $b \in \mathbb{R}^m$ is called the requirement vector.

2. **Objective Function**: The linear function $c^T x$ in the above LP problem (P) is called the objective function.

3. **Activities:** Elements a_{ij} of the constraint matrix $A=(a_{ij})$ are called the price vector

   The matrix $A\in\mathbb{R}^{mxn}$ is called the activity vector.

4. **Price Vector:** The $c\in\mathbb{R}^n$ used in the description of the objective function c^Tx of the LP problem (P) is called price vector.

5. **Positive orthant:** The subset $\{x:x\in\mathbb{R}^n, x\geq 0\}$ of Euclidean space $\mathbb{R}^n$ is called positive orthant.

6. **Feasible Solution (FS):** A feasible solution to a linear programming is the set of values of the variables which satisfies the set of constraints together with the non-negativity restrictions.

7. **Basic Solution(BS):** In the constraint equation Ax=b, the matrix is of order mxn and n>m. Let B be a nonsingular square submatrix of order n. The set of the m variables associated with the m columns of B, together with zeros for the remaining (n-m) variables is a solution for Ax = b, called a Basic solution, $x_B=B^{-1}b$ and $x = (x_B,0)$.

   The components of x_B are called the *basic variables.*

8. **Basic Feasible Solution (BFS):**

   A feasible solution to the constraints of the programme which is also basic is said to be a Basic Feasible Solution.

9. **Optimal Feasible and Optimal Basic Feasible Solution:**

   If a feasible solution to the constraints that achieves the minimum value of the objective function subject to those constraints is called an optimal feasible solution. If this solution is also basic, it is an

Optimal Basic Feasible Solution.

10. **Non-Degenerate BFS:** A BFS of an LP problem is said to be non-degenerate BFS if none of its basic variables is zero.

11. **Degenerate BFS:** A BFS of an LP problem is said to be degenerate BFS if at least one of the basic variables is zero.

12. **Slack Variables:** The positive variables which are added to left hand sides of the constraints of the type $a_{i1}x_1+a_{i2}x_2+...+a_{in}x_n \leq b_i$ to convert them into equalities are called the slack variables.

13. **Surplus Variables:** The positive variables which are subtracted from the left hand sides of the constraints of the type $a_{j1}x_1+a_{j2}x_2+...+a_{jn}x_n \geq b_j$ to convert them to equalities are called the surplus variables.

14. **Convex Set**: A set of points is said to be convex if for any two points in the set, the line segment joining these two points is also in the set. Equivalently, a set S is called a convex set if $x^{(1)}, x^{(2)} \in S \rightarrow \lambda x^{(1)} + (1-\lambda)\, x^{(2)} \in S$ where $0 \leq \lambda \leq 1$

15. **Extreme Point of a Convex Set:** A point x in a convex set C is called an Extreme point if x cannot be expressed as a convex combination of any two distinct points x_1 and x_2 in C.

16. **Hyperplane:** A Hyperplane is defined as the set of points satisfying $c_1x_1+c_2x_2+...+c_nx_n=Z(x)$ (not all $c_i=0$) for prescribed values of $c_1, c_2, ..., c_n$ and Z.

1.3 THE SIMPLEX METHOD

1.3.1 Introduction

As a part of pre-history, one can see that a naive approach for solving problems of type (P) may be to exploit the familiar technique of differential calculus viz., solving the equations obtained by putting the partial derivatives of the objective function equal to zero. But this technique does not work because of the constraints. Even the standard Lagrange multiplier technique can not apply directly to this problem because we have to deal with inequality constraints and this technique only handles equality constraints. Thus, none of the classical methods in differential calculus or linear algebra offers any adequate technique for solving linear programming problems or systems of linear inequalities. In 1947, George B. Dantzig devised a method, known as the simplex method to solve this problem. We discuss this method first.

1.3.2. Some Notations

1. We shall denote the columns of A by α_i. i.e, $A = (\alpha_1, \ldots, \alpha_n)$
2. We shall denote by B, an m x m non-singular square submatrix of A whose column vectors are linearly independent. If we denote the columns of B by β_i. Then $B=(\beta_1, \beta_2, \ldots, \beta_m)$ is called a Basic Matrix.
3. The variables corresponding to $\beta_1, \beta_2, \ldots, \beta_m$ are called the basic variables and are denoted by $x_{B1}, x_{B2}, \ldots, x_{Bm}$ respectively. The vector of these m variables is denoted by x_B i.e., $x_B=(x_{B1}, x_{B2}, \ldots, x_{Bm})$ where $x_B=B^{-1}b$ and is the BFS to the LP problem (P).

4. The coefficients of the basic variables $x_{B1},x_{B2},...,x_{Bm}$ in the objective function Z(x) will be denoted by $c_{B1},c_{B2},...,c_{Bm}$, so that $c_B=(c_{B1},c_{B2},...,c_{Bm})$.

5. Since $B=(\beta_1,\beta_2,...,\beta_m)$ is a non-singular matrix of order mxm , and the vectors $\beta_1,\beta_2,...,\beta_m$ are linearly independent, they form a basis of $\mathbb{R}^m$. So, every vector in$^m\mathbb{R}$ can be expressed as a linear combination of vectors in B.

Let $\alpha_j = \beta_1 y_{ij}+\beta_2 y_{2j}+...,\beta_m y_{mj}$

$$= (\beta_1,\beta_2,...,\beta_m).\begin{bmatrix} y_{1j} \\ y_{2j} \\ \vdots \\ y_{mj} \end{bmatrix} \quad BY_j$$

Where $y_{1j},y_{2j},...,y_{mj}$ are scalars required to express α_j and

$$Y_j = \begin{bmatrix} y_{1j} \\ y_{2j} \\ \vdots \\ y_{mj} \end{bmatrix}, \quad j=1,2,3,...,n+m$$

6. We shall write

$$Z_j=C_BY_j=C_By_{1j}+C_{B2}y_{2j}+...+C_{Bm}y_{mj}.$$

1.3.3. Some important theorems

Theorem 1.1

The set of all feasible solutions to a linear programming problem (P) is a convex set.

Theorem 1.2

Every BFS is an extreme point of the convex set of feasible solutions.

Theorem 1.3 Fundamental Theorem of Linear Programming

If an LP problem

$$\text{minimize } Z = C^T x$$

$$\text{s.t.} \quad Ax = b$$

$$x \geq 0$$

has an optimal solution, then at least one basic feasible solution must be optimal.

Theorem 1.4 To determine improved BFS

Let $x_B = B^{-1}b$ be a BFS of an LPP with $Z = c_B x_B$ as the value of the objective function. If for any column α_j in A, but not in B, the condition $c_j - Z_j < 0$ holds, and if at least one $y_i > 0$, i=1,2,...,m, then it is possible to obtain a new BFS by replacing one of the columns in B by α_j and if the new value of the objective function be Z`, then $Z` \leq Z$. If the given BFS is non-degenerate, then $Z` < Z$.

Simplex method starts with a BFS and continues to improve until the optimality is reached.

Theorem 1.5 Optimality Conditions

Let $x_B = B^{-1}b$ be the basic feasible solutions of the LPP minimize $c^T x$, s.t. $Ax=b$, $x \geq 0$ and $Z^* = c_B x_B$ be the value of the objective function for this BFS If $c_j - Z_j \geq 0$ for every column α_j in A but not in B, then Z^* is the objective function Z and this BFS x_B is an optimal BFS

This theorem ensures that the optimality is reached.

Theorem 1.6 Conditions for Alternative Optimal Solutions

Let there be an optimal BFS of an LP problem (P). Then

(i) If for some α_j in A but not in B, $c_j-Z_j=0$; $y_{ij}\leq 0$ for all i=1,2,...,m, then a non-basic alternative optimum will exist.

(ii) If for some α_j in A but not in B, $c_j-Z_j=0$ and $y_{ij}>0$ for at least one i, then an alternative basic optimum will exist.

The proofs of the theorems may be found in any standard text book on linear programming.

1.3.4. Revised Simplex Method

The revised simplex method is an efficient computational procedure for solving linear programming problems on digital computers. The revised simplex method solves a linear programming problem in the same way as the simplex method but computes and stores only the information that are of current need.

1.4. DUALITY IN LINEAR PROGRAMMING

1.4.1 Introduction:

The profit earned by one organisation is the loss to some other organisation. The profit making organisation attempts to maximize the profit whereas the loosing organisation attempts to minimize the loss. Both organisations work in the same environment; that is, the data of the two problems are the same. Thus, the optimization problem always occur in pair - a profit maximization problem and a loss minimization problem. If one of these problems is called primal problem, the other is called dual

problem. The duality theory in linear programming deals with such a pair of primal-dual problems.

The fundamental notion of duality was first introduced by Von Neumann [128]. Subsequently Gale, Kuhn and Tucker [30] formulated an explicit duality theorem which they proved by using classical lemma of Farkas [22] (Farkars' Lemma: Exactly one of the following statement holds:

(i) $\exists\ x \in \mathbb{R}^n$ with $Ax \leq b$ (ii) $\exists\ w \in \mathbb{R}^m$ with $w^T A=0\ w^T b \leq -1$). A systematic presentation of theoretical properties of dual linear programming problem can be found in Goldman and Tucker [39] and Gale [29]. A review of Von Neumann's contribution can be found in Kuhn [70].

Symmetric form of Dual :

Primal problem	Dual problem
minimize $c^T x$	maximize $b^T w$
subject to $Ax \geq b$	Subject to $A^T w \leq c$
$x \geq 0$	$w \geq 0$

Unsymmetric form of the Duality relation :

Primal problem	Dual problem
minimize $c^T x$	Maximize $b^T w$
subject to $Ax=b$	Subject to $A^T w \leq c$
$x \geq 0$	w unrestricted in sign

In this form the dual vector w is not restricted to be non-negative.

The intimate relationship between the primal and the dual problems gives rise to some very useful "joint properties" enunciated by the following theorems.

Theorem 1.7

Dual of the dual of a given primal is the primal itself.

The proof of the theorem is based on the reversible construction procedures from the primal to the dual.

As the primal and dual problems are closely connected, their solutions will also be related as to their feasibility, boundedness and optimality. We are going to obtain these relations through the following theorems.

We consider the primal problem in the form

Minimize $c^T x$

subject to $Ax \geq b$

$x \geq 0$ (P)

and its corresponding dual as

Maximize $b^T w$

subject to $A^T w \leq c$ (D)

$w \geq 0$

Theorem 1.8

If x is any feasible solution of the primal (P) and w is any feasible solution to its dual problem (D) then $c^T x \geq b^T w$

Proof:

We have

$$A^T w \leq c \text{ and } x \geq 0$$

So, we get $w^T A \leq c^T$

$$w^T A x \leq c^T x$$

$$w^T b \leq c^T x$$

$$b^T w \leq c^T x$$

This theorem shows that a feasible solution to either problem implies a bound on the value of the other value.

Theorem 1.9

If $\hat{x}$ and $\hat{w}$ are feasible solutions for the primal problem (P) and the dual problem (D) respectively, and if $c^T\hat{x} = b^T\hat{w}$, then $\hat{x}$ and $\hat{w}$ are optimal for their respective problems.

Proof. From the above theorem,

$$c^T x \geq b^T \hat{w}$$

or $c^T x \geq c^T \hat{x}$ since $c^T \hat{x} \geq b^T w$

The value of the objective function of the primal at any other feasible solution is greater than that at $\hat{x}$. So, $\hat{x}$ is the optimal solution for the primal.

Again,

$$b^T w \leq c^T \hat{x}$$

Or, $b^T w \leq b^T \hat{w}$

So is the optimal solution for the dual problem. Hence proved.

The above theorem shows that if a pair of feasible vectors can be

found to the primal and dual problems for which the objective values are equal, then these are both optimal. The converse is also true and is stated by the following theorem.

Theorem 1.10: The Fundamental Duality Theorem

If either of the primal problem (P) and dual problem (D) has a finite optimal feasible solution, the other has also a finite optimal feasible solution and the corresponding values of the two objective functions are equal. If either problem has an unbounded objective, the other has no feasible solution.

Proof:

The proof of the theorem can be found in Luenberger [73].

Now, we investigate the interrelationship between the sets of variables of the primal and the dual problems.

Theorem 1.11

If any variable of the primal is unrestricted in sign, the corresponding constraint in the dual will be a strict equality.

Proof:

If x_k of the primal (P) is unrestricted in sign or a free variable, we replace it by $x_{k1} - x_{k2}$ where x_{k1}, x_{k2} are both non-negative. Then in the objective function of the primal corresponding to $c_k^T(x_{k1} - x_{k2})$, we get in the dual objective function,

$$\sum a_{ik} w_i \geq c_k$$

$$-\sum a_{ik} w_i \geq -c_k$$

And these two together equivalent to $\sum a_{ik} w_i = c_k$. Hence the theorem.

Complementary Slackness Theorems

Theorem 1.12 (Complementary Slackness for Unsymmetric Form)

Let x be a feasible solution for the primal problem

minimize $c^T x$

subject to $Ax = b$

$x \geq 0$

and w be a feasible solution for the dual problem

maximize $b^T w$

subject to $A^T w \leq c$

w is unrestricted in sign.

A necessary and sufficient condition that they both are the optimal solutions is that for all i,

(i) $x_i > 0 \Rightarrow a_i^T w = c_i$

(ii) $x_i = 0 \Leftarrow a_i^T w < c_i$

Proof: If the conditions hold,

$(A^T w - c)x = 0$

$(w^T A - c^T)x = 0$

$w^T Ax - c^T x = 0$

$w^T b - c^T x = 0$

$w^T b = c^T x$

Conversely, it x anu w are optimal,

$$b^T w = c^T x$$

$$w^T b = c^T x$$

$$w^T A x - c^T x = 0$$

$$(w^T A - c^T)x = 0$$

$$(A^T w - c)x = 0$$

Hence $x \geq 0$ and $A^T w - c \leq 0$

so, condition (i) and (ii) hold.

Theorem 1.13 **(Complementary slackness for symmetric form)**

For the optimal feasible solutions of the primal and dual system.

(a) Whatever inequality occurs in the ith relation of either system, if the corresponding slack or surplus variable is positive, then the ith variable of its dual is zero.

(b) If the jth variable is positive in either system the ith relation of its dual holds as a strict equality. Here, we observe that a necessary and sufficient condition for any pair of feasible solutions to the primal and dual to be optimal is that:

$$w_i x_{n+i} = 0, \; i=1,2,...,m$$

where x_{n+i} is the slack variable in the primal and

$$x_j w_{m+j} = 0, \; j=1,2,...,n$$

where w_{m+j} is the surplus variable for the dual.

Proof: Can be found in Luenberger [73].

1.4.2 Dual Simplex Algorithm

The optimality criterion of the simplex method for a minimizing linear programming problem is

$$r_j=c_j-Z_j=c_j-c_BB^{-1}a_j \geq 0$$

for all j where B is a basis. The criterion is independent of the requirement vector b. So every basic solution with all $r_j \geq 0$ is certainly an optimal solution but may not be feasible. In the simplex tableau this situation corresponds to having no non-negative elements in the bottom row but an infeasible basic solution.

The dual simplex algorithm starts with a basic optimal solution of the primal, not necessarily feasible and decreases the number of negative variables iteratively maintaining the optimality criterion at each iteration.

The technique of dual simplex algorithm was discovered by C.E. Lemke [72].

Given the Linear programming problem

minimize c^Tx

subject to $Ax=b$

$$x \geq 0$$

Let a basis B be known such that $w = c_BB^{-1}$ is feasible for the dual. In such a case the basic solution $x_B = B^{-1}b$ to the primal is dual feasible. If $x_B \geq 0$, then this solution is also primal feasible and hence optimal.

Now, w is feasible for the dual,

Let us assume $a_j^T w = c_j$, for j=1,2,...,m

and $a_j^T w < c_j$ for j=m+1,...,n, (assuming non-degeneracy)

Let us find a new vector $\bar{w}$ such that one of the equalities becomes an inequality, and the value of the dual objective function is increased. The m equalities in the new solution then determine a new basis.

Let u^l be the ith row of B^{-1} and we set $\bar{w}=w - \epsilon u^l$. Then

$$a_j^T \bar{w} = a_j^T w - \epsilon a_j^T u^l.$$

We also have

$$Z_j = a_j^T w \text{ and } a_j^T u^l = y_{lj},$$

the (i, j)th element of the tableau and

$$a_j^T \bar{w} = c_j, \quad j=1,2,...,m, i \neq j$$

$$a_l^T \bar{w} = c_l - \epsilon$$

$$a_j^T \bar{w} = Z_j - \epsilon y_{lj}, \quad j=m+1,m+2,...,n$$

Also $b^T \bar{w} = b^T w - \epsilon x_{Bl}$

These lead to the following algorithm.

Algorithm.

Step 1: Given a dual feasible basic solution x_B, if $x_B \geq 0$, the solution is optimal. If x_B is not non-negative, we select i such that $x_{Bi} < 0$.

Step 2: If all $y_{ij} \geq 0$, j=1,2,...,n, then the dual has no maximum, since $\bar{w}$ is feasible for all $\epsilon > 0$. If $y_{ij} < 0$ for some j, then let

$$\epsilon_o = \frac{Z_k - c_k}{y_{ik}} = \min_j \left\{ \frac{Z_j - c_j}{y_{ij}} \; ; \; y_{ij} < 0 \right\}$$

Step 3. A new basis is formed by replacing a_j by a_k. Using this basis to determine the corresponding basic dual feasible solution x_B and return to step 1.

Advantage of Dual Simplex Algorithm

The advantage of the dual simplex algorithm over the other method is that here we do not require any artificial variables. Hence a great deal of work is saved whenever this method is applied.

1.4.3 Primal - Dual Algorithm

When artificial variables are necessarily introduced, we use either two phase method or Charne's M-method to remove the artificial variables first and then we proceed for optimality. This may require a large number of iterations to get the optimal solution in phase II. To reduce the number of iterations considerably we can solve linear programming problems by working simultaneously on the primal and the dual problems. We start with a feasible solution to the dual and improve it at each step by optimizing an *associated restricted primal* problem. The method progresses with an aim to achieve the complementary slackness conditions for optimality.

Although, the dual simplex algorithm may be used in some cases to remove the necessity to introduce artificial variables, but it can not be used in general, because a basic solution with all $r_j \geq 0$ may not be always

available easily. Thus, primal-dual method can solve larger class of problems in comparison to the dual simplex method.

1.5 PIVOT AND PROBE ALGORITHM (PAPA)

Sethi and Thompson [100,101] proposed a variant of the simplex algorithm known as pivot and probe algorithm (PAPA) to solve linear programming problems. In this algorithm we first search for *candidate constraints*, constraints with the smallest positive intercepts on at least one of the co-ordinate axes and form the *relaxed LP* containing only candidate constraints. After obtaining the relaxed LP, we find the optimal solution of this relaxed problem by the simplex method and the solution gives an *upper bound* to the optimal objective function value of the original LP problem. We use the compact form of the simplex table as described in Garfinkel and Nemhauser [31,p. 31]. The initial basic feasible solution to the relaxed LP is taken as $x = 0$.

1.5.1 Mathematical Formulation of the Optimal Table for Relaxed LP

We consider a linear programming problem

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{such that } Ax \le b \qquad\qquad (P) \\ &\qquad x \ge 0 \end{aligned}$$

where $x, c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$.

Without any loss of generality, we can assume that $b \ge 0$ which makes $x=0$ a basic feasible solution.

After introducing slack variables, x_{n+i}, $i=1,2,...,m$ to the ith

constraint, the problem reduces to the following form

$$\text{minimize } Z = \sum_{j=1}^{n+m} c_j x_j$$

$$\text{such that } \sum_{j=1}^{n+m} a_{ij} x_j = b_i,\ i=1,2,\ldots,m,$$

where $a_{ij}=0$ or 1 according as $j > n+i$ (1.1)

or $j=n+i$ and $c_j=0$ for $j > n$.

To form the relaxed LP, first of all we search the candidate constraints. The ith constraint is a candidate constraint if

$$\frac{b_i}{a_{ij}} = \min_{1\le h\le m} \left\{ \frac{b_h}{a_{hj}},\ a_{hj} > 0 \right\},\ j=1,2,\ldots,m+n$$

The objective function of LP (1.1) together with these candidate constraints and non-negativity restrictions constitutes the relaxed LP.

Sometimes even for a bounded LP problem, the corresponding relaxed problem may be unbounded. In order to overcome this situation, it has been suggested in Sethi and Thompson [100] to introduce a regularisation constraint of the form

$$\sum_{j=1}^{n} x_j \le M \qquad (1.2)$$

where M is a very large number. We observe that if b_k denote the maximum of all b_i's then the regularisation constraint (1.2) can be replaced by

$$\sum_{j=1}^{n} x_j \le b_k.$$

It is important to note that b_k may be the smallest of all the components of requirement vector b.

Let us suppose that the relaxed LP consists of r-1 constraints then the

remaining m-r (or m-r+1 in the case of using regularisation constraint) constraints have been left out for searching the most violated constraints. Following Garfinkel and Nemhauser [31], we consider the optimal table for relaxed LP in the following form:

Table 1

	..... $-x_j$	..... $-x_k$
x_o	y_{oo}	y_{oj}	y_{ok}
.			
.			
.			
x_i	y_{io}	y_{ij}	y_{ik}
.			
.			
.			

where $x_o = y_{oo}$ is the optimal value of the objective function corresponding to the relaxed LP. x_i's are basic variables, x_j's are non basic variables. The explication of the table -1 can be found in Garfinkel and Nemhauser [31].

After obtaining the optimal solution for the relaxed LP, we have to search for the most violated constraint. To obtain it we form the line segment joining the initial basic feasible solution to the optimal basic feasible solution of the relaxed LP and find all the points of intersection of this line segment with non candidate constraints and consider the one which is inside the feasible region of the original problem. These points are known as the *piercing points*. The non candidate constraint which contains the piercing point is called the most violated constraint. This most violated constraint is incorporated at the bottom of the optimal table for relaxed LP and dual-simplex algorithm is performed.

A situation may arise where we get more than one non candidate

constraints which are most violated constraints. Now, our aim is to specify the most violated constraint whose introduction into the optimal table of the relaxed LP, will ensure the largest improvement in the value of the objective function. This constraint is termed as the specific most violated constraint.

1.5.2 A Proposed New Rule for Selecting the Specific Most Violated Constraint

We consider I^c and ^NI as the sets of indices of candidate constraints and non candidate constraints respectively. Then the set I of indices of all constraints can be written as $I = I^c \cup I^N$.

Sethi and Thompson describes [100] a rule for selecting the most violated constraint among the class of non candidate constraints as follows:

The h th constraint is the most violated constraint if

$$\lambda_h = \min_i \{ \lambda_i : i \in I^N \} \qquad \ldots \qquad (1.3)$$

where λ_i is obtained by

$$\lambda_i = \frac{b_i - a_i x'}{a_i \hat{x} - a_i x'}, \quad a_i \hat{x} > b_i \qquad \ldots \qquad (1.4)$$

Here x' and $\hat{x}$ are the initial and optimal basic feasible solutions to the relaxed LP.

If the minimum value of λ_h obtained from (1.3) occurs for more than one values of h, say, $h_1, h_2, \ldots\ldots, h_i, \ldots, h_s, \ldots$, then constraint corresponding to these indices are the most violated constraints. So, a class of most violated constraints is obtained.

Sethi and Thompson [100] did not provide any rule for selecting a particular most violated constraint from the class of the most violated constraints, which can ensure early termination of the algorithm.

Now, we propose a new rule to identify a particular most violated constraint which can ensure early termination of the algorithm. We name this particular most violated constraint as the *specific most violated constraint.*

88If I^V denotes the set of indices of the most violated constraints am88ong the class of the most violated constraints then the members of the class can be written as

$$\sum_{j=1}^{n+m} a_{h_j} x_j = b_h \,,\ h \in I^V = \{h_1, h_2, \ldots, h_i, \ldots, h_s, \ldots\}$$

i.e., $x_{n+h} = b_h - \sum_{j=1}^{n} a_{hj} x_j$.

In order to use these constraints at the bottom of the optimal table -1 for relaxed LP, we have to transform basic variables in terms of non basic variables. If J^B and J^N denote the sets of indices of basic and non basic variables respectively, the set J of indices of all variables can be written as $J = J^B \cup J^N$. Then the members of the class of the most violated constraints are written as

$$x_{n+h} = b_h - \sum_{i \in J^B} a_{hj} x_i - \sum_{j \in J^N} a_{hj} x_j \,,\ h \in I^V. \qquad \ldots (1.5)$$

To transform basic variables in terms of non basic variables, we obtain basic variables x_i's from table -1 as follows:

$$x_i = y_{io} - \sum_{j \in J^N} y_{ij} x_j \,, \; i \in J^B \qquad \ldots \qquad (1.6)$$

Using (1.6) in (1.5), we get

$$x_{n+h} = b_h - \sum_{i \in J^B} a_{hi} \left(y_{io} - \sum_{j \in J^N} y_{ij} x_j \right) - \sum_{j \in J^N} a_{hj} x_j$$

i.e., $$x_{n+h} = \left(b_h - \sum_{i \in J^B} a_{hi} y_{io} \right) - \sum_{j \in J^N} \left(a_{hj} - \sum_{i \in J^B} a_{hi} y_{ij} \right) x_j \quad \ldots \qquad (1.7)$$

The most violated constraint (1.7) is added at the bottom of table 1, and the resulting table takes the form as given table - 2.

Table - 2

	..... $-x_j$	..... $-x_k$
x_o	y_{oo}	y_{oj}	y_{ok}
.			
.			
x_i	y_{io}	y_{ij}	y_{ik}
.			
.			
x_{n+h}	$(b_h - \sum_{i \in J^B} a_{hi} y_{io})$	$(a_{hj} - \sum_{i \in J^B} a_{hi} y_{ij})$	$(a_{hk} - \sum_{i \in J^B} a_{hi} y_{ik})$

Now, Table -2 is dual feasible, but not a primal feasible, as $(b_h - \sum_{i \in J^B} a_{hi} y_{io}) < 0$.

We know that the solution in table-2 is optimal if it is primal feasible as well as dual feasible. So we use simplex algorithm to make table-2 primal feasible. The rules for selecting the outgoing vector and incoming vector in dual simplex algorithm are as follows:

x_r is the outgoing vector if

$$y_{ro} = \min_{i \in J^B} \{ y_{io} \}$$

and x_k is the incoming vector if

$$\frac{y_{ok}}{y_{rk}} = \max_{j \in J^N} \left\{ \frac{y_{oj}}{y_{rj}} , y_{rj} < 0 \right\}.$$

According to this rule x_{n+h} is the outgoing vector. In order to obtain incoming vector we have to calculate

$$\max_{j \in J^N} \left\{ \frac{y_{oj}}{\left(a_{hj} - \sum_{i \in J^B} a_{hi} y_{ij} \right)}, \left(a_{hj} - \sum_{i \in J^B} a_{hi} y_{ij} \right) < 0 \right\}$$

If we consider

$$\theta_{hj} = \frac{y_{oj}}{\left(a_{hj} - \sum_{i \in J^B} a_{hi} y_{ij} \right)} , h \in I^V$$

and $\theta_{hk_1} = \max_{j \in J^N} \frac{y_{oj}}{\left(a_{hj} - \sum_{i \in J^B} a_{hi} y_{ij} \right)} = \frac{y_{ok_1}}{\left(a_{hk_1} - \sum_{i \in J^B} a_{hi} y_{ik_1} \right)}$

Then

$$\theta_{h_1 k_1} = \max_{j \in J^N} \frac{y_{oj}}{\left\{ a_{h_1 j} - \sum_{i \in J^B} a_{h_1 i} y_{ij} \right\}} = \frac{y_{ok_1}}{\left(a_{h_1 k_1} \sum_{i \in J^B} a_{h_1 i} y_{ik_1} \right)}$$

....

....

$$\theta_{h_t k_t} = \max_{j \in J^N} \frac{y_{oj}}{\left\{ a_{h_t j} - \sum_{i \in J^B} a_{h_t i} y_{ij} \right\}} = \frac{y_{ok_t}}{\left(a_{h_t k_t} - \sum_{i \in J^B} a_{h_t i} y_{ik_t} \right)}$$

....

....

$$\theta_{h_s k_s} = \max_{j \in J^N} \frac{y_{oj}}{\left\{ a_{h_s j} - \sum_{i \in J^B} a_{h_s i} y_{ij} \right\}} = \frac{y_{ok_s}}{\left(a_{h_s k_s} - \sum_{i \in J^B} a_{h_s i} y_{ik_s} \right)}$$

.... (1.8)

This shows that if h_s th constraint is chosen as the most violated constraint from the class of the most violated constraints then h_s th row is the pivot row whereas the k_s th column is the pivot column in table-2. In this case $(a_{h_s k_s} - \sum_{i \in J^B} a_{h_s i} y_{i k_s})$ is the pivot element. After performing dual simplex iteration in table-2, the value of the objective function y_{oo} changes to

$$\left\{ y_{oo} - \frac{y_{ok_s}(b_{h_s} - \sum_{j \in J^B} a_{h_s i} y_{io})}{(a_{h_s k_s} - \sum_{j \in J^B} a_{h_s i} y_{ik_s})} \right\}$$ which is equal to

$$\{y_{oo} - \theta_{h_s k_s}(b_{h_s} - \sum_{i \in J^B} a_{h_s i} y_{io}\} \quad \ldots \quad (1.9)$$

As the value of the objective function decreases after performing dual simplex iteration, we have to minimize the expression (1.9) by suitable selection of $h_s \in I^v$. That is, we have to search for those values of h_s and k_s which make the expression (1.9) minimum; because we want to get the largest decrease in the value of the objective function, after performing dual simplex iteration in table-2.

If $y_{oo} - \theta_{h_i k_t}(b_{h_i} - \sum_{i \in J^B} a_{h_i i} y_{io})$

$$= \min_{h_s \in I^v} \left\{ y_{oo} - \theta_{h_s k_s} \left(b_{h_s} - \sum_{i \in J^B} a_{h_s i} y_{io} \right) \right\}$$

Then $\theta_{h_t k_t}\left(b_{h_t} - \sum_{j \in J^B} a_{h_t i} y_{io}\right) = \max_{h_s \in I^V} \theta_{h_s k_s}\left(b_{h_s} - \sum_{i \in J^B} a_{h_s i} y_{io}\right)$

We consider $\phi_{h_s k_s} = \theta_{h_s k_s}\left(b_{h_s} - \sum_{i \in J^B} a_{h_s i} y_{io}\right)$ ... (1.10)

Then $\phi_{h_t k_t} = \theta_{h_t k_t}\left(b_{h_t} - \sum_{i \in J^B} a_{h_t i} y_{io}\right) = \max_{h_s \in I^V} \left\{\phi_{h_s k_s}\right\}$ ... (1.11)

After the selection of h_t and k_t from (1.11) we suggest that h_t th constraint is the specific most violated constraint. That is, x_{h_t} is the pivot row whereas x_{k_t} is the corresponding pivot column in table-2.

Now, we claim that using this specific most violated constraint, the algorithm terminates early in comparison to an arbitrary selection of the most violated constraint.

Proof of the claim:

From (1.11), we get

$$\phi_{h_t k_t} = \max_{h_s \in I^V} \{\phi_{h_s k_s}\}$$

i.e., $\phi_{h_t k_t} > \phi_{h_s k_s}$ for all $h_s \in I^V - \{h_t\}$; $k, k_s \in J^N$.

i.e., $\theta_{h_t k_t}\left(b_{h_t} - \sum_{i \in J^B} a_{h_t i} y_{io}\right) > \theta_{h_s k_s}\left(b_{h_s} - \sum_{i \in J^B} a_{h_s i} y_{io}\right)$

i.e., $y_{oo} - \theta_{h_t k_t}\left(b_{h_t} - \sum_{i \in J^B} a_{h_t i} y_{io}\right) < y_{oo} - \theta_{h_s k_s}\left(b_{h_s} - \sum_{i \in J^B} a_{h_s i} y_{io}\right)$

i.e., $y_{oo} - \dfrac{y_{ok_t}(b_{h_t} - \sum_{i \in J^B} a_{h_t i} y_{io})}{(a_{h_t k_t} - \sum_{i \in J^B} a_{h_t i} y_{ik_t})} < y_{oo} - \dfrac{y_{ok_s}(b_{h_s} - \sum_{i \in J^B} a_{h_s i} y_{io})}{(a_{h_s k_s} - \sum_{i \in J^B} a_{h_s i} y_{ik_s})}$

Which shows that the objective function obtained by using h_tth constraint, the specific most violated constraint (SMVC), is *less* than the

ADU ZARIA UNIVERSITY

objective function obtained by any other most violated constraint from the class of the most violated constraint. It indicates that the specific most violated constraint takes less time to solve LP by PAPA than any other most violated constraint.

Using this specific most violated constraint and performing dual simplex iteration, we find optimal solution in less number of iterations than an arbitrary selection of the most violated constraint.

A computer programme has been developed to implement this modified method given in the Appendix.

.5.3 Implementation of the Rule to Solve a Numerical Problem

The Rule for searching the specific most violated constraint is presented below:

Step 1: To obtain optimal table 1 for relaxed LP which gives optimal solution $\hat{x}$.

Step 2: To obtain λ_l using (1.4) and λ_h by (1.3). If λ_h is unique, then hth constraint is the most violated constraint. If not, to go to step 3.

Step 3: Let $\lambda_h = \lambda_{h_1}, \quad \lambda_{h_2},\ldots,\lambda_{h_t},\ldots,\lambda_{h_s},\ldots$

To find $\theta_{h_1 k_1}, \quad \theta_{h_2 k_2},\ldots,\theta_{h_t k_t},\ldots,\theta_{h_s k_s},\ldots,\ldots$ using (1.8).

Step 4: To obtain $\phi_{h_s k_s}$ using (1.10) for all $h_s \in I^V$.

Step 5: To obtain $\theta_{h_t k_t}$ using (1.11) and select the corresponding values of h_t and k_t. Then h_t th constraint is the specific most violated constraint.

Example:

Maximize $Z = x_1 + x_2$

such that $-x_1 + 2x_2 < 12$

$2x_1 + x_2 \leq 12$

$x_1 + x_2 \leq 12$

$2x_1 + 3x_2 \leq 30$

$x_1, x_2 \geq 0$

After introducing the slack variables x_i, i=3,4,5,6, it becomes in standard form as follows:

Maximize $x_1 + x_2$

such that $-x_1 + 2x_2 + x_3 = 12$

$2x_1 + x_2 + x_4 = 12$

$x_1 + x_2 + x_5 = 12$

$2x_1 + 3x_2 + x_6 = 30$

$x_i \geq 0$ for i=1,2,3,4,5,6.

Selecting the candidate constraint, its relaxed LP is

Maximize $z = x_1 + x_2$

such that $-x_1 + 2x_2 + x_3 = 12$

$2x_1 + x_2 + x_4 = 12$

$x_i \geq 0$

which shows that

$I^c = \{1,2\}, I^N = \{3,4\}$, $x'(x_1,x_2) = (0,0)$.

Step 1. Optimal table-3 for relaxed LP (except the bottom row) is given below:

Table 3

		$-x_4$	$-x_3$
x_0	24	1	1
x_2	12	1/8	2/3
x_1	12	3/8	1/3
x_5	-12	-1	-1 (specific most violated constraint)

Table-3 shows that,

$\hat{x} = (x_1, x_2) \equiv (12,12)$, $J^B = \{1,2\}, J^N = \{3,4\}$

Step 2. Since $\lambda_i = \lambda_3$ and $\lambda_h = \lambda_3, \lambda_4$

Therefore, $I^V = \{3,4\} = \{h_1, h_2\}$.

Step 3. $\theta_{h_1k_1} = \theta_{33} = -1$, $\theta_{h_2k_2} = \theta_{43} = -\frac{3}{8}$

implies $k_1=3$, $k_2=3$

Step 4. $\theta_{h_1k_1} = \theta_{33} = 12$, $\theta_{h_2k_2} = \theta_{43} = -\frac{45}{4}$

Step 5. $\theta_{h_ik_i}=12$ for $h_i=3, k_i=3$ indicates that third constraint is the specific most violated constraint, which after transformation, is added at the bottom of table-3, performing dual simplex iteration in table-3, results table-4 (the optimal table for LP) as follows:

Table 4

		$-x_4$	$-x_5$
x_0	12	0	1
x_2	4	-1/3	2/3
x_1	8	1/8	1/8
x_3	12	1	-1

By using the Rule of SMVC, we have obtained the optimal solution only in 3 iterations (2 iterations to get table 3 and then 1 iteration) whereas on selecting the most violated constraint arbitrarily, it takes 4

iterations. Thus the use of the Rule of SMVC saves one iteration in this problem.

This example has been solved using the computer programme developed in the Appendix. The programme can be able to solve problems with higher number of variables and constraints.

1.6 COMPUTATIONAL COMPLEXITY AND THE SIMPLEX ALGORITHM

The main computational steps in the simplex algorithm is the pivot step. Hence the efforts involved in solving a linear programming problem of the form

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{such that } Ax=b \qquad\qquad (P) \\ &x \geq 0 \end{aligned}$$

by the simplex algorithm can be measured by the number of pivots made before the algorithm terminates. If a feasible basis for (P) is not known, the simplex method applies the simplex algorithm twice, first on the phase I problem that decides whether (P) is feasible and provides a feasible basis for it if it is feasible and then work for phase II to achieve optimality. So, from the practical point of view, the computational effort required for solving a linear programming problem for which a feasible basis is not known, is approximately twice the effort required for solving a comparable problem in which a feasible basis is not available for starting the simplex

algorithm. Thus starting with a known feasible basis for (P), the number of pivot steps required before the simplex algorithm terminates, depends on m and n; the actual number of entries in matrices A, b, c as well as initial feasible basis used to initiate the algorithm. The number of pivot steps required before termination is expected to go up as m and n go up. When m and n are fixed then the number of pivot steps before termination depends on actual data in A, b and c. Now , a question arises pertaining to what is the average number of pivot steps required as a function of m, n and the size of the linear programming problem, before termination when the simplex algorithm is applied on a class of linear programming problems encountered in practical applications. This average number of pivot steps as a function of m, n and the size of linear programming problem is called *average computational complexity* of the simplex algorithm.

In fact, determination of the average computational complexity is a statistical question that is hard to answer precisely because the kind of linear programming problems that practical applications do generate cannot be easily specified. However, from the computational experience on the large number of linear programming problems solved over several years, an empirical answer to this question has emerged. It indicates that the average number of pivot steps required before the termination of the simplex algorithm is a linear function of m, and seems to be less than 3m.

In recent years some interesting papers have come up about the average speed of the simplex method. Dantzig [18, p.160] observes:

some believe that for a randomly chosen problem with fixed m (the number of constraints) the number of iterations grows in proportion to n (the number of variables). Smale [113] has proved this conjecture of Dantzig with a variant of the simplex algorithm, by using probabilistic model. Another important analysis in this direction has been carried out by Borgwardt in a series of papers [14 ,15]. He has proved that the number of iterations P(m, n) is a polynomial function of two variables m and n.

Recently Megiddo [78] has improved the estimate proposed by Smale [113] and showed that P(m, n) is bounded by a function of m only.

Now, the question arises - whether empirical answer emerged from computational experience remains valid for all the class of linear programming problems. This question has been answered in negative by Klee and Minty [62]. In fact, Klee and Minty [62] constructed an example with m = 2n inequalities, n being the number of variables. In order to solve this example, the simplex algorithm required exponential number of iterations. Therein, it was established that the simplex algorithm with steepest descent pivot rule would take exponential number of iterations to solve a linear programming problem. Since then it remained as an open question before Mathematicians, Operations Researchers and Computer Scientists - whether there is any non-exponential upper bound on the number of iterations to solve a linear programming problem. This question was answered in affirmative by the Russian mathematician L.G. Khachiyan [59] in the year 1979. He showed how a method of convex

optimization developed earlier by Russian mathematicians N.Z. Shor [105], D.B. Iudin and Nemirovskii in papers [54,55] can be adapted to devise a polynomially bounded algorithm for linear programming. This led to the inception of the Ellipsoid method. We propose to discuss this algorithm in chapter 2.

CHAPTER 2

ELLIPSOID METHOD

2.1 INTRODUCTION

Dantzig's simplex algorithm in its usual implementations has been an overwhelming successin the solution of real-world linear programming problems but these implementations are not polynomially bounded as Klee and Minty [62] gave the first example of an infinite family of linearly programming problems in which the number of simplex pivots grew like 2^n, while problem size grew like a polynomial function of n. Researchers were looking for an algorithm where the number of iterations may be bounded by a polynomial function in problem's dimension and input length.

In February, 1979 a note by L.G. Khachian [59], "A Polynomial Algorithm in Linear Programming" indicated how an ellipsoid method for the system of linear inequalities can be implemented to solve a linear programming problem in polynomial time. This result caused great excitement and stimulated a flood of technical papers.

Shor [103] gave the first completely explicit statement of the ellipsoid method as we know it. Khachian [59] gave a modified form of the ellipsoid algorithm and showed that feasibility and infeasibility could be determined in polynomial time by this algorithm.

The immediate significance of Khachian's article was the resolution of the important theoretical question concerning the computational complexity of linear programming. The algorithm given by Khachian indicated how the ellipsoid method for convex optimization developed by D.B. Iudin and A.S. Nemirovskii [54] be adapted to give a polynomial-time

algorithm for linear programming. This algorithm differs dramatically from the simplex method; it is not a pivoting method; it uses metric properties of $\mathbb{R}^n$ and it does not depend directly upon linearity of the objective function or the constraints.

2.2 SOME DEFINITIONS

2.2.1 Distance between two points in $\mathbb{R}^n$

If D is a positive definite matrix then for any two points X and Y in $\mathbb{R}^n$, the function $d(X,Y)=(X-Y)^T D^{-1}(X-y)$ is called the distance between x and y with matrix D^{-1} as the metric matrix. If $D=I_n$ (Identity matrix of order n) then the distance becomes Euclidean distance.

2.2.2 Positive Definite Matrix

A symmetric matrix A is said to be *positive definite* if the quadratic form x^TAx is positive for all non zero vectors x. Similarly, we define positive semidefinite, negative definite and negative semidefinite if $x^TAx \geq 0, < 0$ or ≤ 0 for all x.

2.2.3 Ellipsoid

The ellipsoid E in $\mathbb{R}^n$ with centre x_o is defined as the set

$E=\{x \in \mathbb{R}^n | (x-x_o)^T B^{-1}(x-x_o) \leq 1\}$, where B is a positive definite symmetric matrix of order n.

2.2.4 The Size of a Linear Program

For a real number p, ceiling of p is denoted by $\lceil p \rceil$ and is defined as the smallest integer $\geq p$. Now the number of digits is binary encoding an integer λ is $\lceil 1 + \log_2 (1+ |\lambda|) \rceil$.

Let us recapitulate the standard form of linear programming problem with integer data as follows:

$$\text{Minimize } Z(x) = c^T x$$

$$\text{s.t. } Ax = b$$

$$x \geq 0 \quad .$$

here $A=\mathbb{Z}^{m\times n}$; $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}n$.

Since all the data in the above linear programming problem are integers so its size (or input length) L can be given by

$$L = 1 + \lceil(1 + \log_2(1+m))\rceil + \lceil(1+ \log_2(1+n))\rceil$$
$$+ \sum_{i=1}^{m}\sum_{j=1}^{n} \lceil(1 +\log_2(1 +|a_{ij}|))\rceil + \sum_{i=1}^{m} \lceil(1 +\log_2(1 +|b_i|))\rceil$$

$$+ \sum_{j=1}^{n} \lceil(1 +\log_2(1 +|c_j|))\rceil.$$

2.2.5 Convergence of an algorithm

A desirable property of an algorithm for solving a mathematical programming problem is that it generates a sequence of points converging to a global optimal solution. In many cases, we may have to be satisfied with less favourable outcomes. In fact, as a result of non-convexity, problem size and other difficulties we may stop the iterative procedure if a point belonging to a prescribed set, usually called the solution set, is reached. Thus, in general, convergence of an algorithm is made in reference to the solution set rather than to the collection of global optimal solutions.

2.2.6 Rate of convergence of an algorithm

Theoretical convergence of algorithms to the points in the solution set is a highly desirable property. To a certain extent the significance of the rate of convergence of an algorithm is as important as the fact that it should converge. If an algorithm does converge but very slowly we may

never be able to see its convergence. Thus the rate of convergence is an important factor to measure the efficiency of an algorithm.

2.2.7 Linear Convergence

Let $<x^{(k)}>$ be a sequence of points generated by an algorithm and $<x^{(k)}>$ converge to a point $x^{(*)}$. If there is an $\alpha \in (0,1)$ and $k_o \geq 0$ such that $\| x^{(k+1)} - x^{(*)} \| \leq \alpha \| x^{(k)} - x^{(*)} \|, k \geq k_o$, $k \geq k_o$ then this algorithm is called linearly convergent.

2.2.8 Superlinear convergence

Let $<x^{(k)}>$ be a sequence of points generated by an algorithm and $<x^{(k)}>$ converge to a point $x^{(*)}$. If there is a sequence $<\alpha_k>$ which converges to zero such that

$$\| x^{(k+1)} - x^{(*)} \| \leq \alpha_k \| x^{(k)} - x^{(*)} \|, \; k = k_o, \ldots$$

Then this algorithm is called superlinearity convergent. Dennis and More [19a] have noted that one of the properties of superlinearly convergent algorithm is that

$$\lim_{k \to \infty} \frac{\| x^{k+1} - x^{(k)} \|}{\| x^{(k)} - x^{(*)} \|} = 1$$

provided $x^{(k)} \neq x^{(*)}$ for $k \geq 0$

2.3 THE ELLIPSOID ALGORITHM

In this section we show how the ellipsoid algorithm can be used to determine the feasibility of a system of linear inequalities with integer data in polynomial time.

We consider a system of linear inequalities with integer

(or rational) data as follows:

$$A^T x \leq b$$

$$\text{where } A^T \in \mathbb{Z}^{mxn} \qquad (2.1)$$

$$b \in \mathbb{Z}^m$$

If the columns of A are denoted by $a_1, a_2, \ldots, a_m$ and the components of b are denoted by $\beta_1, \beta_2, \ldots, \beta_m$, then (2.1) is equivalent to

$$a_i^T x \leq \beta_i, \quad i=1,2,\ldots,m \qquad (2.2)$$

we assume throughout $n \geq 2$.

The iteration

The Ellipsoid algorithm constructs a sequence of ellipsoids $E_o, E_1, E_2, \ldots, E_k, \ldots$ each of which contains a point satisfying (2.1), if one exists. On the (k+1)st iteration, the method checks whether the center x_k of the ellipsoid E_k satisfies the inequalities (2.2). If so, the method stops. If not, some constraints violated by x_k, say

$$a^T x_k > \beta \qquad (2.3)$$

is chosen and the ellipsoid of minimum volume containing the half-ellipsoid

$$\{x \in E_k \mid a^T x \leq a^T x_k\} \qquad (2.4)$$

is constructed using the following formulae:

$$x_{k+1} = x_k - \tau(B_k a / \sqrt{a^T B_k a}) \qquad (2.5)$$

$$B_{k+1} = \delta(B_k - \sigma(B_k a (B_k a)^T / (a^T B_k a))) \qquad (2.6)$$

$$\text{where } \tau = 1/(n+1)$$

$$\sigma = 2/(n+1) \qquad (2.7)$$

$$\delta = n^2/(n^2-1)$$

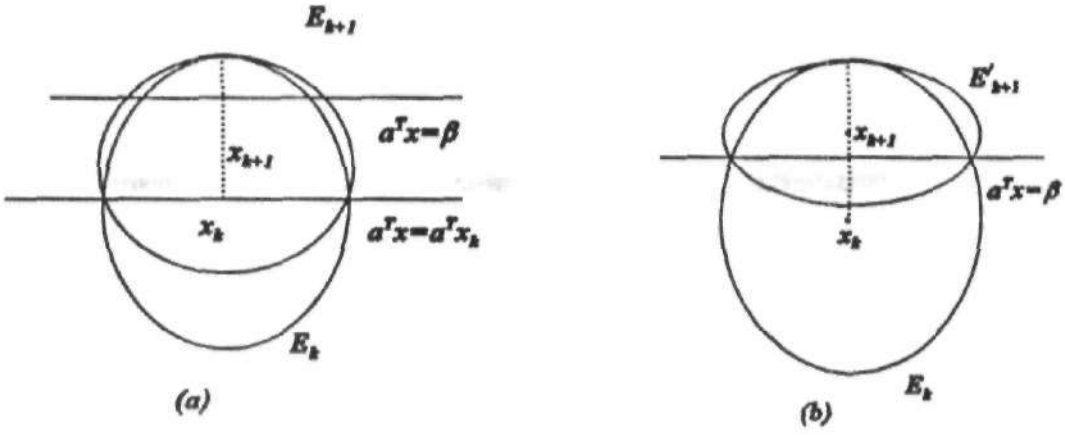


Figure 2.1: The Ellipsoid Method

(a) Without deep cuts (b) With deep cuts

The (k+2)th ellipsoid E_{k+1} determined by the point x_{k+1} and the positive symmetric definite matrix B_{k+1}, using formulae (2.5) and (2.6), is the ellipsoid of minimum volume containing the half ellipsoid (2.4). τ,σ,δ as defined in equations (2.7) are called the step, dilation and expansion parameters respectively. If B_k is a multiple of the identity matrix of appropriate order so that E_k is a ball, then E_{k+1} is shrunk in the direction a by a factor. $\sqrt{\delta(1-\sigma)}=n/(n+1)$ and expanded in all orthogonal directions by the factor $\sqrt{\delta}= n/(\sqrt{n^2-1})$.

A smaller Ellipsoid E'_{k+1} can be employed since it is only necessary that the new ellipsoid contains the section of E_k, $\{x \epsilon E_k | a^T x \leq \beta\}$ rather than the entire half ellipsoid (2.4).

2.4 MODIFICATIONS OF THE ELLIPSOID ALGORITHM

In this section we discuss some simple modifications to the basic algorithm to improve its rate of convergence. The most obvious way to do this is to use *deep* or possibly *deepest* cuts at each iteration to generate smaller ellipsoids. Iudin and Nemirovskii's [64] description of the ellipsoid algorithm allows cuts that do not pass through the center of the ellipsoid. Although they were interested in "shallow" cuts, their formulae apply to deep cuts as well. Shor and Gershovich [106] were the first to propose the use of deep cuts to speed up the convergence of the ellipsoid method.

2.4.1 Deep Cuts

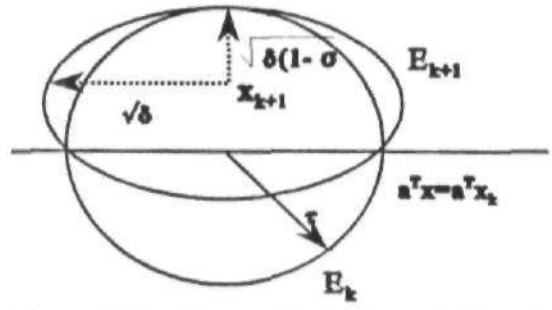


Figure: 2.2: Geometrical interpretation of the parameters.

Let us suppose that (2.3) is the linear inequality violated by the current center x_k, of the ellipsoid E_k at (k+1)th iteration. The ellipsoid E_{k+1} determined by formulae (2.5) and (2.7) contains the half-ellipsoid $\{x \in E_k | a^T x \leq a^T x_k\}$. As we only require that E_{k+1} contains the smaller portion of E_k,$\{x \in E_k | a^T x \leq \beta\}$ of E_k, it seems that we can obtain an ellipsoid of smaller volume by using the "deep cut" $a^T x \leq \beta$ instead of the cut $a^T x \leq a^T x_k$, which passes through the center of E_k. Such smallest ellipsoid is obtained by x_{k+1} and B_{k+1} using (2.5) and (2.6) with the choice of

parameters τ,σ,δ given as:

$$\tau = (1+n\alpha)/(n+1)$$

$$\sigma = (2(1+n\alpha))/((n+1)(1+\alpha))$$

$$\delta = (n^2/(n^2-1))(1-\alpha^2)$$

and $$\alpha = (a^T x_k - \beta)/(\sqrt{a^T B_k a}) \qquad (2.8)$$

The quantity α which now appears in the updating formulas represents the distance of x_k from the half space

$$H = \{x \in \mathbb{R} \mid a^T x \leq \beta\}.$$

By computing α for each inequality in (2.1) we can select the deepest possible cut to ensure better rate of convergence. As the value of α in (2.8) exceeds one, it indicates infeasibility of the system (2.1).

2.4.2 Surrogate Cuts

By combining the inequalities in (2.1), we can sometimes obtain cuts that are deeper than any cut generated by a single constraint in (2.1). Any cut of the form $u^T A^T x \leq u^T b$ (i.e., $a^T x \leq b$, with $a=Au$ and $\beta = u^T b$) is valid as long as $u \geq 0$, for then no points that satisfy (2.1) are cut off by this inequality. In Goldfarb and Todd [37] the term "surrogate" cut was introduced. Let

$$\bar{A}^T x \leq \bar{b}$$

be any subset of the set of constraints (2.1) where the columns $\bar{A}$ are linearly independent and at least one of the constraints in this subset is violated by x_k. If

$$\bar{u} = (\bar{A}^T B_k \bar{A})^{-1}(\bar{A} x_k - \bar{b}) \qquad (2.9)$$

is non negative, then the surrogate cut $\bar{u}^T \bar{A}^T x \leq \bar{u}^T \bar{b}$ is the deepest cut

with respect to that subset. It is shown in Goldfarb and Todd [37] and Todd [118], that if $\bar{A}^T B_k \bar{A}$ has non-negative off-diagonal entries, then $\bar{u}$ given by (2.9) is non-negative.

Solving a quadratic programming problem or computing $\bar{u}$ using-equation (2.9) for a large subset of constraints may be too difficult to obtain the deepest or nearly deepest surrogate cut. In Goldfarb and Todd [37], it is recommended that only surrogate cuts which can be generated from two constraints be considered.

2.5 POLYNOMIAL SOLVABILITY

Polynomial Algorithm

An algorithm A is called a polynomial-time algorithm for a problem P of dimension m x n and input length L, if there exists a polynomial upper bound $\phi(m,n,L)$ on the number of arithmetic operations, when A is applied to find a solution to the problem P.

Strongly Polynomial Algorithm

An algorithm A_s is called a strongly polynomial algorithm for a problem P if there exists a polynomial upper bound $\psi(m,n)$ on the number of arithmetic operations when A_s is applied to find a solution to the problem P.

Existence of strongly polynomial algorithm for linear programming problem is still a difficult open problem. Further details on polynomiality of the algorithms can be found in Bland *et al.* [13] and Singh and Singh [112].

The most important aspect of Ellipsoid Method is that the feasibility or infeasibility of the system of linear inequalities (2.1) with integer (or rational) data can be determined in polynomial time. Proofs of polynomiality of the ellipsoid algorithm proposed by Khachian can be found in Gaćs and Lovász [28], Padberg and Rao [92,93].

2.6 APPLICATION OF ELLIPSOID METHOD TO LINEAR PROGRAMMING PROBLEM

So far, in the preceding sections, we have addressed the problem of detecting feasibility of the system of linear inequalities. In this section we shall address the problem of application of the ellipsoid method to solve linear programing problems. The ellipsoid method can be applied in various ways to solve a linear programming problem.

2.6.1 Simultaneous Solution of the Primal and Dual in $\mathbb{R}^{m+n}$

We consider a linear programming problem with integer data as

$$\text{Maximize } c^T x$$

$$\text{subject to } A^T x \leq b$$

$$x \geq 0 \qquad (2.10)$$

where A^T is an mxn matrix.

Then the dual of (2.13) is

$$\text{Minimize } b^T y$$

$$\text{Subject to } Ay \geq c$$

$$y \geq 0 \qquad (2.11)$$

In view of duality theorem (1.10), combining the above two problems give

rise to the following system of linear inequalities:

$$\begin{aligned} A^T x &\le b \\ -x &\le 0 \\ -Ay &\le -c \\ -y &\le 0 \\ -c^T x + b^T y &\le 0 \end{aligned} \qquad (2.12)$$

In view of the fundamental theorem on duality 1.10, if (x^*,y^*) is a feasible solution to the system of linear inequalities (2.12), then x^* is the optimal solution to the primal problem (2.10) and y^* is the optimal solution to the dual problem (2.11).

In view of this equivalence between linear programming problem (2.10) and system of linear inequalities (2.12), all modifications applicable to the system (2.12) of linear inequalities will be automatically applicable to linear programming problems (2.10). Since (2.12) is a system of linear inequalities, hence ellipsoid algorithm can be used to solve (2.12) and consequently we can get the solutions of primal and dual problems. Since the system of linear inequalities is polynomially solvable, hence by implication linear programming problem is also polynomially solvable. Moreover, this method directly produces an optimal dual solution.

From a practical viewpoint, there are several disadvantages to this approach. The ellipsoid algorithm is applied to a system of linear inequalities is $\mathbb{R}^{m+n}$. This high dimensionality slows the convergence. In many practical problems the feasible region for the linear programming problem (2.10) is bounded and explicit bounds are known. So a method working only in primal space can be initialized with an ellipsoid of large but not astronomical volume, speeding convergence. Next disadvantage

of this approach is that, all solutions to (2.12) lie in the hyperplane $c^Tx=b^Ty$, hence even if (2.12) is feasible, the feasible set has zero volume. Thus perturbation of the right hand sides in (2.12) is necessary. Even if (2.12) is feasible, the volume of the feasible set of the perturbed problem will be very small making the number of iterations to be very large. Another disadvantage is that if the algorithm determines that (2.12) is infeasible, yet no feasible primal or dual solution is produced, it is not clear whether (2.10) is infeasible or unbounded.

To overcome these difficulties the strategy used is to choose constraints of (2.12) to generate cuts for the ellipsoid algorithm. The high dimensionality is not too drastic a problem until feasibility is reached. It seems reasonable to base cuts only on the primal constraints until a primal feasible x is generated, then only the dual feasible y is generated. If no primal feasible solution is generated in 6n(m+n+1)L steps, (2.10) is infeasible. If a primal feasible solution is generated in k steps but then no dual feasible solution is generated after a further

min {k+6m(m+n+1)L, 6(m+n)(m+n+1)L-k} steps, (2.10) is bounded.

Jones and Marwil [56] presents a variant of this approach of simultaneously solving the primal and dual using the complementary slackness conditions (as discussed in the theorem 1.11) for (2.10) and (2.11) to reduce dimensionality of the problem as iterations are performed.

2.6.2 Bisection Method

In 1979, this approach was suggested by Kozolov, Tarasov and Khachiyan [68] for convex quadratic programming. In 1980, Padberg and Rao [92] presented a polynomial time algorithm combining bisection with

ellipsoid method for linear programming.

The bisection method applies the ellipsoid algorithm to the constraints of (2.10) to obtain a feasible solution x' if one exists; if there is none, we terminate. Then $Z'=c^Tx'$ is a lower bound on the optimal value of (2.10). Next we obtain an upper bound on this value. If the feasible region of (2.10) is bounded and contained in the current ellipsoid

$E_k = \{x \in \mathbb{R}^n | (x-x_k)^T B_k^{-1}(x-x_k) \leq 1\}$,

then $Z= c^Tx_k+(c^TB_kc)^{1/2}$ is such an upper bound.

The main disadvantage of bisection method is that the system of inequalities $A^Tx \leq b$, $x \leq 0$, $-c^Tx \leq -Z$ (2.13)

with Z too large will be infeasible and may take a large number of iterations. Deep cuts should be used to allow early termination in such cases.

2.6.3 Sliding Objective Function Method

The sliding objective function method was first proposed by Iudin and Nemirovskii [55] and latter on by Shor [105].

In this method we start, as in the bisection method, by generating a feasible solution x' to (2.10). Next, we consider inequalities (2.13) with $Z=Z'=c^Tx'$ where $Z'=c^Tx'$. We may proceed with the ellipsoid algorithm using the cut $c^Tx \geq c^Tx'=Z'$, since the next ellipsoid can be defined even when the current iterate lies on the chosen cut. In this method, we always consider feasible systems (except possibly the first). Whenever a feasible iterate x_k satisfies $c^Tx > c^Tx'=Z'$, we set $x' \leftarrow x_k$ and $Z \leftarrow Z' \leftarrow c^Tx_k$ and proceed as above.

This method is probably the most efficient for practical implementation. It always considers feasible systems and never

backtracks. All computation takes places in $\mathbb{R}^n$. If the feasible region of (2.10) is bounded and known to lie in E_o, then upper bounds Z_k can be found as

$Z_k=\min\{Z_{k-1}, c^Tx_k+(c^TB_kc)^{1/2}\}$.

2.7 RELATIONS TO THE SIMPLEX METHOD

Shor [104], Iudin and Nemirovskii [55] and more recently by Krol and Mirman [69], and Jones and Marwill [56] consider the space $\mathbb{R}^n$ to be transformed at each iteration so that the current ellipsoid is a sphere. The ellipsoid $E_k \subset \mathbb{R}^n$ is transformed by $z=T(x)=J_k^{-1}(x-x_k)$ where J_k is a non singular matrix which transforms the unit ball into E_k, x_k shifted to the origin and $B_k = J_k J_k^T$ ...(2.17)

As the ellipsoid shrinks in volume on each step, the feasible set $P_k = \{z:(A^k)^Tz \leq b^k\}$ for the system of linear inequalities (2.1) in "dilated" space of the z variables expands, where $A^k=J_k^TA$ and $b^k=b-A^Tx_k$. If the feasible set has a non-empty intersection with the initial ellipsoid, a unit ball, then, since this set expands and is contained in the unit ball after each transformation, after a polynomially bounded number of steps, $0 \in P_k$, indicating feasibility.

We can view simplex pivoting in a similar way. If it is used to find a feasible point satisfying the constraints $Ax=b$, $x \geq 0$, given in standard form, then each simplex pivot can be thought of as an affine transformation from one space of non basic variables to another obtained by exchanging one non basic and basic variable.

mathematically, inherent an essential artificiality which hardly can have correlates in real life. Thus, it may be generally accepted that unless some further breakthrough in the implementation of this algorithm take place, it cannot plausibly be said that this algorithm can be a successful alternative to the simplex algorithm to solve real life linear programming problems. However, as discussed in foregoing sections, despite showing some insurmountable difficulties in its practical applicability, the overall impact of the ellipsoid method, particularly on theoretical developments and combinatorial optimization cannot be denied.

But still, the idea of ellipsoid method will expand our perspective of linear programming and related combinatorial problems. The excitement caused by the ellipsoid method will generate further research in this area. Another polynomial algorithm for linear programming has been proposed by N. Karmarkar [58] whose complexity bound is lower than that of Khachian and is computationally very efficient to solve large-scale linear programming problems. We propose to discuss this algorithm and related development in the next chapter of this thesis.

CHAPTER 3

KARMARKAR'S ALGORITHM

3.1 INTRODUCTION

In 1984, Karmarkar [58] proposed a polynomial algorithm of the complexity $O(n^{3.5}L^2)$ to solve linear programming problems where n is the number of variables and L is the number of bits of the input data. This is better than Khachian's complexity of $O(n^4L^2)$. The running-time of this algorithm is less than the Ellipsoid method by a factor of $O(n^{0.5})$. Given a polytope P and a strictly interior point $a \in P$, projective transformation is used to map (P, a) to (P',a') and to show that the ratio of the radius of the smallest sphere with centre a' containing P' to the radius of the largest sphere with centre a' contained in P' is O(n). The algorithm consists of repeated application of such projective transformations each followed by optimization over an inscribed sphere to produce a sequence of points which converges to the optimal solution in polynomial-time.

3.2 GENERAL PROJECTIVE TRANSFORMATIONS

General projective transformations of $\mathbb{R}^n$ are described by formulas of the form

$$x \to \frac{Cx + d}{f^T x + g}$$

where $C \in \mathbb{R}^n$, d, $f \in \mathbb{R}^n$, $g \in \mathbb{R}$ and $\begin{bmatrix} C & d \\ f^T & g \end{bmatrix}$ is a non-singular square matrix of order (n+1).

3.2.1 Projective Transformation $T(a,a_o)$

We are interested in projective transformations of the hyperplane $\sum = \left\{ x \in \mathbb{R} | \sum_{i=1}^{n} x_i = 1 \right\}$.

For each interior point a of the simplex $\Delta = \left\{ x \in \mathbb{R}^n | x \geq 0, \sum_{n=1}^{n} x_i = 1 \right\}$ there is a unique projective transformation $T(a,a_o)$ of the hyperplane $\sum$ fixing all vertices of the simplex Δ and moving $a=(a_1,a_2,...,a_n)^T$ to the center $a_o=\left(\frac{1}{n}\right)e$.

Let D = diag $\{a_1,a_2,...,a_n\}$ be the diagonal matrix with diagonal entries $a_1,a_2,...,a_n$. Then $T(a,a_o)$ is given by

$T(x) = x' = \frac{D^{-1}x}{e^T D^{-1}x}$, where $e=(1,1,...,1)^T$ or equivalently by

$$x_i' = \frac{x_i/a_i}{\sum_{j=1}^{n} x_j/a_j}, i=1,2,...,n.$$

This transformation $T(a,a_o)$ has the following properties:

1. T is one-one and maps the simplex onto itself. Its inverse is given by $x = \frac{Dx'}{e^T Dx}$

   or, $x_i = \frac{a_i x_i'}{\sum_{j=1}^{n} a_j x_j'}$, i=1,2,...,n.

2. Each facet of the simplex given by $x_i=0$ is mapped onto the corresponding facet $x_i'=0$.

3. Image of the point x=a is the center $\frac{1}{n}(e)$ of the simplex.

4. Let A_i denote the ith column of A. Then $\sum_i A_i x_i = 0$ becomes

$$\frac{\sum_i A_i a_i x_i'}{\sum_i a_i x_i'} = 0$$

or $\sum_i A_i' x_i' = 0$ where A_i'=$a_i A_i$.

We define the affine subspace $\Omega'=\{x' | A'x'=0\}$.

5. If a ϵ Ω then the centre of the simplex $a_o \epsilon \Omega'$.

Let $B(a_o,r)$ be the largest ball with centre a_o and radius r contained in the simplex. And let $B(a_o,R)$ be the smallest ball containing Δ. Then

$$\frac{R}{r} = n-1$$

Now, $B(a_o,r) \subseteq \Delta \subseteq B(a_o,R)$ i.e., $B(a_o,r) \cap \Omega' \subseteq \Delta \cap \Omega' \subseteq B(a_o,R) \cap \Omega'$. But $\Delta \cap \Omega'$ is the image π' of the polytope $\Pi = \left\{ x \in \mathbb{R}^n \mid Ax = 0, \sum_{i=1}^{n} = 1 \right\}$. The intersection of a ball and an affine subspace is a ball of lower dimension in that subspace and has the same radius as the centre of the original ball which lies in the affine subspace. Hence,

$$B'(a_o,r) \subseteq \pi' \subseteq B'(a_o,R), \frac{R}{r} = n-1.$$

3.3 CONVERSION OF A LINEAR PROGRAMMING PROBLEM TO CANONICAL FORM:

We consider the linear programming problem

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } Ax \geq b \qquad (3.1) \\ &x \geq 0 \end{aligned}$$

where $c, x \epsilon \mathbb{R}^n$, $b \epsilon \mathbb{R}^m$, $A \epsilon \mathbb{R}^{mxn}$.

The following is one of the methods to convert the above problem

to canonical form:

minimize $c^T x$

subject to $Ax = 0$

$$\sum_{i=1}^{n+1} x_i = 1 \qquad (3.2)$$

$x \geq 0$

Here $x = (x_1,\ldots,x_{n+1})^T \in \mathbb{R}^{n+1}$, $c \in \mathbb{Z}^{n+1}$, $A \in \mathbb{Z}^{mx(n+1)}$.

STEP 1: To combine Primal and Dual problems.

$Ax \geq b$

$A^T u \leq c$

$c^T x - b^T u = 0$

$x \geq 0, u \geq 0$

In view of theorem 1.10 (Fundamental Theorem on duality) this combined problem is feasible if and only if the original problem has finite optimum solution.

STEP 2: To introduce slack/surplus variables

$Ax - y = b$

$A^T u - v = c$

$c^T x - b^T u = 0$

$x \geq 0, u \geq 0, y \geq 0, v \geq 0.$

STEP 3: To introduce an artificial variable to create an interior starting point

Let x_o, y_o, u_o, v_o be strictly interior points in the positive orthant.

minimize λ

$Ax - y + \lambda(b - Ax_o + y_o) = b$

$$A^T u+v+\lambda(c-A^T u_o-v_o)=c$$

$$c^T x-b^T u+\lambda(-c^T x_o+b^T v_o)=0$$

$$x\geq 0,\ y\geq 0,\ u\geq 0,\ v\geq 0,\ \lambda\geq 0$$

Here $x=x_o, y=y_o, u=u_o, v=v_o$, $\lambda=1$ is a strictly interior feasible solution which can be taken as a starting point. The minimum value of λ is zero if and only if the problem in step 2 is feasible.

Changing the notations, the above problem is re-written as

minimize $c^T x$,

subject to $Ax=b$

$$x\geq 0, \qquad (3.3)$$

where $c,x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}$. We can note that $x=a$ is the strictly interior starting point and the target value of the objective function is zero.

STEP 4: **To use a Projective Transformation of the positive orthant into a simplex.**

Let $x'=T(x)$ where $x \in \mathbb{R}^n$, $x' \in \mathbb{R}^{n+1}$ and $x_n' = \dfrac{x_i/a_i}{\sum_j x_j/a_j + 1}$, i=1,2,...,n.

$$x_{n+1}' = 1 - \sum_{i=1}^{n} x_i'.$$

Let $P_+=\{x\in\mathbb{R}^n \mid x\geq 0\}$ denote the positive orthant and $\Delta=\{x\in\mathbb{R}^{n+1} \mid x\geq 0, \sum_{i=1}^{n+1} x_i=1\}$, denote the simplex. The transformation T(x) has the following properties:

1. It maps P_+ onto Δ
2. It is one-one onto and the inverse transformation is given by $x_i = \dfrac{a_i x_i'}{x_{n+1}'}$, i=1,2,...,n.
3. The image of the point a is the centre of the simplex.

4. Let A_i denote ith column of A. If $\sum_{i=1}^{n} A_i x_i = b$

then $\sum_{i=1}^{n} \frac{A_i a_i x_i'}{x_{n+1}'} = b$

or, $\sum_{i=1}^{n} A_i a_i x_i' - b x_{n+1}' = 0$

We define the columns of a matrix A' by

$$A_i' = a_i A_i, \quad i=1,2,\ldots,n.$$

$$A_{n+1} = -b$$

then $\sum_{i=1}^{n+1} A_i' x_i' = 0$

Let Ω' denote this affine space, $\Omega' = \{x' \in \mathbb{R}^{n+1} | A'x' = 0\}$.

STEP 5:

We define Z to be the affine subspace corresponding to the zer

set of the objective function in the original space.

i.e., $Z = \{x \in \mathbb{R}^n | c^T x = 0\}$

substituting, $x_i = \frac{a_i x_i'}{x_{n+1}'}$

We get $\sum_{i=1}^{n+1} \frac{c_i a_i x_i'}{x_{n+1}'} = 0$

Let

$c_i' = a_i c_i$, $\quad i=1,2,\ldots,n$

$c_{n+1}' = 0$, then $c' \in \mathbb{R}^{n+1}$, and

$c^T x = 0$ implies $c'^T x' = 0$.

So, the image of Z is given by

$Z' = \{x' \in \mathbb{R}^{n+1} | c'^T x' = 0.\}$

The problem (3.3) ultimately reduces to

minimize $c'^{T}x'$

subject to $A'x'=0$

$$x \geq 0, \sum_{i=1}^{n+1} x_i' = 1.$$

The centre 1/n(e) of the simplex is a feasible starting point for the problem in canonical form obtained above.

If there is a solution x' such that $c'^{T}x'=0$. Then the inverse image of x' gives an optimal solution to the original problem. Otherwise, we conclude that minimum value of $c'^{T}x'$ is strictly positive and this corresponds to the case that the original problem does not have a finite optimum, i.e., it is either infeasible or unbounded.

3.3.1 OPTIMIZATION OVER A POLYTOPE

The Linear Programming Problem (3.2) in Canonical form can be stated as follows:

minimize $c^{T}x$

subject to $Ax=0$

$$\sum_{i=1}^{n} x_i = 1$$

$$x \geq 0$$

where $x \in \mathbb{R}^n$, $c \in \mathbb{Z}^n$, $A \in \mathbb{Z}^{mxn}$.

Let Ω be the subspace $\{x|Ax=0\}$,

Δ be the simplex $\{x|x \geq 0, \sum_{i=1}^{n} x_i = 1\}$

and $\pi = \Omega \cap \Delta$.

Then the above problem reduces to

minimize $c^{T}x$

subject to $x \in \pi$.

3.3.2. Optimization over a sphere

If we replace the constraint $x \epsilon \Delta$ by a constraint $x \epsilon S$, where S is a sphere then the optimum can be found by solving a system of linear equations. The intersection of a sphere and an affine space is a lower dimensional sphere inside that affine space. Hence the problem becomes

minimize $c'^{T}x$

subject to $x \in S$ (a sphere).

where c' is the orthogonal projection of c onto Ω. This can be acieved by taking a step along the direction of -c' from the centre of the sphere, of length equal to the radius of the sphere.

3.3.3 Bounds on the Objective function.

Let a_o be a strictly interior point in the polytope P. Let us draw an ellipsoid E with centre a_o contained in the polytope and another ellipsoid E' by magnifying E by a sufficiently large factor v so that E' contains P.

$E \subset P \subset E'$; $E' = vE$.

Let f_E, f_P,$f_{E'}$ be the minimum values of the objective function f(x) on E, P and E' respectively. Then

$$f_{E'} \leq f_P \leq f_E$$

i.e., $f(a_o) - f_E \leq f(a_o) - f_P \leq f(a_o) - f_{E'} = v[f(a_o)-f_E]$

i.e., $\frac{f(a_o)-f_E}{f(a_o)-f_P} \geq \frac{1}{v}$

i.e., $\frac{f_E-f_P}{f(a_o)-f_P} \leq 1-\frac{1}{v}$

Thus by going from a_o to the point, say a', that minimizes f(x) over E' we came closer to the minimum value of the objective function by a factor

$[1 - \frac{1}{v}]$. The same process can be repeated with a' as the centre. The rate of convergence depends on v, the smaller the value of v, the faster the convergence.

3.3.4 Invariant Potential Functions

The set of linear functions is not invariant under a projective transformations but the ratio of two linear functions can be transformed into the function of the same type. Keeping this fact in mind, Karmarkar [69] has introduced a potential function as a ratio of two linear functions as follows:

$$f(x) = \sum_j \ln \frac{c^T x}{x_j}$$

This potential function is invariant under projective transformation and plays an important role in measuring the progress of the algorithm.

3.4 THE ALGORITHM

The algorithm generates a sequence of points $x^{(0)}, x^{(1)}, \ldots, x^{(k)}$ starting with $x^{(0)} = a_o = \left(\frac{1}{n}\right)(e)$ which is the centre of the simplex.

$$\Delta = \{x \geq 0, e^T x = 1\}$$

The steps are as follows:

STEP 0: Initialize $x^{(0)} = \frac{1}{n}(e) = a_o$

STEP 1: At any kth iteration, the point $x^{(k)}$ is mapped to the centre $a_o = \left(\frac{1}{n}\right)(e)$ of the simplex Δ and a point b is obtained in the following substeps whose inverse image gives the next point $x^{(k+1)}$ of the sequence.

Projective Transformation T_k:

Let $D = \text{diag}\left\{x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right\}$, where $x_i^{(k)}$ are the

components of the vector $x^{(k)}$ and $e^T=(1,1,...,1)$ is a vector of ones.

Then T_k is given by

$$x' = T_k(x) = \frac{D^{-1}x}{e^T D^{-1}x}$$

and $T^{-1}(x)$ is given by

$$x = T_k^{-1}(x') = \frac{Dx'}{e^T Dx'}$$

Substep 1.1

Let $B = \begin{bmatrix} AD \\ e^T \end{bmatrix}$

Substep 1.2

Dc is projected orthogonally on the null space of B,

i.e., $c_p = [I - B^T(BB^T)^{-1}B]Dc$

Substep 1.3

c_p is normalized.

$\hat{c} = \frac{c_p}{\|c_p\|}$, i.e., $\hat{c}$ is the unit vector in the direction of c_p.

Substep 1.4

$b = a - \alpha r\hat{c}$, i.e., a steplength αr is taken in the direction of $\hat{c}$, where $\alpha \in (0,1)$ is a parameter which Karmarkar takes as $\alpha = ¼$ and $r = \frac{1}{\sqrt{n(n-1)}}$ is the radius of the largest sphere inscribed in the simplex Δ.

Substep 1.5

Inverse transformation $T^{-1}(x)$ is applied to b so that

$$x^{(k+1)} = \frac{Db}{e^T Db}.$$

The iterations stop where the current iterate $x^{(k)}$ meets the required

convergence check

$$c^T x^{(k)} \leq 2^{-q} c^T x^{(0)}.$$

During the execution of the algorithm we expect certain decrease δ in the potential function value at each iteration.

If no such decrement δ in the potential function value occurs i.e., if $f(x^{(k+1)}) > f(x^{(k)}) - \delta$, then we stop and conclude that min $c^T x > 0$. This indicates that the original linear programming problem (from which canonical form is obtained) is either infeasible or unbounded.

3.5 THEOREMS

Theorem 1:

In 0(n(q+log n)) steps the algorithm finds a feasible point x such that

$\frac{c^T x}{c^T a_0} \leq 2^{-q}$, where q is some positive integer.

The objective function $c^T x$ is associated with a "potential function" f(x) given by

$$f(x) = \sum_j \ln\left[\frac{c^T x}{x_j}\right].$$

Theorem 2:

Either (i) $f(x^{(k+1)}) \leq f(x^{(k)}) - \delta$

or (ii) the minimum value of objective function is strictly positive, where δ is a constant and depends on the choice of the value of the parameter α of sub step 1.4.

We restrict the affine space $\Omega' = \{y|ADy=0\}$ by the equation $\sum y_j=1$. Let Ω'' be the resulting affine space and B be the matrix obtained by

adding a row of all 1's to AD. Then any displacement in Ω" is in the null space of B.

Theorem 3:

There exists a point b'∈ $B(a_o, \alpha r) \cap \Omega''$ such that $f'(b') \leq f'(a_o) - \delta$, where δ is a constant.

Theorem 4:

Let b' be the point that minimizes $c'^T x$ over $B(ao, \propto r) \cap \Omega''$. Then $f'(b') \leq f'(a_o) - \delta$ where δ is a positive constant depending on α. For α = ¼ we may take $\delta = \frac{1}{8}$.

Theorem 5:

The point b' returned by algorithm (step 1.2, 1.3, 1.4) minimizes $c'^T x$ over $B(a_o, \propto r) \cap \Omega''$.

Proofs of the Theorems can be found in Karmakar [69] pp. 379-382.

3.6 EFFICIENCY OF THE ALGORITHM.

3.6.1 Convergence in Karmarkar's Algorithm

Since in the Karmarkar's algorithm the targeted minimum value of the objective function is zero, so if $c^T x^{(0)}, c^T x^{(1)}, \ldots$ are the objective function values corresponding to the iterates $x^{(0)}, x^{(1)}, \ldots$, generated by Karmarkar's algorithm, then we expect the convergence of the sequence $\langle c^T x^{(k)} \rangle$ to zero. Now, in view of the relation between objective function and the potential function value, a faster decrease in the potential function value will lead to an early convergence of the algorithm.

3.6.2 Rate of Convergence of Karmarkar's Algorithm

Linear rate of convergen for Karmarkar's algorithm has been

established by Charnes *et al.* [16] and Singh and Singh [107] independently. Singh and Singh [107] also established superlinear rate of convergence for n=2 which they themselves admit as a trivial case.

Barne's [9] has proposed a variation on Karmarkar's algorithm for solving linear programming problems in standard form which produces a monotonically decreasing sequence of objective function values. In the absence of degeneracy, the algorithm converges to an optimal basic feasible solution with non-basic variables converging monotonically to zero. For this variant of Karmarkar's algorithm, Barnes [9] has obtained an inequality

$$\frac{c^T x^{(k+1)} - c^T x^{(*)}}{c^T x^{(k)} - c^T x^{(*)}} \leq 1 - \frac{R}{(n-m+W_k)^{1/2}}$$

where $x^{(k)}$ and $x^{(*)}$ denote the kth and optimal solutions, respectively, R is a real number in the open interval (0,1) and

$$W_k = \sum \left(\frac{x_i^{(k)} - x_i^{(k+1)}}{x_i^{(k)}} \right)^2 \to 0 \text{ as } k \to \infty.$$

3.7 COMPLEXITY OF THE ALGORITHM

The value of the objective function is reduced by a constant factor in 0(n) steps. As in ellipsoid algorithm we define

$$L = \log (1 + |D_{max}|) + \log(1 + \alpha)$$

where

D_{max} = max {|det (M)| :M is square submatrix of constraint matrix A}

and $\propto$ = max {$|c_i'|$, $|b_i|$, i=1,2,...,n}

Here L is not greater than the length of the input.

During the course of algorithm, we periodically check for optimality by converting the interior point solution to an extreme point solution without degrading the objective function value, and testing the extreme point for optimality. Any two distinct values of the objective function on the vertices of the polytope have to differ by at least 2^{-kL} for some constant k Grotschel et.al [49]. So, in at most 0(nL) steps we reach a situation in which any extreme point better than the current interior point has to be an exact optimal solution.

In each step, we have to solve a linear system of equations which takes $0(n^3)$ arithmetic operations in the worst case. If we solve the equations at each step by modifying the solution to the previous step, we can save a factor $\sqrt{n}$.

For each arithmetic operation, we need a precision of 0(L) bits. This much precision is required for any method to solve Ax=b if det (A) = $0(2^L)$. This factor L must appear in the complexity of any algorithm for linear programming, since representing the output can require 0(L) bits/variable in the worst case.

The overall complexity of Karmarkar's algorithm is $0(n^{3.5}L^2 \ln L \ln\ln L)$ as compared to $0(n^4L^2 \ln L \ln\ln L)$ for the ellipsoid algorithm.

3.8 TERMINATION OF ITERATIONS

Termination of iterations in Karmarkar's algorithm depends significantly on the choice of steplength parameter α and potential function decrease δ achieved at each iteration of the algorithm. Several decrements in potential function value are possible depending on various choices of steplength parameter α. For example, Karmarkar [58] has achieved potential decrement $\delta \approx 1/8$ with the choice $\alpha=1/4$; Franklin [24] has achieved potential function decrement $\delta=1-\ln 2 \approx 0.31$ with the choice $\alpha=1/2$; Singh and Singh [110] have achieved potential function decrement $\delta \approx 0.23$ with the choice $\alpha=2/3$. Blair [12], Padberg [91] and Schrijver [99] have also obtained $\delta \approx 0.31$ decrement in the potential function value. McDiarmid [75] by taking fixed α in (0,1), has achieved potential function decrement $\delta=\ln 2 \approx 0.69$. Anstreicher [6] using optimal steplength at each iteration has achieved potential function decrease $\delta \approx 0.72$.

In Singh and Singh [111] an inequality

$$\frac{c^T x^{(k)}}{c^T x^{(0)}} < \left(\frac{1}{2}\right)^{k/n} \tag{3.4}$$

has been developed based on potential function decrease achieved in McDiarmid [75] using fixed steplength at each iteration. Based on this inequality Singh and Singh[111] has proposed an efficient termination rule as follows:

THEOREM 6

In at most $\bar{k}=nq$ iterations the algorithm finds a solution of the problem (3.2) with

$$\frac{c^T x^{(k)}}{c^T x^{(0)}} \leq 2^{-q}$$

Proof. We have

$$\frac{c^T x^{(k)}}{c^T x^{(0)}} \leq 2^{-q}, \text{ where q is a positive integer}$$

i.e., $\ln c^T x^{(k)} - \ln c^T x^{(0)} \leq -q \ln 2$ (3.5)

Using the inequality (3.4), we get

$$\ln c^T x^{(k)} - \ln c^T x^{(0)} < (k/n) \ln(1/2). \quad (3.6)$$

Now, if follows from (3.6) that (3.5) is true if k satisfies

$$(k/n)\ln (1/2) \leq -q \ln 2$$

i.e., $k \ln (1/2) \leq -nq \ln 2$

$$k \geq nq$$

As in Franklin [24], we may define

$$\bar{k} = \lceil nq \rceil = nq$$

COROLLARY 1:

The number of iterations to find an optimal solution varies as n if q remains fixed.

COROLLARY 2:

For a class of problems of fixed dimension n, the number of iterations k varies as q.

The algorithm will stop if the number of iterations k will reach $\bar{k}$. If the number of iterations k reaches $\bar{k}$ before the convergence check (3.2) is passed, we may conclude that the problem (3.1) has no solution with min $c^T x=0$ i.e., the original problem(3.3) does not have a finite optimum.

Franklin [24] has obtained $k_{max} = \lceil 2.2589 \rceil$ on the number of

iterations taken by Karmarkar's algorithm in finding a solution to (3.2). It has been claimed in Franklin [24] that this is a better result than Karmarkar's theorem 1 [58], p. 379). But Singh and Singh [111] claims that their result $\bar{k}$=nq is even better than Franklin's. They have obtained, corresponding to Karmarkar's decrement $\delta \approx 1/8$ on number of iterations, $k=\lceil 5.57058\ nq \rceil$; $k > 5\bar{k}$. Hence it is expected that this termination rule may save almost 80 percent iterations and can ensure much early termination.

3.9 COMPARISON OF THE ELLIPSOID METHOD AND KARMARKAR'S ALGORITHM

Both Karmarkar's algorithm and the ellipsoid method are polynomial algorithms to solve linear programming problems. But from a practical point of view, analytical and computational investigations of the two algorithms are quite different. The rate of convergence of the ellipsoid method even with deep cut and surrogate cut is extremely slow, specially when compared to the convergence of Karmarkar's algorithm. Recently, some researchers have tried to study the similarities and dissimilarities between these two polynomial algorithms.

Todd [118] observes that the heart of each iteration of these two algorithms is the solution of weighted least-squares subproblems and these subproblems are very closely related. This observation allows further insights into the two algorithms, specially suggesting the cause for extremely slow convergence of the ellipsoid method compared to the very fast convergence of Karmarkar's algorithm.

Both Karmarkar's and the ellipsoid algorithm appear at first sight not to provide solution to the dual linear programming problem; but a closer examination indicates that dual solutions are generated during the execution of the algorithm, essentially from the least-squares subproblems.

In fact, at kth iteration Todd [118] considers a vector $l \in \mathbb{R}^n$ such that

$$Y = \{y \in \mathbb{R}^m : l \leq A^T y \leq c\}$$

where A is an m x n matrix and $c \in \mathbb{R}^n$. As observed in Todd [118], we can easily obtain a suitable lower bound $l^{(k)}$ initially. Let $\bar{D} = D_k$ be a diagonal matrix of order nxn with non-negative entries such that $A\bar{D}A^T$ is positive definite. If

$$E = \{y \in \mathbb{R}^m : (A^T y - c)\bar{D}(A^T y - l) \leq 0\} \qquad (3.7)$$

Then $y \subseteq E$. The inequality in (3.8) can be written as

$$y^T(A\bar{D}A^T) - y^T A\bar{D}(c + l) + c^T\bar{D}l \leq 0.$$

By completing the squares the set E can be expressed as

$$E = \{y \in \mathbb{R}^m : (y - \bar{y})^T(A\bar{D}A^T)(y - \bar{y}) \leq \bar{y}^T(A\bar{D}A^T)\bar{y} - c^T\bar{D}\} \qquad (3.8)$$

where $\bar{y}$ is the solution of the system

$$(A\bar{D}A^T)y = A\bar{D}r \qquad (3.9)$$

where r=(c+l)/2. Since $A\bar{D}A^T$ is a positive definite matrix, the set E in (3.9) represents an ellipsoid with centre $\bar{y}$. Clearly (3.9) forms the normal equations for the weighted least-square problem

$$\min_y \|D^{1/2}(A^T y - r)\| \qquad (3.10)$$

where $D^{-1/2}$ is the diagonal matrix whose diagonal entries are the square root of those of matrix $\bar{D}$. Here $D^{-1/2}$ weighs the components of residual $A^T y - r$.

Karmarkar's algorithm for linear programming essentially consists of two steps: in the first step a search direction is chosen, while the second a steplength is determined. To compute search direction Cp, we project Dc orthogonally on the null space of B, i.e.,

$$Cp = (I - B^T(BB^T)^{-1})Dc.$$

Substituting $B = \begin{bmatrix} AD \\ e^T \end{bmatrix}$, we get

$$BB^T = \begin{bmatrix} AD^2A^T & ADe \\ (ADe)^T & e^Te \end{bmatrix}$$

$$= \begin{bmatrix} AD^2A^T & 0 \\ 0 & n \end{bmatrix}$$

So, in order to compute Cp, it is sufficient to find $\bar{y}$ satisfying

$$(AD^2A^T)y = AD^2c \qquad (3.11)$$

which are the normal equations for the weighted least-squares problem

$$\min_y \|D(A^Ty - c)\| \qquad (3.12)$$

After obtaining $\bar{y}$ from (3.11), we set $\bar{z} = \frac{c^T\bar{x}}{n}$ and then

$C_p = -D(c - A^T\bar{y}) - e\bar{z}$ where $\bar{x}$ is feasible solution to the LP problem in canonical form. The important similarity between the two algorithms which emerges from the above explication is that in both algorithms each iteration requires solution of normal equations (3.9) and (3.11) or weighted least-squares subproblems (3.10) and (3.12). The solution to subproblem (3.10) gives the centre $\bar{y}=y^k$ of the current ellipsoid whereas the solution $\bar{y}$ of the subproblem (3.12) gives search direction c_p.

Moreover, in the extension of Karmarkar's algorithm by Todd and Burrell [119] the solution to a related subproblem gives a vector y^{k+1} which is the part of a feasible dual solution that converges to the optimal dual solution, provided it is unique.

The weighted least-squares subproblem (3.10) in the case of ellipsoid algorithm and (3.10) in the case of Karmarkar's algorithm have different weighting matrices. Also these two subproblems differ in the right hand sides: r in the ellipsoid algorithm and c in the Karmarkar's algorithm. Thus it seems difficult to draw some meaningful conclusion by comparing these two subproblems at any particular iteration. The details of this view point in the similarity of the two algorithms can be found in Todd [118].

Todd [118] has explicated how to obtain an ellipsoid that contains all optimal solution from an extension of Karmarkar's algorithm. He has also shown that under certain conditions the volume of this ellipsoid shrinks to zero as the duality *gap* converges to zero. The volume determinant of the ellipsoid proposed in this extension includes the primal solution as parameters and convergence of the volume depends on the non-degeneracy of the optimal primal solutions. It is important to note that the volume of the ellipsoid in Todd [118] does not necessarily shrink monotonically as in the case of ellipsoid method of Khachiyan [59, 60].

Based on the explications in Todd [117], Ye [129] has proposed a new ellipsoid in Primal-Dual variant of Karmarkar's algorithm developed in Anstreicher [4], Gay [33], Todd and Burrel [119] and Ye and Kojima [131]. Ye [129] has introduced a potential function P(x,z) associated with

linear programming problem in standard form as follows:

$$P(x,z) = (n+1)\ln(cx-z) - \sum_{i=1}^{n} \ln(x_i) \qquad (3.13)$$

where x belongs to the solution set S of the linear programming problem.

minimize cx

s.t. Ax=b **(P*)**

$x \geq 0$

where c, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{mxn}$, $b \in \mathbb{R}^m$ and $z \leq$ minimum objective function value Z^*. If the dual of the problem (P) is

maximize ytb

s.t. $yA \leq c$

$y \in \mathbb{R}^m$

then from the fundamental theorem on duality (Theorem 1.10), we have

$$yb \leq Z^* \leq cx$$

if z^o be the value of the objective function corresponding to initial solution $x^{(0)}$ of the primal problem P,

$$\frac{cx-z}{cx^{(0)}-z} = \lambda(x)\exp\left(\frac{P(x,z)-P(x^{(0)},z^o)}{n+1}\right)$$

where $\lambda(x) = \exp\left(\frac{\sum_{i=1}^{n}\ln(x_i) - \sum_{i=1}^{n}\ln x_i^{(0)}}{n+1}\right)$

Assuming $\lambda(x)$ bounded from above, we can find some finite positive number μ such that

$$\frac{cx-z}{cx^{(0)}-z} \leq \mu\exp\left(\frac{P(x,z)-P(x^{(0)},z^o)}{n+1}\right) \qquad (3.14)$$

If $P(x^{(k)}, z^k)$ tends to $-\infty$ along some sequence $\{x^{(k)}: x^{(k)} > 0$ and $x^{(k)} \in S\}$ and $\{z^k \leq z^* \leq cx\}$ then $^k cx^{k} - z$ converges to zero. This indicates that minimization of potential function P(x,z) subject to constraints $0 < x \in S$ and $z \leq z^* \leq cx$ leads to optimal solution to problem P and dual problem D. The algorithm generates sequences $\{0 < x^{(k)} \in S\}$ and $\{\underline{z}^{-1} \leq \underline{z} \leq z^*\}$ which satisfy

$$P(x^{(k)}, z^k) \leq P(x^{(k-1)}, z^{k-1}) - \delta \qquad (3.15)$$

where $\delta \geq 0.2$.

Putting k=1,2,...,k in (3.15) we get k inequalities and adding these inequalities, we get

$$P(x^{(k)}, z^k) \leq P(x^{(0)}, z^0) - k\delta$$

i.e., $P(x^{(k)}, z^k) - P(x^{(0)}, z^0) \leq -k\delta \qquad (3.16)$

Inequalities (3.14) and (3.16) imply

$$\frac{cx^{(k)} - z^k}{cx^{(0)} - z^0} \leq \mu \exp\left(\frac{-k\delta}{n+1}\right) \qquad (3.17)$$

Inequality (3.17) governs the convergence of the primal-dual variant of Karmarkar's algorithm discussed here. Now, given $x^{(k)} \in S$ and $z^k \leq z^*$, the (k+1)th iteration of the algorithm consists of computing lower bound $z^{(k+1)}$ such that $z^* \geq z^{(k+1)} \geq z^k$ and solution of the problem $P(\underline{z}^{+1})$ transformed from problem.

$$P(z^k): \min \bar{c}(z)\bar{D}x$$

$$\text{s.t.} \bar{A}\bar{D}\bar{x} = 0$$

$$e^T\bar{x} = n+1$$

$$\|\bar{x} - e\| \leq \beta = 0.3$$

where D = $\begin{bmatrix} diag(x^{(k)}) & 0 \\ 0 & 1 \end{bmatrix}$,

$\bar{x} \in \mathbb{R}^{n+1}$ and e=(1,1,...,)$^T \in \mathbb{R}^{n+1}$, is the starting feasible solution. The problem P(z) is solved by first computing orthogonal projection p of Dc(z) onto null space {x$\in\mathbb{R}^{n+1}$: $\bar{AD}$x=0,e^{T}x=0} of B = $\begin{bmatrix} AD \\ e^T \end{bmatrix}$.

i.e., p=(I-B^T(BB^T)$^{-1}$B)Dc(z)^T.

Then, we calculate the unit vector $\frac{p}{\|p\|}$ in the direction of p and obtain a point $\bar{x}$ = e - $\beta\frac{p}{\|p\|}$.

Finally, (k+1)th iterate x$^{(k+1)}$ is given by

$$x^{(k+1)} = \frac{diag(x^{(k)})\bar{x}\ [n]}{\bar{x}_{n+1}}$$

where $\bar{x}$ [n] denote the vector of first n components of $\bar{x}\in\mathbb{R}^{n+1}$. The algorithm stops where we meet the convergence check for the prescribed accuracy ϵ > 0l that is if

$$c\,x^{(k)} - z^k < \epsilon \le 0(2^{-L})$$

where $k \ge \left(\frac{n+1}{\delta}\ln(M(cx^{(0)} - z^o))\right) + 1$

Like Burrel and Todd [119], Ye [132] has obtained dual ellipsoid in the primal dual approach of Karmarkar's algorithm discussed above. The dual ellipsoid S_k at the kth iteration in the primal-dual approach of Karmarkar's algorithm is the set

$$S_k=\{U\in\mathbb{R}^{n+1}:\|(u-u^{(k)})D\| \le \epsilon_k \text{ and } \epsilon_k = cx^{(k)}-z^k \qquad (3.18)$$

which is contained in the row space.

R($\bar{A}$)={x $\in \mathbb{R}^{n+1}$:x=yA, y $\in \mathbb{R}$ } of the constraint matrix $\bar{A}$. This dual ellipsoid is centered at the current dual slack solution U($^{(k)}$ and it contains all of the optimal dual feasible solutions.

The volume $V(S_k)$ of the dual ellipsoid S_k is given as

$$V(S_k) = \frac{\pi(\epsilon_k)^{(n+1)}}{\det(\bar{D})} \qquad (3.19)$$

$$= \frac{\pi(cx^{(k)} - z^k)^{n+1}}{\prod_{i=1}^{n} x_i^{(k)}}$$

i.e., $\ln V(S_k) = (n+1) \ln (cx^{(k)} - z^k) - \sum_{i=1}^{n} \ln x_i^{(k)} + \ln \pi$

i.e., $\ln V(S_k) = P(x^{(k)}, z^k) + \ln \pi \qquad (3.20)$

i.e., $P(x^{(k)},z^k) = \ln \frac{V(S_k)}{\pi} \qquad (3.21)$

i.e., the potential function $P(x^{(k)},z^k)$ at the kth iteration which measures t progress of the algorithm equals the logarithm of the number which is times the volume of the current dual ellipsoid in Karmarkar's algorith

Let S_o be the initial dual ellipsoid in Karmarkar's algorithm,

$\ln V(S_o) = P(x^{(0)},z^o) + \ln \pi \qquad (3.22)$

Now, subtracting (3.22) from (3.20), we get

$\ln V(S_k) - \ln V(S_o) = P(x^{(k)},z^k) - P(x^{(0)},z^o) \leq -k\delta$

i.e., $\frac{V(S_k)}{V(S_o)} \leq \exp(-k\delta)$

Based on the above discussion and explications in Ye and T [132] we can see the following two major dissimilarities between t ellipsoid method and Karmarkar's algorithm:

(a) The ellipsoid in the ellipsoid method is uniquely defined in t space $\mathbb{R}^m$ whereas dual ellipsoid in Karmarkar's algorithm work row space R(A) of the constraint matrix, which is itself contair in the space $\mathbb{R}^n$. The initial volume of the ellipsoid in the ellips method takes the worst case bound as $n^4 2^n L$ which is a very la number even for small value of n, but the volume $V(S_o)$ of the in

dual ellipsoid S_0 in Karmarkar's algorithm cannot exceed the number $P(x^{(0)},z^0) + \ln \pi$ where $P(\hat{x}^{(0)}, \hat{z})$ is the value of potential function corresponding to the initial solution. Also the spaces in which these two methods work are different. In fact, the ellipsoid method works only in dual space whereas Karmarkar's algorithm works in both the primal and dual spaces which helps in accelerating the convergence for the primal and dual both.

(b) The steplength in the ellipsoid method is limited to a certain size; therefore, the theoretical shrinking ratio

$$\frac{V(E_{k+1})}{V(E_k)} < \exp\left(-\frac{1}{2(n+1)}\right) \qquad (3.23)$$

is precisely true for the ellipsoid method, but in Karmarkar's algorithm steplength can be increased by using line search technique in primal space to minimize the potential function. This results in much bigger reduction in the volume, assured by theoretical shrinking ratio

$$\frac{V(S_k)}{V(S_{k-1})} \leq \exp(-\delta) \qquad (3.27)$$

The key resemblance which emerges from the above discussion is that the volume of the dual ellipsoid in primal-dual approach of Karmarkar's algorithm shrinks uniformly as in the case of the ellipsoid method. We hope that further study of this resemblance can provide some proper grounding for the meaningful comparison between these two algorithms. Singh [108] has established similarity between the inequalities governing volume reduction in the ellipsoid method for linear inequalities and convergence in Karmarkar's algorithm.

When we apply Karmarkar's algorithm [58] to a linear programming

problem with n decision variables in standard form, then first it is transformed into the following canonical form:

$$\min c^T x$$

$$\text{S.t. } Ax=0$$

$$\sum_{i=1}^{n+1} x_i = 1$$

$$x \geq 0$$

where $c \in Z^{n+1}$, $A \in \mathbb{R}^{m\times(n+1)}$ and $x \in \mathbb{R}^{n+1}$.

If $x^{(0)}, x^{(1)}, \ldots, x^{(k)}$

be the sequence of points in $\mathbb{R}^{n+1}$, generated by Karmarkar's algorithm then it follows from Karmarkar's theorem 2 [58, p.380] that

$$f(x^{(k+1)}) \leq f(x^{(k)}) - \gamma$$

where γ is the fixed potential function decrease at each iteration which depends on the choice of steplength parameter α.

Putting k=0,1,..,k-1 in above inequality we get k inequalities. Adding these k inequalities we get

$$f(x^{(k)}) - f(x^{(0)}) \leq -k\gamma$$

$$\text{i.e., } \sum_{i=1}^{n+1} \ln \frac{c^T x^{(k)}}{x_i^{(k)}} - \sum_{i=1}^{n+1} \ln \frac{c^T x^{(0)}}{x_i^{(0)}} \leq -k\gamma$$

$$\text{i.e., } (n+1)\left(\ln c^T x^{(k)} - \ln c^T x^{(0)}\right) + \left(\sum_{i=1}^{n+1} \ln x_i^{(0)} - \sum_{i=1}^{n+1} \ln x_i^{(k)}\right) \leq -k\gamma$$

$$\text{Since } \sum_{i=1}^{n+1} \ln x_i^{(0)} \geq \sum_{i=1}^{n+1} \ln x_i^{(k)}, \text{ we have}$$

$$(n+1)\ln \frac{c^T x^{(k)}}{c^T x^{(0)}} < -k\gamma$$

i.e., $\ln \frac{c^T x^{(k)}}{c^T x^{(0)}} < \exp \frac{-k\gamma}{n+1}$ (3.24)

If we consider (3.23) governing volume-reduction in the ellipsoid algorithm and put k=0,1,...,k-1, we get k inequalities. On multiplying these k inequalities, we get

$$\frac{V(E_k)}{V(E_o)} < \exp\left(\frac{-k}{2(n+1)}\right) \quad (3.25)$$

McDiarmid [75] has obtained γ=0.693147178 using fixed steplength α (near equal to 1) at each iteration. Franklin [24] has achieved γ=0.30685 by taking α=0.5. Now if we choose α such that γ=0.5 then the right hand sides of the inequalities (3.24) and (3.25), governing convergence in Karmarkar's algorithm and volume-reduction in the ellipsoid method respectively become identical.

3.10 CONCLUDING REMARKS

As discussed in section 3.7 Karmarkar's algorithm has a polynomial complexity bound of $O(n^{3.5}L)$ which is lower than Khachian's complexity bound of $O(n^4L)$ by a factor of 0.5. Thus, from theoretical point of view this reduction in complexity bound is not significant. In practice difference becomes clear. Khachian's ellipsoid algorithm appears to require a number of iterations close to its worst case bound. On the other hand, a number of studies have established that many variants of Karmarkar's algorithm only take a number of iterations between 20 to 50 to get a very accurate solution, and this number appears to grow very slowly even with substantial increase in the number of variables in the linear programming problems. Thus this algorithm is very efficient to solve large linear programming problems. The efficiency

of this algorithm depends significantly on the steplength parameter α (discussed in substep 1.4 of the algorithm). Various choices for α have been proposed. It has been found that if we consider $\alpha > 1$ in substep 1.4 of the algorithm, its efficiency increases, but with this choice polynomiality of the algorithm cannot be assured. Simplex method for linear programming tests one extreme point at a time and makes local improvement whenever possible, but Karmarkar's interior point method provides us the ability to achieve global approximation to the set of all solutions. Probably this is one of the reasons that several types of interior point methods have been proposed following Karmarkar's seminal work [69]. These methods can be broadly classified in the following three categories:

Affine scaling methods,

Potential Reduction methods and

Central Trajectory methods.

CHAPTER 4

PATH-FOLLOWING METHODS

4.1 INTRODUCTION

In mathematical programming, algorithms can be interpreted as path-following procedures. This interpretation applies to the simplex method developed in Dantzig [18], Lemke's algorithms [71] for the complementarity problem, homotopy methods for piecewise linear equations and most of the methods for nonlinear optimization. More recent algorithms for linear programming by Murty [88] are also based on natural paths that lead to optimal solutions. Iterative algorithms for nonlinear optimization usually assign to any point x in a certain set $S \subset \mathbf{R}^n$ (usually convex) a "next point" $x' = \phi(x) \in S$. Given a starting point x^0, the iterative scheme generates a sequence of points $\{x^k\}$, where $x^{k+1}=\phi(x^k)$, that converges to a solution. If the algorithm generates the next point x' close to the preceding point x then the path may be a good approximation to the sequence generated by the algorithm. This is true at least during the later stages of the execution if the sequence converges to a solution point. If the algorithm makes large steps during the early stages then the path may be a bad approximation.

The study of solution paths is highly desirable for the design and analysis of the algorithms for optimization problems. Nazareth [90] interprets Karmarkar's algorithms [58] (discussed in chapter 3 of this thesis) as a homotopy method with restarts. Results about the infinitesimal version of Karmarkar's algorithm and related algorithms were

recently obtained in Bayer and Lagarias [11] and Megiddo and Shub [80]. Smale [113] showed that the path generated by the self-dual simplex algorithm developed in Dantzig [18] can be approximated by the Newton's method path for solving a certain system of nonlinear equations.

4.2 GEOMETRICAL ASPECTS OF THE CENTRAL PATH

The logarithmic barrier function technique, usually used in *nonlinear constrained optimization, can of course be applied to the linear* programming problem. This method recently came up in Barnes [9], where Karmarkar's algorithms [58] was analysed from the barrier function viewpoint, but the idea of using this function in the context of linear programming is attributed to Frisch [27]. The barrier function is

$$x \in \mathbb{R}^n,\ x>0 \rightarrow p(x) = -\sum_{i=1}^{n} \log x_i = -\log \prod_{i=1}^{n} x_i$$

This function *penalizes* variables that approach zero, and hence penalizes points near the boundary of the feasible region S determined by the constraints of the problem. The unique minimizer of p(·) in the interior S^o of the feasible region S is called *analytic center* of S, and it coincides with the point that maximizes the product of its components.

It is important to note that the Newton-Raphson's method can be used to determine the analytical center with a prescribed precision with excellent theoretical and practical performance. For the time being, we assume that an exact solution is available.

The main idea behind all interior point methods for minimization problems is that costs should be decreased and simultaneously move away from the boundary. So, we take the combinations of the cost and the barrier function usually known as the *internal penalized function* as follows:

$$\alpha \in \mathbb{R},\ x \in \mathbb{R}^n_{++} \rightarrow f(x) = \alpha c^T x + p(x),$$

where $\mathbb{R}^n_{++} = \{x \in \mathbb{R}^n,\ x > 0\}$

This function was extensively studied by Fiacco and McCormick in their book [23].

Now, associated to each value of the parameter $\alpha \in \mathbb{R}^n$, a *central point* $x(\alpha)$ is uniquely defined by

$$x(\alpha) = arg \min_{x \in S^n} [f_\alpha(x) = \alpha c^T x + p(x)] \qquad (4.1)$$

The curve $\alpha \in \mathbb{R} \rightarrow x(\alpha)$ is the *central path* for the linear programming i.e., the set $\{ | x(\alpha) = arg \min_{x \in S^n} [f_\alpha(x) = \alpha c^T x + p(x)] \}$ of all central points is called the central path for the linear programming problem

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } Ax = b \qquad (P) \\ &x > 0 \end{aligned}$$

The curve is smooth and as α increases, it converges to an optimal solution of (P).

The central path may be described in several ways corresponding to different parameterization of the curve. One of them has a simple

geometrical interpretation:

A central point is

$$x(\alpha) = arg \min_{x \in S^o} [f_\alpha(x) = \alpha c^T x + p(x)]$$

This point obviously solves the problem obtained by constraining the cost to its values at x(α).

$$x(\alpha) = arg \min_{x \in S^o} [f_\alpha(x) \{ p(x) \mid c^T x = c^T x(\alpha) \}]$$

This problem describes the analytic center of a constant-cost slice of the original feasible set S. Path-following algorithms follow the central path. Let $p_C(w^-,w^+) \rightarrow x(\rho)$ be a parameterization of the central path, with $w^+ > w^-$, w^+ possibly infinite.

All algorithms follow the model below:

ALGORITHM 4.2.1 Conceptual Path-following:

Given $x_o \in S^o$, $\rho_o(w^-,w^+)$, with $x^o = x(\rho_o)$

k:=0

REPEAT

choose $\rho_{k+1} > \rho_k$

call an internal minimization algorithm

to find $x^{k+1} = x(\rho_{k+1})$

k := k+1

UNTIL convergence.

This model simply generates a sequence of independent central points. Actual algorithms will depends on the parameterization, the initialisation (choice of ρ_o and x^o) and the criterion for updating the parameter. The parameterization and the updating rule characterize

different algorithms.

The internal algorithm is essentially the same for all methods. Finding a central point is a minimization problem with a nonlinear objective function with a simple Hessian matrix, and thus the natural choice is the algorithm of Newton-Raphson. In all cases we shall use an algorithm called scaling-steepest descent (SSD). This method is in some cases exactly equivalent to Newton-Raphson. The crucial point to be noted in relation to the internal algorithm is its termination rule.

To find an exact central point infinite number of iterations is almost impossible. Our intention is to achieve a polynomial algorithm. So, instead of determining the exact central points we work "near" the central path. If we are given a good criterion for measuring proximity, we can improve our model, as follows:

ALGORITHM 4.2.2 Implementable Path-following:

Given $x_o \in S^o$, $\rho^o \in (w^-, w^+)$ with x^o near $x(\rho_o)$

$k=0$

k:=0

REPEAT

choose $\rho_{k+1} > \rho_k$

Call an internal minimization algorithm to find x^{k+1} near $x(\rho_{k+1})$. The algorithm starts at x^k.

k = k+1

UNTIL convergence.

Fig. 4.1 shows two possible combinations of parameter updates and proximity criteria. In the first case, a short-step update forces the algorithm to trace the path, so that all points generated are near the path. In this case, the internal algorithm usually executes exactly one iteration per iteration of the main algorithm. In the second case, the parameter is updated by large steps, the several iterations of the internal algorithm are needed to approach the central point corresponding to ρ_{k+1}.

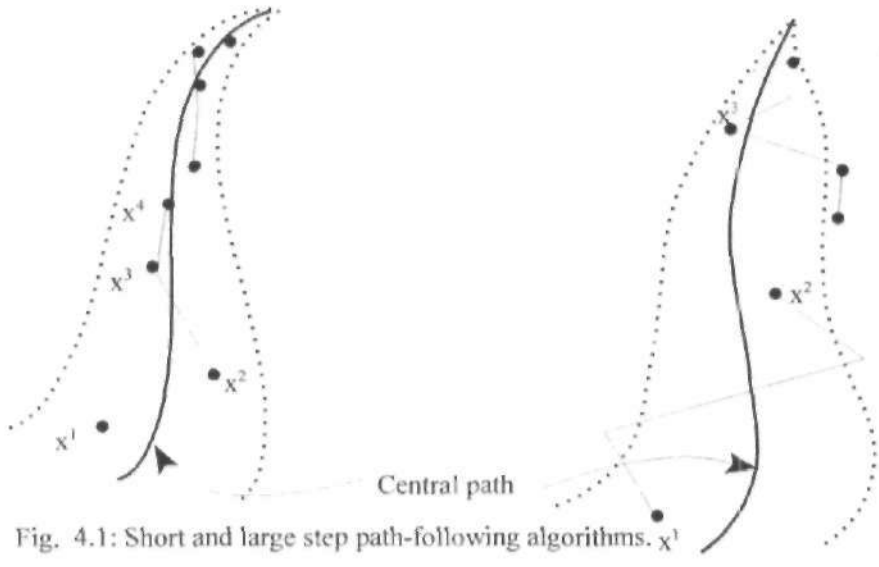


Fig. 4.1: Short and large step path-following algorithms.

4.3 TOOLS FOR GENERAL INTERIOR POINT METHODS

4.3.1 The Linear Programming Problem

In order to develop some useful tools, we consider a linear programming problem in standard form as follows:

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

$$\text{subject to } Ax=b \qquad (P)$$

$$x \geq 0,$$

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

We assume that the matrix A is of full-rank and that the feasible set S is bounded with nonempty relative interior S°. We also assume that an initial interior point $x^\circ \in S^\circ$ is at hand. This assumption in practice is replaced by an initial procedure that modifies the problem. A typical procedure is a big -M method like the one discussed in Adler, Karmarkar, Resende and Veiga [2]. With these assumptions, the problem has an optimal solution $\hat{x}$ and the optimal value of the problem be $\hat{v} = c^T \hat{x}$

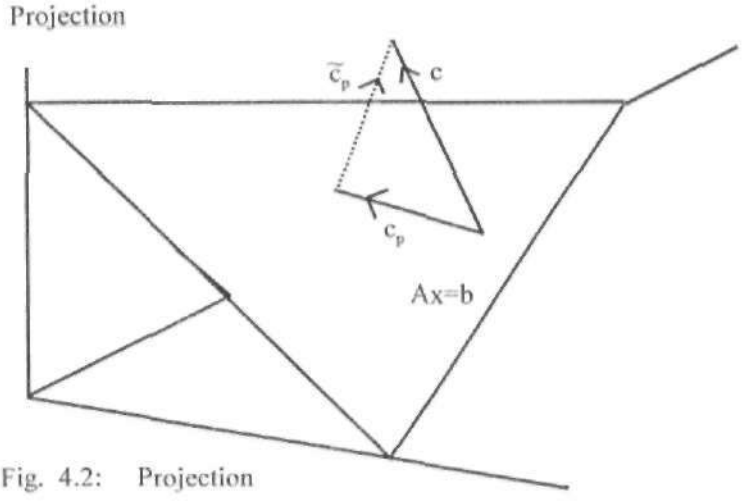


Fig. 4.2: Projection

Fig. 4.2 illustrates the problem (P) and the projection of price vector C into null-space $N(A) = \{x \in \mathbb{R}^n | Ax = 0\}$ of A and its orthogonal complement. The range space of A^T is denoted by $\Re(A^T)$ and is defined by

$\Re(A^T) = \{x \in \mathbb{R}^n | x = A^T w, w \in \mathbb{R}^m\}$.

Now any vector $d \in \mathbb{R}^n$ can be uniquely decomposed as $d = d_p + \tilde{d}_p$, where d_p $\in \mathbb{N}(A)$ and $\tilde{d}_p \in \Re(A^T)$.

The projection operator is linear and can be represented by a matrix P_A such that $d_p = P_A d$. The orthogonal complement $\tilde{d}_p = \tilde{P}_A d$ where $\tilde{P}_A = I - P_A$.

If A is of full rank, then

$P_A = [I - A^T(AA^T)^{-1}A]$,

where I is the identity matrix of appropriate order.

The projection of d into null space N(A) is the point in N(A) with smallest Euclidean distance to d.

$d_p = \text{argmin } \{\|x-d\| \mid x \in N(A)\}$

$\tilde{d} = \underset{z}{argmin} \{\|d - d^T w\| \mid w \in \mathbb{R}^m\}$

$\|d_p\| = \min \{\|d - A^T w\| \mid w \in \mathbb{R}^m\}$ (4.2)

The optimal set for (P) does not change if we replace the objective $c^T x$ by $c_p^T x$. It provides the steepest ascent direction for the cost from an interior point. Given any differentiable function $f: S^o \to \mathbb{R}$ and an interior point x, the steepest descent direction for f(.) from x is $-P_A \nabla f(x)$.

Dual Problem

The dual problem associated to (P) is

$$\begin{aligned} &\text{maximize } b^T w \\ &\text{subject to } A^T w + z = c \qquad (D) \\ &\qquad z \geq 0, \end{aligned}$$

where $z \in \mathbb{R}^n$ is the vector of dual slack variables.

Let $(\hat{w},\hat{z})$ be the optimal solution of (D) and $\hat{v}$ be the optimal value of the objective function b^Tw then $\hat{v} = b^T\hat{w}$.

The Duality Gap

Given any pair $(w,z) \in \mathbb{R}^{m+n}$ is feasible for (D), the duality gap $x^Tz = c^Tx - b^Tw$ is obtained by substituting $z=c-A^Tw$.

By the theorem 1.11.1 of the chapter 1, (Complementary Slackness Theorem), achieving optimality is equivalent to the condition $x^Tz = 0$.

The dual problem seems to have too many variables. In fact, the variables (w_i) can be eliminated, leading to a very convenient symmetrical primal-dual pair. This has been thoroughly studied in Todd and Ye [121], and we use a reduction procedure.

Lemma 4.3.1

$z \in \mathbb{R}^n$ is a feasible dual slack for (D) if and only if $z \geq 0$ and $P_Az = P_Ac$.

Proof:

Let $z \geq 0$ be any vector.

Then z is a feasible dual slack if and only if for some $w \in \mathbb{R}^m$

$$c - z = A^Tw$$

But $c-z = P_A(c-z) + A^Tw$

$\Rightarrow P_A(c-z) = P_Ac - P_Az = 0$. Hence proved.

The above lemma provides a simple rule for testing the feasibility of a dual slack. Some useful conclusion can be drawn as follows:

Given any point $\tilde{x} \in S$, an equivalent dual problem can be written as

$$\text{minimize } \tilde{x}^T z$$

$$\text{Subject to } P_A z = P_a c \qquad (4.3)$$

$$z \geq 0.$$

Here, the objective is the duality gap, and the optimal value is $\tilde{x}^T \tilde{z} = c^T \tilde{x} - \hat{v}$.

The primal problem can also be similarly modified: its objective can be replaced by $c_p^T x$ or by $\tilde{z}^T x$ for any feasible dual slack $\tilde{z}$.

The equivalent primal problem will be:

$$\text{minimize } \tilde{z}^T x$$

$$\text{subject to } Ax=b$$

$$x \geq 0$$

The dual feasible set and its interior for the problem (4.3) can be defined as follows:

$$Z = \{z \in \mathbb{R}^n \mid P_A z = P_A c,\ z \geq 0\}$$
$$Z^o = \{z \in \mathbb{R}^n \mid P_A z = P_A c,\ z > 0\}$$

4.3.2 The Scaling-Steepest Descent Algorithm

A scaling transformation on problem (P) is a change of variables $x = Dy$, where D is a positive diagonal matrix. Given a point $x^o \in S^o$, scaling about x^o is the scaling transformation $x = X_o y$, where X_o is the diagonal matrix whose diagonal elements are precisely the components of vector $x^o \in \mathbb{R}^n$, i.e., $X_o = \text{diag}\ (x_1^o, \ldots x_n^o)$. The linear programming problem scaled

about x^o is written as follows:

$$\text{minimize } \bar{c}^T y$$

$$\text{subject to } \bar{A}y = b \qquad \text{(SP)}$$

$$y \geq 0,$$

where $\bar{A} = AX_o$, $\bar{c} = X_o c$, are obtained by substituting $x = X_o y$ in (P). The point x^o is transported to e, the vector of ones. Scaling affects dual variables as follows:

Lemma 4.3.2

(w,z) is a feasible dual solution for (P) if and only if $(w, X_o z)$ is a feasible dual solution for (SP).

Proof

(w,z) is feasible for (P) if and only if $A^T w + z = c$, $z \geq 0$

Multiplying by X_o

$(Ax_o)^T w + X_o z = X_o c$, $X_o z > 0$.

This shows that $(w, X_o z)$ is dual feasible pair for (SP) and completes the proof.

Scaling does not change problem (P). Scaling affects the steepest descent direction as follows:

Let us consider a generalization of problem (P) for a differentiable objective f(·) as follows:

$$\underset{x \in S^o}{\text{minimize}} f(x).$$

The steepest descent direction (studied by Cauchy around 1840) is the direction that solves the minimization of the linear approximation of

$f(\cdot)$ about x^o over a unit ball with centre x^o.

$$\text{minimize}_{\epsilon} \{\nabla f(x^o)^T d \mid \|d\| \leq \epsilon,\ d \in N(A)\}$$

where $\nabla f(x^o) = \begin{pmatrix} \frac{\partial f(x^o)}{\partial x_1^o} \\ \vdots \\ \frac{\partial f(x^o)}{\partial x_n^o} \end{pmatrix}$ is a vector of partial derivatives

evaluated at x^o.

The optimal solution stems, as a consequence of the Cauchy-Schwarz inequality, and is always a multiple of

$$h = - P_A \nabla f(x^o).$$

The steepest descent direction is illustrated in Fig. 4.3 for a very simple problem with $S = \mathbb{R}^2_{+,+} = \{ (x_1, x_2) \in \mathbb{R}^2 : x_1 \geq 0, x_2 \geq 0 \}$.

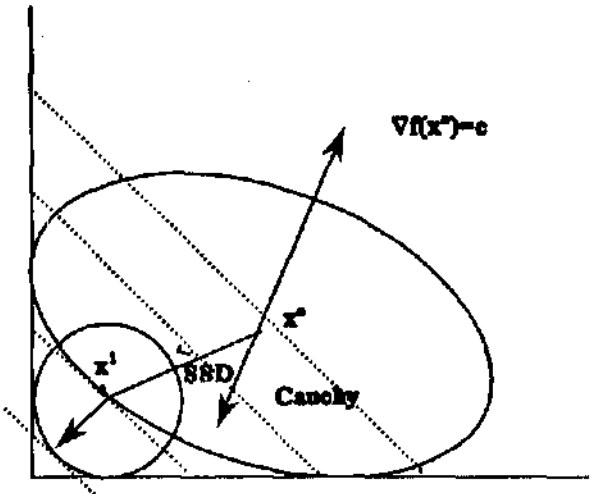


Fig. 4.3: Trust region minimization in Cauchy and SSD algorithms.

The steepest descent computation is known today as a trust-region minimization: a simple objective (linear approximation) is minimized in a simple region (a ball) to obtain a hint on the behaviour of the function around the point. A ball is chosen for trust region because it is easy and the presence of positivity constraints motivates the search for an easy region capable of reflecting the shape of the region of interest more precisely. The easiest large shape available is the largest possible simple ellipsoid in the positive orthant, shown in Fig. 4.3. The ellipsoid, with axes parallel to the coordinate axes, and hence simple, provides a large trust region when intersected with S.

Scaling the problem about x^o deforms this ellipsoid into a ball centered at $e=(1,...,1)^T$ and hence the solution of the trust region minimization is obtained by scaling followed by the projection of the resulting gradient vector (fig. 4.3).

This analysis results in a general first-order interior trust region minimization algorithm that can be used for any continuously

differentiable objective function. This algorithm is called Scaling-Steepest Descent (SSD).

ALGORITHM 4.3.3 (SSD)

Given $x^0 \in S^0$, f: $S^0 \to \mathbb{R}$ continuously differentiable.

K : =0

REPEAT

Scaling: $\bar{A}$: = AX_k, $\bar{g}$: = $X_k \nabla f(x^k)$

Direction: $\bar{h}$: = $-P_{\bar{A}}\bar{g}$

Line search: $\bar{y}$: = $e + \lambda \bar{h}$, $\bar{y} > 0$

Scaling: x^{k+1}: = $X_k \bar{y}$.

k: = k+1

UNTIL convergence.

The scaling transformation transports x^k to the vector e. The direction $\bar{h}$ minimizes the linear approximation of $y \to \bar{f}(y) = f(X^{-1}y)$ in a ball I(corresponding to the largest simple ellipsoid in the original space.

Affine Scaling Algorithm

An efficient algorithm for linear programming is obtained by the direct application of SSD to the original problem (P). This method is known as Affine-Scaling, first proposed by Dikin [20] in 1967. Dikin took always a step of length one in the line search, i.e., $\lambda = 1/\|\bar{h}\|$. Other researchers used large steps, a fixed percentage (above 95 percent) of the maximum possible steplength in the positive orthant. Like in any

interior point algorithm, the computational work is concentrated in the projection operation needed in each iteration.

The affine scaling algorithm was rediscovered by Barnes [9] and Vanderbei McKeton and Freedman [126] after publication of Karmarkar's algorithm [58]. The algorithm is globally convergent for problems with no primal degeneracy, as Dikin proved in 1974 [21] for unit steplength.

His proof was improved and clarified by Vanderbei and Lagarias [125] and extended to large steps by Gonzaga [45]. The method has been successfully implemented by Adler, Karmarkar, Resende and Veiga [2] and Monma and Morton [84]. The affine scaling algorithm is very attractive due to its simplicity and its good performance in practice. It is important to note that its performance is quite sensitive to the starting points. The proofs of convergence of the algorithm in the absence of degeneracy is fairly straight forward, but under degeneracy, such a proof is surprisingly long and difficult. There have not been any result on bounds on the efficiency, but due to analysis carried out in Megiddo and Shub [80] it is believed that the algorithm is not polynomial is worst case. But experimental results indicate that this algorithm competes favourably with the MINOS Code which implements the Simplex algorithm for linear programming.

The Search Direction

We wrote the SSD algorithm using an explicitly scaling operation at each iteration. This is not needed, since scaling was only a method for the trust region minimization; the search direction can be explicitly

expressed in the original space by

$$h = X_k \bar{h} = -X_k P_{AX_k} X_k \nabla f(x^k)$$

The SSD algorithm can be written directly in the original spaces as follows:

ALGORITHM 4.3.4 (SSD):

Given $x^0 \in S^0$, $f: S^0 \to \mathbb{R}$ continuously differentiable.

K=0

REPEAT

Direction: $h = X_k P_{AX_k} X_k \nabla f(x^k)$

Line search: $x^{k+1} := x^k + \lambda h$, $x^{k+1} > 0$

k: = k+1

UNTIL convergence

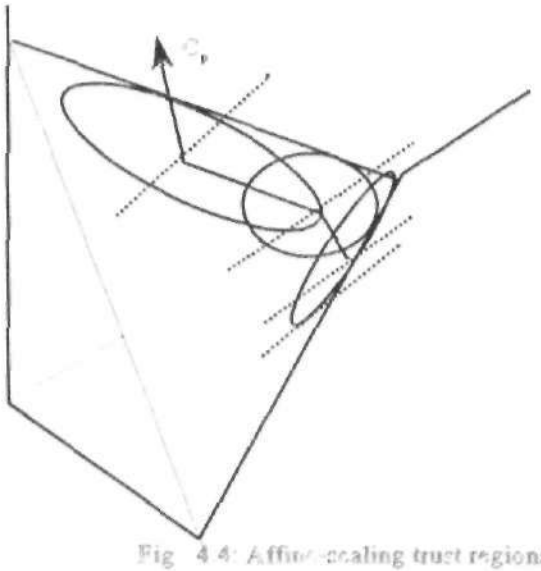


Fig. 4.4: Affine scaling trust regions

The affine scaling iterations are illustrated in fig. 4.4. The ellipsoidal trust regions are shown in original space: they correspond at

each point to the intersection of S and the largest simple ellipsoid in $\mathbb{R}^n$, centered at the point. Here we took unit steplengths.

4.3.3 Barrier Method

Barrier methods are applicable to

minimize f(x)

subject x $\in$ S,

where the constraint set S has a non-empty interior that is arbitrarily close to any point in S. This means that the set has an interior and it is possible to get to any boundary point by approaching from the interior.

The method works by establishing a barrier on the boundary of the feasible region that prevents a search procedure from leaving the region.

In the barrier methods a barrier function is added to the objective function which favours points interior to the feasible region over those near to the boundary.

We take the barrier function defined by

$$p: \mathbb{R}^n_{++} \to \mathbb{R}$$

$$p(x) = -\sum_{i=1}^{n} \log x_i$$

465646
QA
329.2
Qua

It has derivatives

$$\nabla p(x) = -x^{-1}, \nabla p(e) = -e \qquad (4.4)$$

$$\nabla^2 p(x) = X^{-2}, \nabla^2 p(e) = I$$

where $x^{-1} = \left(\frac{1}{x_i}, \ldots \frac{1}{x_n}\right)^T$

$$X^{-2} = diag\left(\frac{1}{x_1^2}, \ldots \frac{1}{x_n^2}\right)$$

RASHIM IBRAHIM LIBRARY
RECEIVED
- 1 FEB 2005
ZARIA NIGERIA
Ahmadu Bello University

Convexity:

$\nabla^2 p(x)$ is positive definite in S^o, so $p(\cdot)$ is strictly convex. Also p(x) grows indefinitely as x approaches the boundary of S. Thus the analytic center is well defined.

Effect of Scaling

If D is a positive diagonal matrix, we have

$$p(Dx) = p(x) - \sum_{i=1}^{n} \log d_i$$

Given two points $x^1, x^2 > 0$,

$$p(Dx^2) - p(Dx^1) = p(x^2) - p(x^1)$$

and hence scaling operations do not affect variations of p(.). Rather scaling yields easy derivatives. Moreover, at e the Hessian matrix is the identity, and the steepest descent direction from e coincides with the Newton-Raphson direction. Hence, the SSD algorithm and Newton-Raphson method with line searches coincide for the barrier function.

Linear approximation around e

To investigate the efficiency of algorithms we shall use linear approximations of the barrier function. Now, we like to establish a bound on the error of the linear approximation around e.

Lemma 4.3.5

Let $\lambda \in (-1, 1)$ be given.

Then $\lambda \geq \log(1 + \lambda) \geq \lambda - \frac{\lambda^2}{2} \frac{1}{1 - |\lambda|}$ (4.5)

Proof

The first inequality is a direct consequence of the concavity of the

logarithm. The second inequality was proved by Karmarkar [58], by developing the Logarithm in Taylor's series.

$$\log(1+\lambda) = \lambda - \frac{\lambda^2}{2} + \frac{\lambda^3}{3} - \frac{\lambda^4}{4} + \ldots$$

$$= \lambda - \frac{\lambda^2}{2}\left(1 - \frac{2}{3}\lambda + \frac{2}{4}\lambda^2 - \ldots\right)$$

$$\geq \lambda - \frac{\lambda^2}{2}\left(1 + |\lambda| + |\lambda^2| + \ldots\right)$$

$$= \lambda - \frac{\lambda^2}{2}\frac{1}{1-|\lambda|}$$

Hence proved.

Variation of the barrier function around e

Lemma 4.3.6

We consider a vector

$d \in \mathbb{R}^n$ such that $\|d\|_\infty < 1$,

where

$p(e+d) \geq \nabla p(e)'d = -e'd$.

$$p(e+d) \leq \nabla p(e)'d + \frac{\|d\|^2}{2}\frac{1}{1-\|d\|_\infty}$$

$$= e'd + \frac{\|d\|^2}{2}\frac{1}{1-\|d\|_\infty}$$

Proof

We have $p(e+d) = -\sum_{i=1}^{n} \log(1+d_i)$.

Since $d_i \in (-1, 1)$ by hypothesis, using the properties (4.15) and adding the inequalities for i=1 to n, we obtain the above results.

4.3.4 Analytic Center

An elegant method to avoid the boundary is to define a center for the polytope $S=\{x \in \mathbb{R}^n: Ax = b, x \geq o\}$ and to consider the following two objectives simultaneously: (i) reducing costs and (ii) centering.

The center of gravity is probably the best center, but its computation is a cumbersome task.

Another nice center is the center of an ellipsoid of maximum volume inscribed in polytope S, defining the feasible region for the linear programming problem. Although, this ellipsoid has been computed in polynomial time by Khachian and Todd [61] but, it is also equally cumbersome.

Apart from these above geometrical centers, a third good center has been defined by Vaidya [124] and can also be computed in polynomial time for a given precision. This is the volumetric center which is like before the center of a maximum volume ellipsoid in polytope S, but maximum is taken among the simple ellipsoids (intersection of S and ellipsoids with axes parallel to the coordinate axes). But it is also not much suitable for practical implementation. Thus above discussed centers are not much helpful in avoiding boundary of the feasible region S.

To the best of our knowledge, the most useful center to serve our purpose is the *analytic center*, proposed by Sonnevend [114]. A point x

belonging to the polytope S is called the analytic center if the barrier function $p = -\log\prod_{i=1}^{n} x_i$ attain its minimum at χ, i.e.,

$$\chi = \operatorname{argmin}_{x \in S^\circ} p(x).$$

Since the SSD algorithm coincides with Newton-Raphson method with line searches for the minimization of the barrier function, either method must be efficient for the determination of the analytic center.

4.3.5 Auxiliary Functions

Given a point in the interior S° of the polytope S, our aim is to obtain a relatively better point which is close to the center and ensures improvement in the objective function. Thus while constructing auxiliary function these two objectives (cost improvement and centering) will be taken into considerations.

Following the arguments outlined above, different auxiliary functions are constructed, each one leading to a different family of algorithms. Each auxiliary function uses some parameter that weighs in some way the importance given to each of the two objectives (cost improvement and centering). Each auxiliary function is associated to one parameterization of the central trajectory (or central path) given as follows;

(i) **The Penalized Function** (Frisch [27], Fiacco and McCormic [23]) - Parameter α associated to a duality gap:

$$x \in S^\circ \rightarrow f_\alpha(x) = \alpha c^T x + p(x)$$

(ii) **The Center Function** (Huard [52], Renegar [95]) - parameter k, upper bound to the optimal cost; $q \geq n$ constant:

$$x \in S^{\circ} \text{ s.t } c^T x < K \rightarrow f_k(x) = -q \log (K - c^T x) + p(x)$$

(iii) **The Potential Function** (Karmarkar [58]) - parameter v, lower bound for the optimal cost; q ≥ n constant;

$$x \in S^{\circ} \rightarrow f_v(x) = q \log (c^T x - v) + p(x)$$

In all the auxiliary functions, the second term is the barrier function which repels points trying to approach the boundary. The first term involves the cost and the parameter weighs both terms.

In (i) increasing α increases the importance of the cost term.

In (ii) decreasing K increases - log (K - $c^T x$).

In (iii) increasing v decreases log($c^T x$-v).

The gradients of these functions at $e=(1,\ldots,1)^T$ result from a scaling operations in algorithms) are respectively.

(i) $\alpha \bar{c} - e$

(ii) $\dfrac{q}{K - \bar{c}^T e} \bar{c} - e$

(iii) $\dfrac{q}{\bar{c}^T e - v} \bar{c} - e$

The steepest descent directions at $e=(1,\ldots,1)^T$ are all combinations of two vectors $-P_{\bar{A}} \bar{c}$ and $P_{\bar{A}} e$ called cost-reduction direction and centering direction respectively.

Given K or v, a value for α may be found such that $\nabla f_{\alpha}(e)$ coincides with $\nabla f_K(e)$ or $\nabla f_v(e)$ as $\alpha = \dfrac{q}{K - c^T x}$, $\alpha = \dfrac{q}{c^T x - v}$ (4.6)

Descent Directions for the Auxiliary Functions

Given an interior point x in the polytope $S = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ the descent directions in the original space will be denoted by $h(x,\rho)$, where $\rho \in \{\alpha, K, v\}$ and the corresponding directions is scaled space will be denoted by $\bar{h}(x, \rho)$ and defined by

$$\bar{h}(x,\rho) = -P_{AX} X \nabla f_{\rho}(x)$$

whereas the relation between the descent directions in original and scaled spaces will be given by

$$h(x,\rho) = X\bar{h}(x,\rho)$$

Now, if we consider the relationship between the SSD direction $h(x,\rho)$ and the Newton-Raphson step (NR step) for the penalized function, we shall see that these two directions coincide.

Lemma 4.3.7

Consider the function $f_{\alpha}(\cdot)$ for a fixed $\alpha \geq 0$. The NR step from x coincides with the SSD direction, given by

(i) $h(e,\alpha) = -P_A \nabla f_{\alpha}(e) = -\alpha c_p + e_p$.

(ii) $h(x,\alpha) = -XP_{AX} X \nabla f_{\alpha}(x) = -XP_{\bar{A}}(\alpha \bar{c} - e)$

Proof

Let x=e, initially,

Then the quadratic approximation of $f_{\alpha}(\cdot)$ about e has derivatives given by

$$\nabla f_{\alpha}(e+h) = \nabla f_{\alpha}(e) + \nabla^2 f_{\alpha}(e)h = \nabla f_{\alpha}(e) + Ih$$

Since $\nabla^2 f_{\alpha}(e) = \nabla^2 p(e) = 1$.

where I is the unit matrix of appropriate order.

The NR step corresponds to the minimizer of the quadratic

approximation, obtained by setting the projected gradient to zero:

$P\nabla f_\alpha(e) + h(e,\alpha) = 0$ is proved.

To prove (ii), it is important to note that NR algorithm is scale invariant. Because, the quadratic approximation of a function foes not depend on the metric of the space (in contrast with the norm-dependent steepest descent direction). So for $x \in S^\circ$, the NR step in scaled space will be computed as in (i) that is space.

$$\bar{h}(x,\alpha) = -P_{\bar{A}} \nabla f_\alpha(x)$$

Hence proved.

For the other auxiliary functions, the Hessian matrix is influenced by the first term and SSD is no longer equivalent to NR. But it can be interpreted as a quasi-Newton method in the following sense:

a quasi-newton method minimizes at each iteration a quadratic model of the function, which may differ from the Taylor expansion:

$$f_p(x+h) \approx f_p(x) + \nabla f_p(x)^T h + \frac{1}{2} h^T E h$$

The SSD algorithm uses $E=\nabla^2 p(x)$ instead of $E=\nabla f_p(x)$. Thus we loose the contribution of the second derivatives of the first term of the functions. This contribution is null in the penalized function and the methods coincide. For the potential function, the first term contributes a negative definite rank-one matrix that may destroy the positive definiteness of the Hessian matrix and thus it is ignored.

4.3.6 Guessing a Dual Slack and a Lower Bound

The parameter used by the potential function $f_v(\cdot)$ must be a lower bound to the value $\hat{v}$ of an optimal objective function value. A procedure for guessing a feasible dual slack and consequently a lower bound for $\hat{v}$ was presented in Gonzaga [46] which gives the same bounds as the methods developed by Todd and Burrell [119] and by Ghellaink and Vial [34] using projective geometry. The dual slacks generated by it have the same format as the ones used in all existent primal potential reduction methods. Given any feasible point $x \in S$, the procedure associated to it a lower bound $\tilde{v}(x) = -\infty$. If $\tilde{v}(x) = -\infty$, then the procedure fails and no dual slack is generated. The usefulness of the procedure will be ensured by the fact that a good lower bound will always be generated at points on or near the central path.

Let us suppose initially that $e \in S$. From lemma 4.3.1, we deduce that a vector $z \in \mathbb{R}^n$ is a feasible dual slack if and only if $z \geq 0$ and $z = c_p + \gamma$ where $\gamma \perp N(A)$. Our guess consists in trying to find a "very positive" vector $\gamma \perp N(A)$ and adding it to c_p to obtain a non-negative vector. The ideal guess would be γ proportional to e, but in general e is not orthogonal to N(A). We try γ proportional to $e-e_p$.

Let us define the vector $\tilde{a} \perp N(A)$ given by $\tilde{a} = \dfrac{e-p}{\|e-e_p\|^2}$

If for some $\mu \in \mathbb{R}$, $c_p - \mu\tilde{a} \geq 0$ then, $z = c_p - \mu\tilde{a}$ is a feasible dual slack. The duality gap associated to the primal-dual pair (e,z) will be

$$\Delta = e^T z = c_p^T e - \mu$$

Since $\tilde{a}^T e = 1$ because $(e-e_p)^T(e-e_p) = (e-e_p)^T e$.

Now, $v = c^T e - \Delta$ is a lower bound for $\hat{v}$. The best lower bound will correspond to the maximum admissible value for μ. We can now formalize the procedure, associating to each $x \in S$ a lower bound $\tilde{v}(x) \in [-\infty, \hat{v}]$ obtained by the procedure above after scaling the problem about x:

$$\tilde{\Delta}(x) = inf \left\{ e^T P_{\bar{A}} \bar{c} - \mu \mid P_{\bar{A}} \bar{c} - \mu \tilde{a} \geq 0 \right\} \qquad (4.7)$$

$$\tilde{v}(x) = c^T x - \tilde{\Delta}(x)$$

with the convention that inf $\phi = +\infty$.

If $\tilde{\Delta}(x) < +\infty$, then the procedure defines the dual feasible slack

$$\tilde{z}(x) = X^{-1}(P_{\bar{A}} \bar{c} - \tilde{\mu} \tilde{a}), \qquad (4.8)$$

where $\tilde{\mu}$ is the minimizer in (4.7).

4.4 THE CENTRAL PATH

In this section we discuss some properties related to parameterization of central path and present a complete treatment of conceptual path following algorithm for linear programming. The central path was first studied by Bayer and Lagarias [11]. Latter on, in 1989 Megiddo explicated the primal-dual properties of the central path in his notable contribution Megiddo [79]. This analysis proved important basic ingredient for several primal-dual interior path following algorithms.

If we consider penalized function $f_\alpha(x)=\alpha c^T x + p(x)$, defined on the interior S^o of the feasible region S, then as discussed in the section 4.2, the central point $x(\alpha)$ associated to $\alpha \in \mathbb{R}$ is defined as

$$x(\alpha) = \underset{x \in S^o}{argmin} f_\alpha(x)$$

Since $f_\alpha(\cdot)$ is always strictly convex and grows indefinitely close to the boundary of feasible region S, so the central points discussed above are well-defined. It is important to note that $x(\alpha)$ is a unique point in S^o, where

$$p\nabla f_\alpha(x) = 0.$$

Thus the central path is the curve $\alpha \in \mathbb{R}_+ \rightarrow x(\alpha) \in S^o$. It is important to note that the curve has been defined corresponding to non-negative value of α.

4.4.1 Primal-dual properties

As discussed earlier in chapter 1, the convergence of duality gap to zero ensures the convergence of the sequence of point generated by the algorithm to the optimal solution. Thus, it provides reliable termination rules for the algorithms for linear programming problems. In fact, it provides the driving force for the algorithms which try to reduce the duality gap at each iteration. Corresponding to central point $x(\alpha)$ for the penalized function $f_\alpha(x)=\alpha c^T x+p(x)$, the duality gap can be obtained using the following lemma.

Lemma 4.4.1

If $x(\alpha)$ be the central point associated to $\alpha > 0$, then

$$z(\alpha) = \frac{(x(\alpha))^{-1}}{\alpha}$$

is a feasible dual slack, and the duality gap associated to $x(\alpha)$ and $z(\alpha)$ is

$$x(\alpha)^T z(\alpha) = \frac{n}{\alpha}$$

where $(x(\alpha))^{-1} = \left(1/x_1(\alpha), \ldots, 1/x_n(\alpha)\right)$

Proof: From $p\nabla f_\alpha(x) = 0$, we get

$$\alpha c_p - px^{-1} = 0$$

i.e., $px(\alpha)^{-1} = \alpha c_p$

i.e., $\frac{px(\alpha)^{-1}}{\alpha} = c_p$.

Since $x(\alpha) > 0$, $\frac{x(\alpha)^{-1}}{\alpha}$ satisfies the necessary and sufficient conditions to be a feasible dual slack, prescribed in Lemma 4.3.1. The duality gap is $\frac{1}{\alpha} x(\alpha) x(\alpha)^{-1} = \frac{n}{a}$, completing the proof.

4.4.2 Parameterization of the Central Path

The auxiliary functions as defined in 4.3.5 generate parameterization $\rho \to x(\rho)$ of the central path. The main features of each parameterization are listed in the table 1 given below.

Description of Table 1:

The first row describes each auxiliary function at a point x in its domain.

The second row describes their gradient at x. If x is a central point then $p_A \nabla f_\rho(x) = 0$ which gives

$$\frac{q}{K - c^T x} \qquad \alpha = \frac{q}{c^T x - v}$$

TABLE 1: Features of Parameterizations of Central Path

	PARAMETERIZATION		
	PRIMAL	PRIMAL-DUAL	DUAL
$f_\rho(x)$	$-q\log(K-c^Tx)+p(x)$	$\alpha c^Tx+p(x)$	$-q\log(c^Tx-v)+p(x)$
$\nabla f_\rho(x)$	$\frac{q}{K-c^Tx}c-x^{-1}$	$\alpha c-x^{-1}$	$\frac{q}{c^Tx-v}c-x^{-1}$
Gap$\Delta(\rho)$	$\frac{n}{q}(K-c^Tx)$	$\frac{n}{\alpha}$	$\frac{n}{q}(c^Tx-v)$
Updated ρ(reduce gap)	$K'-c^Tx=\beta(K-c^Tx)$ $\beta\in(0,1)$	$\alpha'=\frac{\alpha}{\beta}$	$(c^Tx-v')=\beta(c^Tx-v)$, $\beta\in(\frac{n}{q},1)$
Update	$K'=\beta K+(1-\beta)c^Tx$		$v'=\beta v+(1-\beta)c^Tx$

The third row associates a duality gap $\Delta(\rho)$ to each of the parameterizations.

The fourth row shows the parameter updating for a central point $x(\rho)$ that is to take a step along the central path. The updating is to reduce the duality gap by the ratio $\beta\in(0,1)$. The row indicate how a parameter ρ' should be chosen to enforce a gap reduction $\Delta'=\beta\Delta(\rho)$.

Primal Parameterization

We consider the auxiliary function $x \in S^\circ$ s.t. $c^T x < K \rightarrow f_K(x) = -q\log(K - c^T x) + p(x)$ and define

$$K > \hat{v} \rightarrow x(K) = argmin\left\{ f_K(x) \mid x \in S^\circ, c^T x < K \right\}$$

This parameterization is called "primal" because the parameter K is an upper bound for the Primal objective function value. This characterization of central points was first used by Huard [52] in his famous work "method of centers" for nonlinear programming. It was the basis of Renegar's algorithm [95] for linear programming, the first method to achieve a complexity of $O(\sqrt{n}\, L)$ iterations. His method was modified by Vaidya [122] to prove a complexity of $O(n^3 L)$ arithmetical operations.

In this case, the definition of analytical center is extended to regions defined by inequality constraints by considering slacks for the inequalities. Thus the center of the set $\{ x \in \mathbb{R}^n \mid x \in S, c^T x \leq K \}$ is the minimizer of $-q\log(K - c^T x) + p(x)$. Here $x(K)$ is the analytic center of the region $\{x \in S \mid c^T x \leq K\}$ with the constraint $c^T x \leq K$ repeated q times.

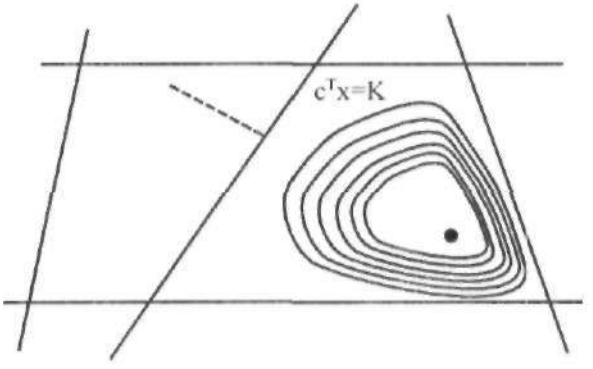


Fig. 4.5: Central points in primal parameterization.

Now we use this opportunity to explain why the center is "analytic" and not geometric. The inclusion of q copies of the constraint $c^Tx \leq K$ increases the weight of this constraint in the auxiliary function with the effect of pushing the center away from its active region. Renegar found the property that for $q \geq n$ the center is more than half way toward optimal solutions, that is, $c^Tx(K) \leq (\hat{v}+K)/2$, where $\hat{v}$ is the optimal value of objective function.

Since $f_K(x)$ is strictly convex function that grows indefinitely near the boundary of the restricted feasible region so this characterization of central points is well defined. Also the central point associated to $K > \hat{v}$ is uniquely determined by the condition

$$\frac{q}{K - c^T x} c_p - Px^{-1} = 0$$

where $x^{-1} = \left(\frac{1}{x_1}, \ldots, \frac{1}{x_n} \right)^T$.

The features of this parameterization are described in the first column of Table 1. We consider x = x(K) and the case of duality gap q=n, we have $\Delta(K)=K-c^Tx(K)$.

The duality gap equals the slack in the extra restriction $c^Tx \leq K$. This shows that c^Tx is lower than half way between $\hat{v}$ and K and provides plenty of room to decrease K, keeping that constraint $c^Tx \leq K$ inactive. There is still more room if q > n. The conceptual algorithm discussed in section is specialised by incorporating the new updating rule:

Set $K_{k+1} = \beta k_k + (1-\beta)c^Tx^k$.

The duality gap at $x(K_{k+1})$ will be such that $\Delta(K > \beta\Delta(K)$, since

$\Delta(K_{k+1})=(n/q)(K_{k+1}-c^Tx^{K+1})$ and $c^Tx^{K+1} < c^Tx^K$.

Although the actual duality gap reduction is smaller than the desired, but it does not have much adverse affect on the convergence of the algorithm. Since $(k-\hat{v})$ has a reasonable decrease at each iteration, as we see it in the following lemma:

Lemma 4.4.4

If we consider the central point x=x(K) for $K > \hat{v}$ and the center function $f_K(\cdot)$ with $q \geq n$ and define r=n/q and

$K'=\beta K+(1-\beta)c^Tx$.

Then
$$\frac{k' - \hat{v}}{K - v} \leq \frac{r+\beta}{1+r} \leq \frac{1+\beta}{2} \qquad (4.9)$$

and the gap is related to $K-\hat{v}$ by

$$\Delta(K) \leq K - \hat{v} \leq \frac{q+n}{n}\Delta(K) \qquad (4.10)$$

Proof: Let $v \leq \hat{v}$ be defined by

$v=c^Tx - \Delta(K)$ using

$\Delta(K) = r(K - c^Tx)$, (3rd row)

$K - v = K - c^Tx + r(K-c^Tx)$ (4.11)

Now using

$K' - c^Tx = \beta(K-c^Tx)$ in (4.11) (4th row)

we get

$K' - v = \beta(K - c^Tx) + r(K-c^Tx)$

Dividing this by (4.11),

$$\frac{K' - v}{K - v} \leq \frac{\beta + r}{1 + r} \leq \frac{1+\beta}{2} \text{ since } r \leq 1$$

Again, since $v < \hat{v} < K' < K$,

$$\frac{K'-\hat{v}}{K-\hat{v}} \leq \frac{K'-v}{K-v}$$ (4.9) is proved.

The left inequality of (4.10) stands because

$\Delta(K) = (n/q)(K-c^Tx) < K-\hat{v}$,

since $n \leq q$ and $K - c^Tx \leq K - \hat{v}$.

Also $K - \hat{v} \leq K - c^Tx + \Delta(K)$

But $K - c^Tx = (q/n)\Delta(K)$ proved.

We observe that $K_K - \hat{v}$ decreases by a fixed ratio per iteration and the above discussed lemma 4.4.4 defines the role of parameter q: the reduction in the quantity $(K_k - \hat{v})$ per iteration and thus the steplength in the path following algorithm can be arbitrarily chosen by sufficiently increased q.

Primal-Dual Parameterization

The Primal-Dual parameterization is based on the penalised function $f_\alpha(x)$. It describes the close relationship between the parameter α and the duality gap associated to a central point, i.e., $\Delta(\alpha) = n/\alpha$,

The central path based on this parameterization was first achieved by Fiacco and McCormick [23]. Gonzaga [42] exploited it to develop path-following algorithm with a complexity bound of $0(n^3L)$ arithmetical operations.

The Conceptual Algorithm 4.2.1 is updated for this function simply by specifying

"Choose $\rho_{k+1} > \rho_k$":

set $\rho_{k+1} = \frac{1}{\beta}\alpha_k$

where $\beta \in (0,1)$ is an arbitrary number.

After one iteration of the conceptual algorithm x^{k+1} is found, we have $\Delta(\alpha_{k+1}) = \beta\Delta(\alpha^k)$. The duality gap reduction obtained in one step of the algorithm is equal to the desired reduction in β. So, the method is realistic.

Dual Parameterization

Corresponding to auxiliary function $h(x,\rho) = X\bar{h}(x,\rho)$, we define the dual parameterization by

$$v < \hat{v} \rightarrow x(v) = argmin\{f_v(x) \mid x \in S^a\}$$

The dual parameterization is based on the potential function that Karmarkar [58] defined with q = n. It is called "dual" because the parameter v is a lower bound for the optimal value $\hat{v}$ of dual objective function and is always associated to the dual cost of a feasible dual solution.

Here, the objective is not convex, but we have

$$f_v(x) = \log \frac{(c^T x - v)^q}{\Pi_{i=1}^n x_i}$$

and the argument of the logarithm (the multiplicative potential function) is strictly convex as proved in Imai [53]. Since the logarithm is a monotonically increasing function, the minimizers of both potential functions coincide, and x(v) is well defined for any $v < \hat{v}$, then $f_v(.)$ grows indefinitely near the boundary of S. The central point x(v) is uniquely characterized by

$$\frac{q}{c^T x - v} c_p - P x^{-1} = 0$$

The duality gap associated with the central point x(v) is given by $\Delta(v) = (n/q)(c^T x - v)$.

So, for a given lower bound v, the central point x(v) defines a new lower bound v' given by $v' = c^T x - (n/q)(c^T x - v)$.

If n=q, then there is no gap reduction. If q > n, then any value of v' such that $c^T x - v' = \beta(c^T x - v)$, $\beta \in (n/q, 1)$ provides a new lower bound. This gives a safe range for β at a central point: $\beta \in (n/q, 1)$. The minimum value of β is

$(c^T x - \hat{v})/(c^T x - v)$, but this is not accessible. The minimum safe value of β is $\tilde{\beta} = \tilde{\Delta}(X)/(c^T - v)$, where $\tilde{\Delta}$ is the guessed gap. This would generate the largest possible steps. The Conceptual Algorithm 4.2.1 is specialised for the dual parameterization by the command

Set $v_{k+1} = \beta v_k + (1 - \beta) c^T x^k$,

where $\beta \in (n/q, 1)$.

Like the primal parameterization, the value of q determines the maximum allowed gap reduction per iteration, and consequently the maximum steplength in the path-following algorithm. As the lower bound is updated, there is an immediate gap reduction. At x^{k+1}, the gap will be still lower because $c^T x^{k+1} < c^T x^k$.

Interrelation among the methods

The three parameterizations are related by the following lemma.

Lemma 4.4.5

Let x be an interior point in the feasible region S then the following

statements are equivalent.

(i) $x = x(\alpha)$ for some $\alpha > 0$

(ii) $x = x(K)$ for some $K > \hat{v}$

(iii) $x = x(v)$ for some $v < \hat{v}$.

If either statement is satisfied, then the parameters are related by

$$\alpha = \frac{q}{K - c^T x} = \frac{q}{c^T x - v}$$

Proof follows from the comparison of necessary and sufficient conditions for centrality in the three cases.

4.4.3 Complexity of the Conceptual Path-following Algorithm

Since exact central points can not be exactly computed in finite time. So the algorithms will have to work with nearly central points. Let us assume that centering is done by an oracle in fixed time and examine the complexity of a path following algorithm.

The algorithm will stop when the duality gap is small. We assume that an initial central point $x^o = x(\rho_o)$ is given and that the updates are made according to Table 1. For primal and dual parameterizations, we assume that $q=O(n)$, $q \geq n$.

ALGORITHM 4.4.6

Conceptual Path-following:

Given $\rho_o, x^o=x(\rho_o)$, $\beta \in (0,1)$ and an integer $L > 0$ (for the dual parameterization, $\beta \in (n/q, 1)$.

k:=0

MADU BELLO UNIVERSITY
KASHIM IBRAHIM LIBRARY

REPEAT

Compute ρ_{k+1} by the update procedure in Table 1.

Call an oracle compute $x^{k+1}:=x(\rho_{k+1})$

k:=k+1

UNTIL $\Delta(\rho_k) < 2^{-L}$.

Significance of L

In complexity studies, we seek the order of magnitude of the number of computations (iterations, arithmetic operations, bit operations) needed to find an exact solution of the problem in the worst possible case. This number is computed in terms of the size of problem.

As discussed in chapter 2, the size of a linear programming problem was defined by Khachian [59] in course of devising the first polynomial algorithm for linear programming. This was further developed by Gács and Lovász [28], Bland, Goldfab and Todd [13], Megiddo [78], Schrijver [99] and Karmarkar [58]. If we assume that all data in the problem (P) are integer or rational (as rational data can be converted to integer data), then the size of the problem is defined as

$$L = l + n + 1.$$

Where l is the total length of the input i.e., the total number of bits used by the numbers in A, b, c and n is the number of variables in (P). As discussed in chapter 2, the feasible region $S \subset B(0, 2^L)$ i.e., feasible region S is contained in an sphere of radius 2^L.

Significance of 2^L

2^L is a very large number, greater than the result of any arithmetical

computation using data without repetitions and also greater than any minor determinant of constraint matrix A.

2^{-L} is a very small number and has the following properties:

(i) If x is a vertex of S, then no component can be in the interval $(0,2^{-L})$

(ii) If x is a vertex of S, then $c^T x - \hat{v} \notin (0, 2^{-L})$.

Proof of (i)

If x is a vertex of S, then $A_B x_B = b$, where $B \subset \{1,...,m\}$ are indices of a basis and A_B is a non-singular submatrix of A. Then

$$x_B = A_B^{-1} b = \frac{(cof(A_B))^T b}{\det(A_B)}$$

But $\det(A_B) > 2^{-L}$, and all entries in the numerator are integers. Hence proved.

Proof of (ii)

Can be done by using the integrality of the costs.

The stopping rule for all algorithms is precisely (ii) above. If an algorithms obtains a point such that $c^T x - \hat{v} < 2^{-L}$, then an exact optimal solution can be obtained through the following lemma:

Lemma 4.4.7 (Purification)

Given any point $x \in S$, there exists a procedure that computes a vertex $\tilde{x}$ of S such that $c^T \tilde{x} \leq c^T x$, in no more than $0(n^3)$ arithmetical operations. (Proof of this lemma can be found in Kortanek and Zhu [67]).

The algorithm can be stopped whenever a solution is found such

that $c^T x - \hat{v} < 2^{-L}$ i.e., the difference between $c^T x$ *and* $\hat{v}$ is sufficiently small.

In this case, a purification leads to a vertex $\hat{x}$ with $c^T \hat{x} - \hat{v} < 2^{-L}$ which implies that $c^T \hat{x} - \hat{v} = 0$.

For our conceptual algorithm, we set $\in = 2^{-L}$ and the stopping rule becomes $\alpha_k > n\, 2^L$.

Complexity of Algorithm 4.4.6

Lemma 4.4.8

Let the algorithm 4.4.6 starts with $\Delta(\rho_o) < 2^L$. Then the number of iterations after which algorithm obtains optimal solution is bounded by $0(L/\log \theta)$, where $\theta = 1/\beta$ for the primal-dual or dual parameterization, and $\theta = 2/(1+\beta)$ for the primal parameterization.

Proof:

(i) For $\rho=\alpha$ or $\rho=v$, we have after k iterations and before stopping, $2^{-L} \le \Delta(\rho^k) \le \beta^k \Delta(\rho_o)$. Taking logarithms in base 2, we obtain $-L \le k \log \beta + L$, or equivalently, $k \le 2L/\log(1/\beta) = 0(L/\log\theta)$. (Proved).

(ii) If $\rho = K$, we get by lemma 4.4.4.

$$K_K - \hat{v} \le \theta^{-k}(K_o - \hat{v})$$

Using again lemma 4.4.4

$$2^{-L} \le \Delta(K_k) \le \frac{q+n}{n} \theta^{k}(K_o)$$

Taking logarithms and using

$Q = 0(n)$, $-L \le -k \log \theta + 0(L)$.

or equivalently $k \le 0(L/\log \theta$ (Proved).

The choice of β: We have used parameter β in our conceptual path-

following algorithm and we have also seen that the bound on the number of iterations depends on β. Now we see also how its dependence/ independence influences the complexities of the algorithms.

(i) β independent of n:

In this case the complexity of the conceptual algorithm is 0(L), but the internal SSD algorithm may need a large number of iterations. We shall see that the best global complexity obtained for these large steps is 0(nL) SSD iterations.

(ii) β dependent on √n such that $1/\beta = 1 + v/\sqrt{n}$:

In this case, the number of iterations in the internal algorithms is dependent on v, and the global number of SSD iterations will be bounded by $0(v\sqrt{n}L)$.

For small values of v(short steps), it is possible to ensure exactly one SSD iteration per iteration of the path-following algorithm (to obtain nearly centered points). This scheme gives the best theoretical complexity bound obtained until now in terms of SSD iterations, and was obtained by Renegar [95]. Each SSD iteration has the complexity of one matrix inversion, used in the projection computation, $0(n^3)$ arithmetical operations, bringing the overall complexity to $0(n^{3.5}L)$ operations. A further reduction in this number of $0(n^3L)$ can be obtained by computing the inverses by rank-one updates in a scheme first used by Karmarkar [58] and later by Gonzaga [42] and by Vaidya [122].

Higher values of v keep the same theoretical complexity bound, and produce "not-so-short step methods" first proposed by Gonzaga [43,44]

44].

4.5 NEARLY CENTRAL POINTS

Since it is almost impossible to compute a central path x(ρ) exactly, it seems desirable to develop some criteria to decide when a point x is "near" to the central point x(ρ) so that the internal algorithm will then stop based on such a proximity criterion. Polak [94] devised an implementable barrier function nearly thirty eight years ago, using as stopping rule for the internal iterations for α_k a condition like $\| P\Delta f_{\alpha_k}(x) \| < \epsilon_k$, where $\epsilon_k > 0$ decreases with k. Similar rules were proposed by Fiacco and McCormick [23] and used in the implementation of their SUMT-3 and SUMT-4 computer codes, described in Mylander, Holmes and McCormick [99]. This last algorithm, with a proper choice of parameters, has recently been shown by Anstreicher [7] to be polynomial for linear and quadratic programming problems, with a bound of $0(\sqrt{n}\, L \log L)$ iterations.

We now see that Polak's proximity criterion is not satisfactory, and the main reason is that it is scale dependent. That is why it requires an increasing precision given by $\epsilon_k \to 0$.

We now discuss some suitable proximity criteria and conclude that a very convenient one that coincides with Polak's if the computation is preceded by a scaling transformation.

All our computations are scale-independent. In fact SSD precedes all computations by a scaling, and Newton-Raphson is naturally scale-

independent. Corresponding SSD directions h(x,ρ) and $\bar{h}$(x,ρ) in the original space and transformed space are connected by

$$h(x,\rho) = X\,\bar{h}\,(x,\rho).$$

If we define

$$\delta\,(x,\rho) = \|\,\bar{h}\,(x,\rho)\|$$

the length of the SSD (or Newton-Raphson step) in scaled space, the magnitude of real number δ(x,ρ) is a good measure of proximity from x to x(ρ).

4.5.1 Proximity

As we know that the computation of exact central points in finite time is almost impossible. So, instead of computing exact central points, we find it easy to work near the central path. Thus it seem highly desirable to develop some notion to measure nearness or proximity in terms of numerically identifiable quantity.

If χ denotes the analytic center of the feasible region S, then for χ=e, the potential function p(e+h) can be written as follows:

$$p(e+h) = p(e) + \nabla p(e)^T h + \frac{1}{2} h^T \nabla^2 p(e) h + 0(e)$$

We know that

p(e) = 0

∇^2p(e) = I (From 4.4)

And $\nabla p(e)^T h=0$ (because e is an analytic center)

Substituting these values, above equation reduces to

$$p(e+h) = \tfrac{1}{2}h^{T}h + 0(h)$$

$$= \tfrac{1}{2}h^{T}h + 0(h)$$

$$= \tfrac{1}{2}\|h\|^{2} + 0(h)$$

If we draw level curves for this function, as shown in fig. 4.6 we see that for small values of $\|h\|$ (small in comparison to 1).

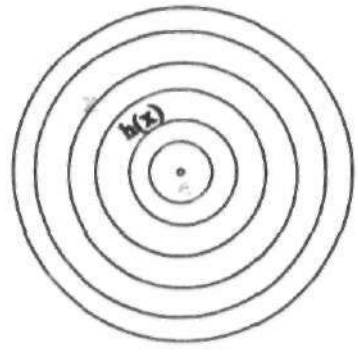


Fig. 4.6: Level curves for $p(\cdot)$ near the analytic center.

the level curves are almost spherical as a result of excellent behaviour of logarithm near 1.

This means, the quadratic approximation works very well near the center. It can be easily seen that for $\|h\| \in (0,0.3)$ even the maximum possible error in the quadratic approximation cannot exceed 30 per cent, which is independent of the dimension of the space.

Here, two magnitudes $\|x-e\|$ and $|p(x)-p(e)|$ appear good measures of the proximity to the center e. If either $\|x-e\|<0.03$ or $|p(x)-p(e)<0.05$, x is in a region where the quadratic approximation works well. It is important to note that the large values for these quantities does not provide any useful information except the fact that x is not near the central

path.

These two proximity criteria discussed above may not be much suitable for practical methods, as their computation heavily depends on the knowledge of the center. It seems reasonable to look for some other criteria for proximity. In our opinion Newton-Raphson step (NR step) h(x) from x can serve this purpose, as it does not depend on the prior knowledge of the center. This direction has been depicted in the above figure 4.6.

At those points where the quadratic approximation works well, NR step from x is a good approximation to the step to the point that minimizes the function i.e., $h(x) \approx e\text{-}x$ or $\|h(x)\| \approx \|x\text{-}x(\alpha)\|$.

This explication leads to three related quantities for a center at e, namely function variation, Euclidean distance, and length of NR step. It is important to note that function variation is scale-invariant whereas the Euclidean distance and length of NR step are norm dependent. Since all the criteria discussed above are similar near e, the extension of general case is obvious i.e., precede the computation of proximity by scaling. Thus given a point x, it stands a good chance of being near the analytic center χ (in terms of function values) if the norm of scaled NR direction is very very small in comparison to 1 i.e., $\|\bar{h}(x)\| << 1$.

The extensions of these results to the penalized function $f_\alpha(x)$ is quite simple. Since the penalized function $f_\alpha(x)=\alpha c^T x+p(x)$ differs from the barrier function p(x), by a linear term $\alpha c^T x$, so nothing changes in the analysis of second order approximation. Thus the proximity criteria in the

case of penalized function can be given as follows:

$f_\alpha(x)-f_\alpha(x(\alpha))$,

$\|X^{-1}(x-x(\alpha))\|$, and

$\delta(x,\alpha) = \|\bar{h}(x,\alpha)\|$,

in which first one is scale-invariant and the other two are norm independent.

4.5.2 Treatment of Proximity

As we have seen in section 4.1 that the central points posses nice primal-dual properties, the SSD direction allows the extension of these properties to a large region that contains the nearly central points. We shall use $h(x,\alpha)=X\bar{h}(x,\alpha)$ and the proximity criterion $\delta(x,\delta)=\|\bar{h}(x,\alpha)\|$ to prove the following lemmas:

Lemma 4.5.1

We consider a penalty parameter $\alpha > 0$ and a point x in the interior S° of the feasible region S.

If $\bar{h}(x,\alpha) < \epsilon$ then

$$z = X^{-1}\frac{e-\bar{h}(x,\alpha)}{\alpha}$$

is a feasible dual slack and the duality gap is

$$x^T z = \frac{n-e^T\bar{h}(x,\alpha)}{\alpha} \le \frac{n+\delta(x,\alpha)\sqrt{n}}{\alpha}$$

Proof

From the hypothesis, z>0. From Lemma 4.3.2, z is feasible for (P) *if and only if*

$$\bar{z} = \frac{e-h(x,\alpha)}{\alpha}$$

If feasible for the scaled problem. Now, in order to prove that z is feasible dual slack it will be sufficient to prove that $P_{\bar{A}}\bar{z} = P_{\bar{A}}\bar{c}$

(Lemma 4.3.1)

Now $P_{\bar{A}}\bar{z} = \dfrac{P_{\bar{A}}e - P_{\bar{A}}\bar{h}(x,\alpha)}{\alpha}$

But $\bar{h}(x,\alpha) \in N(\bar{A})$, where $\bar{A}$ is the constraint matrix obtained after scaling $P_{\bar{A}}\bar{h}(x,\alpha) = \bar{h}(x,\alpha) = -\alpha P_{\bar{A}}\bar{c} + P_{\bar{A}}e$.

Simplifying, we get $P_{\bar{A}}\bar{z} = P_{\bar{A}}\bar{c}$.

The value of the gap in the scaled problem (and also in the original problem) is $e^T\bar{z} = \dfrac{n + e^T\bar{h}(x,\alpha)}{\alpha}$.

The last inequality follows from

$e^T\bar{h}(x,\alpha) \le \| e \| \bar{h}(x,\alpha) \| = \sqrt{n}\ \delta(x,\alpha)$ completing the proof.

This lemma can be exploited to provide termination rules for algorithms and also for updating the lower bounds v in potential reduction algorithms. The particularization of the lemma to nearly central point is immediate.

Corollary 4.5.2

we consider a penalty parameter $\alpha \in R_+$ and a point x in the interior of the feasible region S. If $\delta(x,\alpha) = \| \bar{h}(x,\alpha) \| < 1$ then the conclusions of lemma 4.5.1 hold.

Proof We assume that $\delta(x,\alpha) - \| \bar{h}(x,\alpha) \| < \alpha$

Then the absolute value of each component of vector $\bar{h}(x,\alpha)$ will

be <1, i.e., $|\bar{h}_i(x,\alpha)| < 1$ for i=1,...,n.

Hence $\bar{h}(x,\alpha) > -e$ completing the proof.

The lemma above provides a method of guessing a dual slack (not necessarily feasible), associated with a given interior point $x \in S^0$ and a valid parameter value ρ, we introduce new notations for this slack and for the associated duality gap and lower bound. Here $d_p = P_A d$.

Primal-dual parameter α appears in the expressions given below

$$z(x,\rho) = X^{-1}\frac{e-\bar{h}(x,\alpha)}{\alpha} = X^{-1}(c_p + (e-e_p)/a)$$

$\Delta(x,\rho) - X^T z(x,\alpha)$ if $z(x,\alpha) \geq 0$;

$\Delta(x,\alpha) = +\infty$ otherwise.

$$v(x,\rho) = c^T x - \Delta(x,\alpha) \qquad (4.12)$$

For other parameterizations we substitute the expressions in (4.6).

Now computing the gap in the expressions above for a given point in all parameterizations,

we get

$$\Delta(x,\rho) = \frac{n-e^T\bar{h}}{\alpha} = \frac{n-e^T\bar{h}}{q}(K-c^T x) = \frac{n-e^T\bar{h}}{q}(c^T x - v), \qquad (4.13)$$

where $\bar{h} = \bar{h}(x,\rho)$, provided that $\Delta(x,\alpha) < \infty$. Thus, we have three closely related definitions for these entities $z(x,\rho)$, $\Delta(x,\rho)$ and $v(x,\rho)$ which are associated with any $x \in S^0$ and valid by lemma 4.5.1. also $z(\rho) = z(x(\rho),\rho)$, $\Delta(\rho)=\Delta(x(\rho),\rho)$, $v(\rho) = v(x(\rho)\rho p)$ are

associated with central points by lemma 4.4.3. $\tilde{z}(x)$, $\Delta(\tilde{x})$ and $\tilde{v}(x)$ are associated with any $x \in S^o$ by (4.7) and(4.8). In such a situation it is easy to see that $\tilde{\Delta}(x)=\inf\{\Delta(x,\alpha) \mid \alpha > 0\}$ (4.14)

Now, we discuss the following lemma which is very useful in the analysis of primal-dual methods and also relates proximity criterion discussed in this thesis to the criterion in Roos and Vial [98].

Lemma 4.5.3

We consider a penalty parameter $\alpha>0$ and a point $x \in S^o$. Then

$$\delta(x,\alpha) = \min_{z} \{ \| \alpha Xz - e \| \mid A^T w + z = c,\ w \in \mathbb{R}^m \}$$

and the minimum occurs at $z(x,\alpha)$.

Proof

From (4.2) we know that for any given vector d and $\alpha>0$,

$\|P_{\bar{A}} d\| = \min \{d - \alpha \bar{A}^T w \mid w \in \mathbb{R}^m\}$. Hence

$$\delta(x,\alpha) = \|\bar{h}(x,\alpha)\| = \min\{ \| \alpha \bar{c} - e - \alpha \bar{A}^T w \| \mid w \in \mathbb{R}^m \}$$

$$= \min \{\|\alpha X(c-A^T w)-e\| \mid w \in \mathbb{R}^m\}$$

$$= \min \{ \|\alpha Xz - e\| \mid A^t w + z = c, w \in \mathbb{R}^m\}$$

we have to prove that the minimizer is $z(x,\alpha)$ i.e., that $\delta(x,\alpha)=\|\alpha X z(x,\alpha)-e\|$. This can be achieved by substituting the expression for $z(x,\alpha)$ from (4.12). This last equality becomes $\delta(x,\alpha) = \|\bar{h}(x,\alpha)\|$. (Proved).

The treatment of proximity can be found in almost all papers that deal with path following algorithms. Since 1987, several works dealing with proximity appeared. Now we summarise some important results in the following lemma:

Lemma 4.5.4

We consider a penalty parameter $\alpha>0$, a point $x\in S^0$ and its proximity $\delta=\delta(x,\alpha)$.

(i) Efficiency of the SSD step:

If $\delta<1$, then $\tilde{x} = x+h(x,\alpha)$ is feasible and $\delta(\tilde{x},\alpha)\leq\delta^2$.

(ii) Proximity of Euclidean distance:

If $\delta<1$, then $\| X^{-1}(x-x(\alpha))\| \leq \frac{\delta}{1-\delta}$.

(iii) Proximity and function values:

If $\delta<1$, then $f_\alpha(x) - f_\alpha(x(\alpha)) \leq \frac{\delta^2}{1-\delta^2}$.

(iv) Guaranteed descent:

$\tilde{x} = x+(1/(1+\delta))h(x,\alpha)$ is feasible and

$f_\alpha(\tilde{x})\leq -f_\alpha(x)-(\delta-\log(1+\delta))$.

(Proofs of these results can be found in Hertog, Roos and Vial [51], Roos and Vial [97] and Gonzaga [43]).

In our judgement, the best of all proximity results, are the one achieved in Roos and Vial [97]and Hertog *et al.* [51], stated as parts (i) and (iv) of the above lemma 4.5.4.

Comments on various items of the lemma:

Property exhibited in item (i) of the lemma 4.5.4 seems to be the most powerful of all items in the lemma. It states that NR step decreases

in a quadratic rate from any initial point such that $\delta(x,\alpha) < 1$. This means that SSD or NR algorithms reduce the proximity in a sequence of values $\delta,\delta^2,\delta^4,\ldots,\delta^{2^k}$.

Thus as the iterations go high it compel newly generated iterations to remain close to the central path. Parts (ii) and (iii) of the lemma 4.5.4 show that the concept of proximity is well defined from a geometrical point of view. Part (iv) of the lemma 4.5.4 all the information needed for non proximal points. If $\delta(x,\alpha)$ is large then one SSD iteration results in a substantial decrease in the objective function value.

The lemma above applies only to the primal-dual parameterization. All these properties can be adapted to the center function, either by using the Newton-Raphson direction instead of the SSD direction, or by a simple transformation in the problem. There are no results similar to (i), (ii), (iii) for the potential function, but (iv) is trivial because its first term $\log(c^Tx-v)$ is strictly concave.

4.6 CENTRALIZATION

In this section we discuss centralization in the context of various types of parameterization. The centralization problem is defined as: given $x^o \in S^o$ and $\epsilon \in (0, 0.5)$ find a point $x \in S$ such that

$$\delta(x,\rho) \equiv \| \bar{h}(x,\rho) \| < \epsilon,$$

where ρ is a parameter. If the proximity measure $\delta(x,\rho) < \epsilon$ then the point x is called "nearly centered". Although the choice of ϵ is quite free in the range $0 < \epsilon < 0.5$ depending on the algorithm but the most popular values

that work well are around $\epsilon = 0.5$.

4.6.1 Finding the Analytic Center

When the objective function is simply the barrier function $p(\cdot)$ (identical with $f_\alpha(\cdot)$ for $\alpha=0$), we are looking for an approximation for the analytic center χ of S. The proximity measure is given simply by $\delta(x) = \|P_{Ax}e\|$. This problem was solved by Vaidya [123] using the SSD algorithm, or equivalently, NR algorithm with line searches.The number of iterations depends on ϵ and on $p(x^0) - p(\chi)$.

The convergence of any algorithm depends on the nature of the sequence of points generated by the algorithm. Now, we discuss the properties of the sequence generated by SSD algorithm initiated at some interior point x^0 of the feasible set S in the following lemma.

Lemma 4.6.1

Let $x^0 \in S^0$ and $\epsilon \in (0,0.5)$ be given. The sequence (x^k) of points generated by the SSD algorithm initiated at x^0 has the following properties:

(i) x^k is nearly centered i.e.,

$$\delta(x^k) \leq 0.5$$

for $k \geq k_1 = 11(p(x^0) - P(x))$

(ii) $\delta(x^k) \leq \epsilon$ for $k \geq k_1 + \log_2|\log_2\epsilon|$.

Proof:

(i) It is a direct consequence of lemma 4.5.4 (iv) which ensures a descent of 0.5-log 1.5 > 0.094 for points not nearly centered. The number of iteration at such points can

not exceed $(p(x^0)-p(x))/0.094$.

Completing the proof of (i).

(ii) Let us assume that $\delta(x^{k_1}) \leq 0.5$.

Then $\delta(x^{k_1+j}) \leq 0.5^{2^j}$ by lemma 4.5.4(i).

For any iteration before $\delta(x^{k_1+j}) \leq \epsilon$, $0.5^{2^j} > \epsilon$.

Taking logarithms twice,

$$2^j < -\log_2 \epsilon \,,\; j < \log_2 | \log_2 \epsilon |$$

Completing the proof.

We conclude that for ϵ=0(1) an approximate center with $\delta(x) \leq \epsilon$ is found in 0(p(x°)-p(x)) iterations.

4.6.2 Centralization in Primal-Dual Parameterization

Since $f_\alpha(\cdot)$ differs from $p(\cdot)$ by a linear term $\alpha c^T x$ the result above on the performance of Newton-Raphson's method can be immediately extended to the computation of nearly central points. Again, we conclude that for ϵ=0(1) the SSD algorithm starting at $x^\circ \in S^\circ$ finds x such that $\delta(x,\alpha) < \epsilon$ in $0(f_\alpha(x^\circ)-f_\alpha(x(\alpha)))$ iterations.

The situation created by the path-following Algorithm 4.2.2 is the following: a point x^k near $x(\alpha_k)$ is known and SSD is called to find a point x^{k+1} near $x(\alpha_{k+1})$, where $\alpha_{k+1}=\alpha_k/\beta$, $\beta<1$. This situation is reduced to the one above, and solved using the following lemma,

Lemma 4.6.2

Let us suppose that $x^\circ \in S^\circ$ and $\alpha>0$ are given and $\delta(x,\alpha)<0.5$. Let $\alpha'=(1+\mu)\alpha$, $\mu>0$. Then

(i) if $\mu<0.1/\sqrt{n}$ then SSD finds x such that $\delta(x,\alpha')\leq 0.5$ in one iteration;

(ii) Otherwise, SSD finds x such that $\delta(x,\alpha') \leq 0.5$ in $0(n\mu^2/(1+\mu))$ iterations.

(Proof of this lemma can be found in Gonzaga [43].

4.6.3 Centralization in Primal Parameterization

In primal parameterization, the objective function

$$f_K(x) = -q \log (K - c^T x) - \sum_{i=1}^{n} \log x_i$$

differs from the barrier function p(x) by a nonlinear term $-q\log(K-c^Tx)$, and thus SSD and Newton-Raphson do not coincide. Although it is possible to prove results similar to those in Lemma 4.5.4 for this function. The minimization of $f_K(\cdot)$ is equivalent to the computation of the analytic center of the region defined by S° with the extra constraint $c^Tx \leq K$, repeated q times. This problem can be put in our formats (with equality constraints) by defining q slack variables $x_{n+1},\ldots,x_{n+q}$, all constrained by $x_{n+j}=K-c^Tx$, and now the objective function becomes a barrier function

$$f_K(x) = -\sum_{i=1}^{n+q} \log x_i.$$

containing q more variables $x_{n+1},\ldots,x_{n+q}$, than $p(\cdot)$.

The application of SSD to this transformed problem is equivalent to Newton's method, and all results obtained for the primal-dual parameterization can be naturally extended to this case. This reduction of the centering problem is trivial if the original linear programming problem is defined with inequality constraints.

Results equivalent to lemma 4.6.2(i) and (ii) have also been proved in Renegar [95] and Renegar and Shub [96], and Hertog *et al.* [50,51].

4.6.4 Centralization in Dual Parameterization

The first term q log ($c^T x$-v) of the potential function q log ($c^T x$-v)+p(x) is not convex, and thus the results in lemma 4.6.2 cannot be extended to this case. In potential reduction algorithms SSD is used and thus the complexity analysis must follow a different path. This is probably one of the reasons why the complexity of the methods belonging to this class was lowered to $O(\sqrt{n}\,L)$ iterations only two years later than for the other approaches.

4.7 IMPLEMENTABLE PENALTY FUNCTION AND CENTER METHODS

Before going into the details of implementable penalty function method, we discuss some of the basic ingredients used in the construction of the algorithm.

The implementable path-following algorithm 4.2.2 will be completely specified by using the updating formulae in chapter 4.

The word "near" will be placed by the proximity criterion developed in 4.5.1. The complexity analysis for primal-dual and primal parameterization can be carried out by merging the results for the outer algorithm in section 4.4.3 and for the centralization methods described in section 4.6.

Initialization

in order to initialize an algorithm we need an initial parameter value ρ_o and a starting point x^o near$_o$x(ρ). This can be obtained by a centralization starting at a given feasible point. An initial feasible point is

usually obtained by a problem manipulation (Big-M Method), like in Adler, Karmarkar, Resende and Viega [2]. It is also possible to manipulate the original problem to obtain a nearly central point directly, as achieved in Gonzaga [42] or in Monteiro and Adler [85] but these approaches do not work well in practice.

The Implementable Algorithm

The algorithm below uses any of our three parameterizations discussed in section 3.4. It has a general termination rule based on the duality gap $\Delta(x,\rho)$ defined in (4.12). This gap replaces $\Delta(\rho)$ in the conceptual path following algorithm 4.4.6 discussed in section 4.4 of this thesis.

ALGORITHM 4.7.1

(Implementable path-following algorithm).

Given $\epsilon \in (0,0.5]$, $\beta \in (0,1)$, an admissible initial parameter ρ_o and $x_o \in S^o$ such that $\delta(x^o,\rho_o)<\epsilon$

k:=0

REPEAT

Compute ρ_{k+1} by the update procedure in Table 1.

Call the SSD or Newton-Raphson Algorithm to find

$x^{k+1} \in S^o$ such that $\delta(x^{k+1}, \rho_{k+1})<\epsilon$.

K:=k+1

UNTIL $\Delta(x^k, \rho_k) < 2^{-L}$.

4.7.1 Penalty Function Methods

Soon after Karmarkar published his seminal paper [58], Gill,

Murray, Sounders, Tomlin and Wright [35] established its resemblance to the traditional barrier function method. Gonzaga [42] described a short-step algorithm that follows the central path based on the penalized function $f_\alpha(x)$ which we have discussed in section 4.5.3 of this thesis. This short-step algorithms solve problem (P) in $0(\sqrt{n}\ L)$ SSD iterations, with an overall complexity of $0(n^3L)$ arithmetical operations.

The need for short steps was eliminated by Roos and Vial [98] and Gonzaga [43], who proved a bound of 0(nL) SSD iterations for parameter updates $\alpha_{k+1} = (1+\mu)\alpha_k$, where μ is an arbitrary constant. It has also been observed in Gonzaga [43] that the complexity bound of $0(\sqrt{n}\ L)$ iteration for this algorithm can be achieved, provided that we set $\mu=0(1/\sqrt{n})$.

Short-Step Method

Construction of a short-step method is an application of Lemma 4.6.2(i) in the algorithm 4.7.1 with $\epsilon=0.5$ and the update rule $\alpha_{k+1}= (1+0.1/\sqrt{n})\alpha_k$.

kth iteration starts with x^k near $x(\alpha_k)$, updates α and in one SSD iteration finds x^{k+1} near α_{k+1}.

Lemma 4.7.2

Let us assume that the short-step algorithm starts with $\Delta(\alpha_o) < 2^L$. Then it stops in $0(\sqrt{n}\ L)$ SSD iterations.

Proof

From lemma 4.6.2(i), the number of SSD iterations equals the number of outer iterations of the algorithms. From Lemma 4.4.8, the outer algorithm obtains $\Delta(\alpha_k) < 2^{-(L+1)}$ in

$$K = 0\left(\frac{L}{\log(1+\mu)}\right)$$

iterations, where $\mu = 0.1/\sqrt{n}$.

But $\log(1+\mu) \geq \mu/(1+\mu)$, and hence

$$K = 0(L(1+\mu)/\mu) = 0(10L(\sqrt{n}+0.1) = 0(\sqrt{n}L)$$

After $K=0(\sqrt{n}\,L)$ iterations, $\Delta(\alpha_K)<2^{-(L+1)}$.

Using Lemma 4.5.1,

$$\Delta(x^K, \alpha_K) \leq \frac{n+\epsilon\sqrt{n}}{\alpha_K} \leq 2\frac{n}{\alpha_K} = 2\Delta(\alpha_K) < 2^{-L},$$

and the algorithm stops, completing the proof.

Thus, the updating formula $\alpha_{k+1} = (1+0.1/\sqrt{n})\alpha_k$ works well, but larger steps could still lead to $\delta(x^k, \alpha_{k+1}) < \sqrt{\epsilon}$, with the consequence that $\delta(x,\alpha_{k+1})<\epsilon$ after one SSD iteration. The "largest possible short steps" are obtained by computing α_{k+1} such that the proximity measure $\delta(x^k, \alpha_{k+1})$ exactly equals $\sqrt{\epsilon}$. The computation of α_{k+1} is easy and can be seen in the largest short-step algorithm given below:

ALGORITHM 4.7.3 (Largest short steps algorithm)

Given $\alpha_o > 0$, $\epsilon \in (0,1)$, $x^o \in S^o$

such that $\delta(x^o,a^o) \leq \epsilon$

k:=0

REPEAT

scaling: $\bar{A}=AX_k$, $c_p=P_{\bar{A}}Xc$, $e_p=P_{\bar{A}}e$.

Update: compute α_{k+1}, such that $\|\alpha_{k+1} c_p - e_p\| = \sqrt{\epsilon}$.

Next point: $y=e-\alpha_{k+1} c_p + e_p$

scaling: $\alpha^{k+1} = X_k y$

k:=k+1

UNTIL $\Delta(x^K, \alpha_k) < 2^{-L}$.

It is important to note that all the steps here have approximately the same length in the Euclidean metric after scaling (Lemma 5.4(ii)). In the original space, the steplength measured at each point in the metric induced by $r \to \| X_k^{-1} r \|$ is approximately constant. This defines a Riemanian metric, and the length of the central path between $\alpha=0$ and $\alpha=2^{-L}$ in this metric will be bounded both below and above by numbers proportional to the number of steps of the algorithm above.

Large-Step Method

A large step method is obtained by using the following formula:

$\alpha_{k+1} = \alpha_k(1+\mu)$, $\mu > 0.1/\sqrt{n}$

The complexity analysis of this algorithm is very much similar to the short-step case.

Lemma 4.7.4

If the short-step algorithm starts with $\Delta(\alpha_o) < 2^L$, then it stops in 0(nl) SSD iterations.

Proof:

From Lemma 4.4.8, the number of outer iterations is bounded by

$$K=0\left(\frac{L}{\log(1+\mu)}\right) = 0\left(L\frac{1+\mu}{\mu}\right)$$

By Lemma 4.6.2(ii) the number of SSD steps in each iteration is $0(n\mu^2/(1+\mu))$. Multiplying this number by K, we immediately get the global bound of 0(nl) SSD steps, completing the proof.

The theoretical complexity grows with the steplength.

In this connection we discuss the following two cases as follows:

(i) "Not-so-short-steps": if $\mu=v/\sqrt{n}$ with $v=0(1)$, then the complexity is bounded by $0(v\sqrt{n}L)=0(\sqrt{n}L)$. So, the complexity of $0(\sqrt{n}\,L)$ is kept for any $\mu=0(1/\sqrt{n})$ but it decreases as μ increases.

(ii) "Large-steps": if $\mu=0(1)$, then the complexity is $0(nL)$.

This last case was independently studied by Roos and Vial [98] who presented a very elegant complexity proof.

I.7.2 Methods of Centers

The most natural setting for the methods of centers is the formulation of the linear programming problem with inequality constraints, that can be obtained by dualizing the problem.

We consider the following linear programming problem:

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

$$\text{subject to } Ax \le b.$$

$$x \ge 0$$

In this formulation, the barrier function would be

$$p(x) = -\sum_{i=1}^{m} \log (b-Ax)_i.$$

The center function for this formulation would be $f_K(x) = -q \log [K - c^T x] - \sum_{i=1}^{m} \log (b-Ax)_i$, and this is the barrier function for the problem

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

$$\text{subject to } Ax \le b$$

$$c^T x \le K$$

with the extra constraint repeated q times.

This construction was made by Renegar [95]. Each iteration of a path-following algorithm is now a search for the analytic center of polyhedron, which is a well-understood problem with a simple geometry. The simplicity of this centering problem is probably the reason why this has been the first successful approach for path-following methods for linear programming, quadratically constrained quadratic programming and convex programming.

A similar reduction of the centralization problem to the problem of finding an analytic center has been indicated for the problem in primal formulation, with the introduction of slack variables for the constraints $c^Tx \leq K$. The conclusions for both formulations must be equivalent, although the dual formulation gives a better geometrical insight.

Essentially the same results as we obtained for the primal-dual formulation can now be replicated for the primal formulation: values of β near 1 produce short steps, with one internal iteration per centralization and a complexity of $0(\sqrt{n}\, L)$ NR iterations. This is the result obtained by Renegar [95], the first algorithm for linear programming with this low complelxity bound on the number of iterations. Updating formulae $\beta=1/(1+0(1/\sqrt{n}))$ were used by Hertog, Roos and Terlaky [59] for the dual problem. They obtained "not-so-short" steps with the same complexity and properties as for the primal-dual approach.

4.8 PRIMAL-DUAL METHODS

Primal-dual properties of the central path were first studied by

Megiddo [79], and motivated a primal-dual algorithm by Kojima, Mizuno and Yoshise [65]. The first version of their algorithm did not reduce the complexity obtained by Karmarkar, but improved variants developed by the same authors Kojima *et al.* [64] and also independently by Monteiro and Adler [85] achieved the bound of $O(\sqrt{n}\,L)$ iterations and $O(n^3L)$ arithmetical operations.

We consider a pair of primal-dual and linear programming problem as follows:

$\underset{x}{minimize}\ z^T x$ $\qquad\qquad$ $\underset{z}{minimize}\ x^T z$

subject to constraints $Ax=b$ $\qquad\qquad$ subject to constraints $P_A z=P_A c$

$x \geq 0$ $\qquad\qquad$ $z \geq 0.$

In the primal problem, $z \in Z^o$ $\{z\in\mathbb{R}^n \mid P_A z=P_A c,\ z > 0\}$ is a fixed dual feasible slack and in the dual problem $x \in S^o$ is a primal feasible solution.

The primal-dual penalized function is defined for $\alpha \geq 0$ as

$$x \in S^o,\ z \in Z^o \rightarrow f_\alpha(x,z) = \alpha x^T z + p(x) + p(z)$$

$$= \alpha x^T z - \sum_{i=1}^{n} \log x_i z_i$$

4.8.1 Scaling

Given a positive diagonal matrix D, the scaled problems are obtained by using the scaling operations as follows

$$x = D\bar{x},\ z = D^{-1}\bar{z}. \qquad \text{(Lemma 4.3.2)}$$

Thus the above primal-dual linear programming problem reduces to the following pair:

$\underset{\bar{x}}{minimize}\ \bar{z}^T\bar{x}$ $\qquad\qquad$ $\underset{\bar{z}}{minimize}\ \bar{x}^T\bar{z}$

subject to $AD\bar{x}=b$ $\qquad\qquad$ subject to $P_{AD}\bar{z}=P_{AD}Dc$

$\bar{x} \geq 0$ $\bar{z} \geq 0.$

By substitution, we get

$$f_\alpha(x,z) = f_\alpha(D\bar{x}, D^{-1}\bar{z}) = f_\alpha(\bar{x},\bar{z})$$

4.8.2 Primal-dual Central Points

The primal-dual central points are the minimizers of $f_\alpha(x,z)$ for $x \in S^\circ$ and $z \in Z^\circ$.

For each $\alpha>0$, and a primal central point $x(\alpha)$, we have defined in lemma 4.4.8, a dual slack $z(\alpha)$ as $z(\alpha)=x^{-1}(\alpha)/\alpha$. Now we shall see that the primal-dual central points are consistent with these definitions.

Lemma 4.8.1

For $\alpha \in \mathbb{R}_{++}$ and an interior feasible pair $(x,z) \in \mathbb{R}^{2n}$, the following statements are equivalent:

(i) (x,z) form a primal-dual central point associated to α.

(ii) $\nabla_x f_\alpha(x,z)=0$ and $\nabla_z f_\alpha(x,z)=0$

(iii) $\alpha XZ = 1$

(iv) $x=x(\alpha)$ and $z=z(\alpha)$

Proof:

We assume that $(\bar{x},\bar{z})$ is central.

Fixing $\bar{z}$, $f_\alpha(\bar{x},\bar{z}) = \min_{x \in X^\circ} f_\alpha(x,\bar{z})$, where $X^\circ = \mathrm{diag}((x_1^o,\ldots,x_n^o)$

Hence $\bar{x} = x(\alpha)$. Computing gradients, we obtain for any $z \in Z^\circ$,

$$\nabla_x f_\alpha(\bar{x}, z) = \alpha z - \bar{x}^{-1}, \nabla_z f_\alpha(\bar{x}, z) = \alpha \bar{x} - z^{-1}$$

Substituting $z=z(\alpha) = \bar{x}^{-1}/\alpha$ in these expressions

$$\nabla_x f_\alpha(\bar{x}, z) = \nabla_z f_\alpha(\bar{x}, z) = 0$$

Since $f_\alpha(\cdot)$ is strictly convex, this ensures that z is the unique minimizer of $f_\alpha(\bar{x}, .)$, and thus $\bar{z} = z = z(\alpha)$. This proves the equivalence of (i), (ii) and (iv). The equivalence of (ii) and (iii) is immediate from the expressions for the gradients, completing the proof.

4.8.3 Nearly Central Point

Conditions (iii) in the Lemma above provides a simple criterion for measuring centrality as $\delta(x,z,\alpha)=\|\alpha X_x e\|$.

Now if $\delta(x,z,\alpha)=0$

$<=> \|\alpha Xz - e\| = 0$

$<=> \alpha Xz-e = 0$

$<=> \alpha x_i z_i-1=0$

$<=> x_i z_i = 1/\alpha$

for i=1,2,...,n

Thus the complementary slacks are balanced. Otherwise, $\delta(x,z,\alpha)$ measures the unbalance of the slacks. When $\alpha->\infty$, $x_i z_i->0$ along the central path. In view of Lemma 4.5.3 the above discussed proximity criterion can be related to the primal proximity $\delta(x,\alpha)$ as follows:

$$\delta(x,\alpha) = \delta(x, z(x,\alpha), \alpha)$$

$$= \min \{\delta(x,z,\alpha) | A^T w+z=c, w\in\mathbb{R}^m\}$$

where $z(x,\alpha)$ is the dual slack defined in (4.12). Thus $\delta(x,\alpha) \le \delta(x,z,\alpha)$. Besides this, if $\delta(x,z,\alpha) < 1$, then a better dual slack is provided by $z(x,\alpha)$ in the sense of proximity.

4.8.4 Centering

We consider $\alpha > 0$ and an interior pair $(x,z) \in S^0 \times Z^0$. Finding $(x(\alpha),z(\alpha))$ from (x,z) can be seen either as one problem in $\mathbb{R}^{2n}$ or as two separate n-dimensional problems.

The feasible set

$$SXZ = \{(x,z) \in \mathbb{R}^{2n}: Ax=b; A^T w+z = c; (x,z) \geq 0\}$$

for the 2n-dimensional problem seems to be quite interesting. In order to obtain $(x(\alpha), z(\alpha))$ from(x,z), we define steepest descent direction (h_x,h_z) for $f_\alpha(.,.)$ at (x,z) as follows:

$$h_x = -P_A \nabla_x f_\alpha(x,z) \text{ and } h_z = [(I - P_A)\nabla_z f_\alpha(x,z)], \qquad (4.15)$$

where I is an identity matrix of appropriate order.

Now each feasible steepest descent direction (h_x,h_z) originated at (x,z) is such that

$h_x \in N(A)$ and $h_z \in N(P_A) = \Re(A^T) \perp N(A)$.

We now discuss the following options for descent directions.

(i) **Independent Scaling**

The first option is to obtain independent SSD directions for functions $f_\alpha(x,.)$ and $f_\alpha(.,z)$ as

$$h_x = -XP_{AX}X\nabla_x f_\alpha(x,z) = XP_{AX}(\alpha Xz - e),$$

and $$h_z = -Z\tilde{P}_{AZ^{-1}} Z\nabla_z f_\alpha(x,z) = -Z\tilde{P}_{AZ^{-1}}(\alpha Zx - e) \qquad (4.16)$$

Here h_x is obtained by scaling the primal problem about x, and h_z by scaling the dual problem about z.

Although these may be the best possible direction (they are the Newton directions for each problem separately), using them for

centering with a *fixed* α is not reasonable for two reasons: first because projections into two different spaces must be computed, which doubles the work per iteration in comparison to primal only centralization. The second reason is that the primal and dual centralization problems are totally uncoupled in the sense that the sequence of primal solution $\langle x^k \rangle$ generated by SSD method with any reasonable line search (fixed steplength, fixed percentage to the boundary or one-dimensional minimization) will be totally independent of the sequence $\langle z^k \rangle$ of dual variables. This is so because variations in z affect $f_\alpha(.,z)$ by a constant value. Also h_x in (4.16) is not affected by z since $P_{AX} X_z = P_{AX} X_e$ by lemma 4.3.1. If (x^k) approaches the central point $x(\alpha)$, then a good approximation $z(x^k,\alpha)$ to $z(\alpha)$ is available from (4.12) and the dual iterations are of no use. This may not be true in potential reduction methods or if α is allowed to change during the line searches.

(ii) **Joint Scaling**

Duality is preserved if the problems are jointly scaled by a single positive diagonal matrix D, with $x=D\bar{x}$ and $z=D^{-1}\bar{z}$. The steepest descent directions at the point (x,z) in this case will be given as

$$h_x = -DP_{AD}D\nabla_x f_\alpha(x,z) = -DP_{AD}D(\alpha z - x^{-1})$$

$$h_z = -D^{-1}\tilde{P}_{AD}D^{-1}\nabla_z f_\alpha(x,z) = -D^{-1}\tilde{P}_{AD}D^{-1}(\alpha x - z^{-1}).$$

(iii) **Primal Scaling**

Using D=X, the point $(x,z) \in \mathbb{R}^{2n}$ is mapped in (e,Xz). The primal direction coincides with $h(x,\alpha)$ which is Newton Raphson direction for the primal problem. In this case, the same projection operation is applied to two different vectors which reduces the work significantly as compared to independent scaling.

(iv) **Primal-Dual Scaling**

Nice results are obtained by the following scaling, known as primal-dual scaling. Let $D=X^{1/2}Z^{1/2}$, where $X^{1/2}=\text{diag}(x_1^{1/2},\ldots,x_n^{1/2})$ and

$Z^{-1/2} = diag(z_1^{-1/2}\ldots,z_n^{-1/2})$ are diagonal matrices. Now, x and z are mapped into $D^{-1}x = Dz = (ZX)^{1/2}e$. The SSD directions in this case are given as

$$h_x = -DP_{AD}(\alpha(XZ)^{-1/2}e - (XZ)^{-1/2}e)$$

and
$$h_z = -D^{-1}\tilde{P}_{AD}(\alpha(XZ)^{1/2}e - (XZ)^{-1/2}e) \qquad (4.17)$$

respectively.

It is important to note that only one projection is required in this case, since the primal and dual scaled directions are, respectively, projection and orthogonal complement of the same vector. These are the Newton direction for the optimality condition $\alpha XZe - e = 0$.

These results have been used in Kojima, Mizuno, and Yoshise [65], Monteiro and Adler [85] and in most of the studies in primal-dual penalty and potential reduction methods.

with $\alpha \to \infty$, we obtain the primal-dual affine-scaling directions

$$h_x = -DP_{AD}(XZ)^{1/2}e,\ h_z = -D^{-1}\tilde{P}_{AD}(XZ)^{1/2}e.$$

Monteiro, Adler and Resende [87] obtained a polynomial algorithm by taking very short steps along these directions. To the best of our knowledge presently , this is the only polynomial-time interior point method that is not based on any auxiliary function.

4.8.5 Some New Convergence results for Primal-Dual interior point path-following algorithms

We rewrite our usual primal and dual problems (P) and (D) as follows:

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

$$\text{subject to } x \in S=\{x \in \mathbb{R}^n : Ax=b, x>0\} \qquad \textbf{(P)}$$

$$\text{maximize } b^T w$$

$$\text{Subject to } (w,z) \in H=\{(w,z) \in \mathbb{R}^{m+n} : A^T w+z=c, z \geq 0\} \quad (D)$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and we assume that the matrix A has full rank.

Now, in order to solve (P) we consider the problem - (P_μ)

$$\text{minimize } c^T x - \mu \sum_{j=1}^{n} \log x_j \qquad (P_\mu)$$

subject to $x \in S^o$, the relative interior of S;

where μ is a penalty parameter.

Then each sub-problem (P_μ) has a unique global minimum solution x for which Karush-Kuhn-Tucker conditions are:

$$c - \mu X^{-1} e + A^T w = 0$$

$$Ax - b = 0,\ x \geq 0 \qquad \text{where } X = \text{diag}(x_1,\ldots,x_n)$$

$$\text{and } e = (1,\ldots,1)^T.$$

On introducing slack variable $z=\mu X^{-1} e$, the above stationary conditions

can be restated as

$$\bar{Z}x - \mu e = 0 \quad i.e., \quad x_i z_i = \mu, \ (i=1,2,\ldots,n)$$

$$Ax - b = 0$$

$$A^T w + z - c = 0 \qquad (4.18)$$

where $\bar{Z} = diag(z_1, z_2, \ldots, z_n)$

It is important to note that all products of complementarity pair of variables x_i and z_i take a common value μ. We fix two control parameters σ and τ such that $0 < \tau < \sigma < 1$ to control penalty parameter μ and steplength respectively. At the kth iteration we first set the value of the parameter μ as $\mu = \sigma \left(\sum_{i=1}^{n} x_i z_i \right) / n$ in (4.18) and compute Newton's directions $(\Delta x^k, \Delta w^k, \Delta z^k)$ at (x^k, w^k, z^k) as the solution of the system of linear equations

$$F_x \Delta x^k + F_w \Delta w^k + F_z \Delta z^k = F(x^k, w^k, z^k), \qquad (4.19)$$

where $F(x,w,z)$ denote left-hand side of the system of equations (4.18) and F_x, F_w, F_z denote the Jacobian matrices of the mapping $F(x,w,z)$ at (x^k, w^k, z^k) with respect to x, w and z respectively. On further simplification the system of linear equation (4.19) reduces to

$$\bar{Z}\Delta x^k + X\Delta z^k = y(\mu)$$

$$A\Delta x^k = 0 \qquad (4.20)$$

$$A^T \Delta w^k + \Delta z^k = 0,$$

where $y(\mu) = (x_1^k z_1^k - \mu, x_2^k z_2^k - \mu, \ldots, x_n^k - z_n^k - \mu) \in \mathbb{R}^n$.

After obtaining Newton's direction $(\Delta x^k, \Delta w^k, \Delta z^k)$, as the solution of the system of equations (4.20), we obtain the next iterate as follows:

$$x^{k+1} = x^k - \alpha^k \Delta x^k$$

$$w^{k+1} = w^k - \alpha^k \Delta w^k$$

$$z^{k+1} = z^k - \alpha^k \Delta z^k$$

where $0 < \alpha^k \leq 1$.

Iterations continue until duality gap $|c^T x^k - b^T w^k|$ becomes sufficiently small.

Deviation Parameter and Convergence

Given an initial point $(x^0, w^0, z^0) \in S \times H$; the algorithm generate a sequence $\{(x^k, w^k, z^k)\}$ of points in $S^0 \times H^0$, where S^0 and H^0 denote interior of the sets S and H respectively.

Let G denote the curve consisting of the solutions of the system (4.18) i.e., $G = \{x(\mu), w(\mu), z(\mu); \mu > 0\}$.

As penalty parameter $\mu \to 0$, the curve G leads to a pair of optimal solution $\hat{x} \in \mathbb{R}^n$ of (P) and $(\hat{w}, \hat{z}) \in \mathbb{R}^{m+n}$ of (D). Now in order to measure the deviation of any iterate $(x^k, w^k, z^k) \in \mathbb{R}^{m+n}$ from the curve G, we introduce a quantity δ_k (deviation parameter) as follows:

$$\delta_k = \frac{\left(\sum_{i=1}^{n} x_i z_i \right) / n}{\underset{1 \leq i \leq k}{\text{minimum}} \{ x_i^k z_i^k \}}$$

Obviously $\delta_k \geq 1$. The iterate $((x^k, w^k, z^k))$ lies on the curve if $\delta_k = 1$.

Theorem 4.8.1

An iterate (x^k, w^k, z^k) generated by primal-dual interior algorithm lies on the curve G, all the product $x_i z_i$ of the complimentarity pair of variables x_i and z_i (i=1,2,...,n) are equal.

Proof:

Let $x_t^k z_t^k = \min \{x_i^k z_i^k : i = 1,2,...,n\}$. Now if kth iterate $(x^k,w^k,z^k) \in S^0 \times H^0$ lies on G, we have

$$\frac{\left(\sum_{i=1}^{n} x_i^k z_i^k\right)/n}{\underset{1 \le i \le k}{minimum} \{x_i^k, z_i^k\}} = 1$$

i.e., $$\frac{\left(\sum_{i=1}^{n} x_i^k z_i^k\right)/n}{x_t^k, z_t^k} \le 1$$

i.e., $$\sum_{i=1}^{n} x_i^k z_i^k \le n\, x_t^k z_t^k$$

i.e., $$\sum_{i=1}^{n} x_i^k z_i^k - n\, x_t^k z_t^k \le 0$$

i.e., $$\sum_{i=1}^{n} (x_i^k z_i^k - x_t^k z_t^k) \le 0 \qquad (4.21)$$

Since $x_t^k z_t^k \le x_i^k z_i^k =$; i=1,2,...,n.

Hence it follows that $x_i^k z_i^k - x_t^k z_t^k \ge 0$.

Now equality in (4.21) will hold only if each term on the left hand side of it are separately zero.

i.e., $(x_1^k z_1^k - x_t^k, z_t^k) = 0$, $(x_2^k z_2^k - z_t^k z_t^k) = 0$, ..., $(x_n^k z_n^k - x_t^k z_t^k) = 0$,

which in turn implies

$$x_1^k z_1^k = x_2^k, z_2^k = \ldots = x_n^k z_n^k = x_t^k z_t^k.$$

Hence proved.

Now, we investigate the dependence of the convergence of the algorithm on the deviation paramter δ_k. Kojima *et al* [66] has introduced two control parameters τ and σ such that $0 \le \tau \le \sigma \le 1$ which control the penalty parameter μ and steplength α^k. In particular if we choose $\sigma=1/2$ and $\tau=1/4$. It follows from Kojima *et al* [66, p. 41, cor. 2.2] that $\alpha^k \ge 4/n\delta_k$ and

$$\frac{c^T x^{k+1} - b^T w^{k+1}}{c^T x^k - b^T w^k} \le 1 - \frac{2}{n\delta_k} \qquad (4.22)$$

where n denote the number of variables in primal problem.

Theorem 4.8.2

If primal-dual interior point algorithm [66] is applied to a given linear programming problem, the smaller the value of deviation parameter δ_k the faster the convergencne.

Proof

Putting k=0,1,...,k in equality (4.22) we get (k+1) inequalities. On multiplying these (k+1) inequalities, we get

$$\frac{c^t x^{k+1} - b^T w^{k+1}}{c^T x^o - b^T w^o} \le \left(1 - \frac{2}{n\delta_o}\right)\left(1 - \frac{2}{n\delta_1}\right) \cdots \left(1 - \frac{2}{n\delta_k}\right) \qquad (4.23)$$

If $\left(1 - \frac{2}{n\delta_t}\right)$ denote the maximum of all (k+1) factors in the right hand side of inequality (4.23), i.e.,

$$\left(1 - \frac{2}{n\delta_t}\right) = \underset{0 \le i \le n}{maximum}\left\{1 - \frac{2}{n\delta_i}\right\}$$

Now, on replacing all (k+1) factors in the right hand side of inequality

(4.23) by $\left(1 - \frac{2}{n\delta_t}\right)$ we get

$$\frac{c^t x^{k+1} - b^T w^{k+1}}{c^T x^0 - b^T w^0} \leq \left(1 - \frac{2}{n\delta_t}\right)^{k+1} \qquad (4.24)$$

Now, it follows from the inequality (4.24) that the duality gap will conv to zero as fast as the sequence $\left(1 - \frac{2}{n\delta_t}\right), \left(1 - \frac{2}{n\delta_t}\right)$ will converg zero. When n remains fixed, smaller the value of deviation paramet faster the convergence of the sequence $\left(1 - \frac{2}{n\delta_t}\right), \left(1 - \frac{2}{n\delta_t}\right)$. He it follows that the smaller the value of the deviation parameter δ_t ir algorithm the faster the convergence. When δ_t attains its minimum v i.e., when $\delta_t = 1$, the inequality (4.24) becomes

$$\frac{c^T x^{k+1} - b^T w^{k+1}}{c^T x^0 - b^T w^0} \leq \left(1 - \frac{2}{n}\right)^{k+1} \qquad (4.25)$$

which governs convergence of the algorithm when iterates lie or curve G. It also follows from (4.25) that as n increases algorithm exh slow convergence.

Control parameter σ and the convergence

At the beginning of the kth iteration we set $\mu = \sigma\left(\sum_{i=1}^{n} x_i z_i\right.$ where $0 \leq \tau < \sigma < 1$. Here we investigate the dependence of the converg

of the primal-dual interior point algorithm on the control parameter σ.

Theorem 4.8.3

If primal-dual interior point algorithm is applied to solve a linear programming problem, smaller the value of the control parameter σ faster the convergence.

Proof:

We know that $0 < \alpha^k \leq 1$.

Case 1: When $\alpha^k < 1$.

When $\alpha^k < 1$, then it follows from theorem (2.1) of Kojima *et al.* [66, p. 38] that

$$\frac{c^T x^{k+1} - b^T w^{k+1}}{c^T x^k - b^T w^k} = (1 - (1 - \sigma)\alpha^k) \qquad (4.26)$$

Let $1-\sigma=\phi$. Since $0<\sigma<1$, so $0 < \phi < 1$. Also $\alpha^k < 1$, so $\phi\alpha^k<1$, which in turn implies that $(1-\phi\alpha^k) < 1$. Thus equation (4.26) reduces to

$$\frac{c^T x^{k+1} - b^T w^{k+1}}{c^T x^0 - b^T w^k} = (1 - \phi\,\alpha^k)$$

Putting k=0,1,2,..., k-1, we get k equations. On multiplying these k equations, we get

$$\frac{c^T x^k - b^T w^k}{c^T x^0 - b^T w^0} = \prod_{i=0}^{k-1} (1 - \phi\,\alpha^k) \qquad (4.27)$$

Let α^t = minimum $\{\alpha^0, \alpha^1, \ldots, \alpha^{k-1}\}$, maximum $\{(1-\phi\alpha)\}=(1-\phi\,\alpha^t)$. Now replacing each term in the product $\prod_{i=0}^{k-1} (1-\phi\,\alpha^t)$ on the right hand side of (4.27) by $(1-\phi\,\alpha^t)$ we get

$$\frac{c^T x^k - b^T w^k}{c^T x^0 - b^T w^0} \leq (1 - \phi\,\alpha^t)^k \qquad (4.28)$$

Now, it follows from inequality (4.28) that the duality gap will converge to zero as fast as the sequence $(1-\phi\alpha^t),(1-\phi\alpha^t)^2,\ldots$ will converge to zero. But convergence of this sequence will depend on the value of the

expression$(1-\phi\alpha^t)$. In fact, smaller value of expression$(1-\phi\alpha^t)$ will ensure faster convergence. Since α^t=minimum $\{\alpha^o, \alpha^1,...,\alpha^{k-1}\}$, so the maximum value of ϕ can ensure faster convergence of the sequence $(1-\phi\alpha^t),(1-\phi\alpha^t)^2,...$. But ϕ will be maximum only if σ will take minimum value in the interval $0 \le \tau < \sigma$. Hence it follows that minimum the value of control parameter σ faster the convergence of the algorithm.

Case II: when $\alpha^k=1$, it follows from Kojima *et al* [66, p. 38]

$$\frac{c^T x^{k+1} - b^T w^{k+1}}{c^T x^o - b^T w^k} = \sigma$$

Putting k=0,1,...,k-1, we get k equations. On multiplying these k equations we get

$$\frac{c^T x^{k+1} - b^T w^{k+1}}{c^T x^o - b^T w^o} = \sigma^k \qquad (4.29)$$

Now, it follows from (4.29) that faster convergence of duality gap to zero can be achieved if we take smaller value of the control parameter σ in the interval $0 \le \tau < \sigma < 1$.

4.8.6 Newton-Raphson Direction

The optimality conditions for the centering problem can be stated in several different equivalent ways. As observed in Ye [130], each different way of writing the condition will generate a different set of Newton directions i.e., corresponding to three different ways of writing the optimality conditions as:

$$\begin{array}{lll} \alpha XZe-e = 0 & \alpha Ze-X^{-1}e=0 & \alpha Xe-Z^{-1}e=0 \\ Ax=b, & Ax=b \quad \text{and} & Ax=b \\ A^Tw+z=c & A^Tw+z=c & A^Tw+z=c \end{array} \qquad (4.30)$$

will generate three sets of Newton directions.

Although, the systems above are equivalent, their solutions by Newton's method behave differently. The last two equations in each system simply mean that the resulting directions h_x and h_z must be, respectively, primal and dual feasible.

(i) Primal-Dual Scaling

The Newton directions for the first system in (4.30) are given by

$$Z h_x + X h_z = -XZ_e + e/\alpha$$

$$Ah_x = 0$$

$$h_z = A^T y$$

It is easy to see that the SSD directions h_x and h_z given by (4.17), can satisfy the above, system of equations and thus h_x and h_z are the solutions of the above system. This system can also be solved by a change of scale and projection as we shall do in the next case.

(ii) Primal Scaling

The Newton equations for the second system in (4.30) are

$$\alpha h_z + X^{-2} h_x = -\alpha z + x^{-1}$$

$$Ah_x = 0$$

$$h_z = A^T y$$

The system is solved by scaling i.e., by setting $x = X\bar{x}$, $z = X^{-1}\bar{z}$ and simplifying,

$$\alpha \bar{h}_z + \bar{h}_x = -\alpha \bar{z} + e,$$
$$\bar{h}_x \in N(AX),$$
$$\bar{h}_z \perp N(AX).$$

Computing now the projection in N(AX) and its orthogonal complement

with

$$P_{AX}\bar{h}_x = \bar{h}_x \text{ and } \tilde{P}_{AX}\bar{h}_z = \bar{h}_z$$

$$\bar{h}_x = -P_{AX}(\alpha\bar{z} - e),\ \alpha\bar{h}_z = -\tilde{P}_{AX}(\alpha\bar{z} - e)$$

or finally,

$$h_x = -XP_{AX}(\alpha X_z - e),\ \alpha h_z = -X^{-1}\tilde{P}_{AX}(\alpha X_z - e)$$

The direction h_x coincides with the primal SSD dirction in (4.16), whereas h_z does not coincide with the corresponding direction in (4.15).

(iii) Dual Scaling

A similar treatment applied to the third system in (4.30) leads to

$$\alpha h_x = -Z^{-1}P_{AZ^{-1}}(\alpha Zx - e),$$
$$h_z = -Z\tilde{P}_{AZ^{-1}}(\alpha Z_x - e)$$

Here, h_z coincides with the dual SSD direction in (4.17). These directions have been classified in Ye [130]. To the best of our knowledge no comparative study of these three sets of Newton's directions is available so far. Primal-dual scaling was implemented by McShane, Monma and Shanno [76], who reported excellent results.

The Path-Following Algorithms

ALGORITHM 4.8.2

(Implementable primal-dual path-following algorithm):

Given $\epsilon \in (0,0.5)$, $\beta \in (0,1)$, an admissible initial pararneter $\alpha_o, \in \mathbb{R}$; $x^o \in S^o$ and $z^o \in Z^o$ such that $\delta(x^o, z^o, \alpha_o) < \epsilon$

k:=0

REPEAT

α_{K+1}:=α_K/β

Call the SSD or Newton-Raphson algorithm to find a feasible pair (x^{k+1}, z^{k+1}) such that $\delta(x^{k+1}, z^{k+1}, \alpha_{k+1}) < \epsilon$

k:=k+1

UNTIL $z^{k\,T} x^k < 2^{-L}$.

Here the internal algorithm can use any of the directions described above. If primal-dual scaling is used, then short steps lead to a complexity 0(√n L) internal iterations.

4.8.7 The Proximity to the Central Path

We defined two measure of proximity from an interior point (x,z) to a *central point* (x(α), z(α)), namely δ(x,α) and δ(x,z,α), associated, respectively, to the primal and primal-dual central paths. The proximity to the *central path* can be defined in both cases as

$$\tilde{\delta}(x) = \min_{\alpha} \tilde{\delta}(x,z)$$
$$\tilde{\delta}(x,z) = \min_{\alpha} \delta(x,z,\alpha).$$

The solution of each of these problems defines a parameter value α such that x(α) or (x(α), z(α)) is the nearest central point in relation to x or (x,z) respectively.

The computation of $\tilde{\delta}(x)$ was made in Gonzaga [40] giving

$$\alpha = \frac{c_p^{\,T} e_p}{\| c_p \|^2},$$

where $c_p = P_{AX} X_c$ and $e_p = P_{AX} e$,

The scaled SSD direction corresponding to this value of α is interesting: it is the centering direction used by Barnes, Jensen, and Chopra [10], the SSD direction for the centering problem restricted to a constant-cost slice.

The primal-dual proximity measure $\tilde{\delta}(x,z)$ was used by Kojima, Mizuno and Yoshise [65] and by Monteiro and Adler [85].

The computation of primal-dual proximity measure $\tilde{\delta}(x,z)$ is straightforward and can be obtained using the following formula:

$$\tilde{\delta}(x,z)^2 = \min_{\alpha} \| \alpha ZXe - e \|^2 .$$

Differentiating with respect to α and equating to zero, we obtain

$$1/\alpha = x^T z/n.$$

The above result is used in algorithms as follows: at each path-following iteration, instead of setting $\alpha_{k+1} = \alpha_k/\beta$ as in Table 1, we set

$$\alpha_{k+1} = \frac{1}{\beta} \frac{n}{x^{k\,T} z^k}.$$

A non-path-following algorithm, (possibly not polynomial) can be obtained by using this rule before all SSD iterations, instead of keeping α constant until a nearly central point is obtained.

4.9 POTENTIAL REDUCTION METHODS

Potential reduction algorithms work either with the primal potential fucntion defined for lower bounds $v \leq \hat{v}$ and for a constant $q \leq n$:

$$x \in S^o \rightarrow f_v(x)=q \log(c^T x-v)+p(x) \qquad (4.31)$$

or with the primal-dual potential function

$$x \in S^o,\ x \in Z^o \rightarrow F_q(x,z)=q \log x^T z+p(x)+p(z) \qquad (4.32)$$

Potential-reduction methods can be classified in the following three categories.

(i) Primal Methods, based on potential function defined in (4.31).

(ii) Primal Methods that use potential function defined in [4.32] in the convergence analysis or in updating lower bounds, but keep no memory of dual variables, and

(iii) Primal-Dual Methods.

The first primal potential reduction method is Karmarkar's algorithm. With Todd and Burnell's lower bound updates it is a non-path-following algorithm. It is based on projective geometry and was the last one to have the complexity reduced to $0(\sqrt{n}\ L)$ iterations.

Primal affine polynomial methods were first described in Gonzaga [47] with the following result: the SSD algorithm applied to the potential function with $q \leq n + \sqrt{n}$, $q=0(n)$, and $v = \hat{v}$ solves the linear programming problem with the same complexity as Karmarkar's algorithm. Ye [129] showed how to use approach (ii) to obtain methods with a complexity of $0(\sqrt{n}L)$ iterations. Although the first version of his method uses short steps, it can be extended to "not-so-short" and large steps with conclusions similar to what we obtained for penalty function methods. Further developments of Ye's approach can be found in Freund [25], Anstriecher and Bosch [8] and also in Ye [130]. Primal only methods were extended to large steps in Gonzaga [44]. Ye's method is a path-following method in our definition, since it only updates the lower bounds (or dual variables) at nearly central points. But his analysis opens the way

to real primal-dual methods that work directly with the primal-dual potential function, keeping the low complexity without restoring to the central path.

4.9.1 Primal only Methods

Algorithm 4.7.1 will be specialized here for the potential function. There will be a change in notation: the iteration counter k will be increased at each internal iteration, and not when a nearly central point is reached. The following algorithms is Algorithm 4.7.1 in a slightly different presentation.

ALGORITHM 4.9.1

(Potential Reduction Path-Following Algorithm):

Given $\epsilon \in (0,0.5)$, $\beta \in (0,1)$, $\nu < \hat{\nu}$ and $x \in S^o$.

k:=0

REPEAT

Direction: Compute

$$h(x^k, v_k) = -P_{AX_k}\left(\frac{q}{c^T x^k - v_k} X_k c - e\right)$$

$$h(x^k, v_k) = X_k \bar{h}(x^k, v^k),$$

$$\delta(x^k, v_k) = \| \bar{h}(x^k, v_k) \|.$$

Line search: $x^{k+1} = x^k + \lambda h(x^k, v_k)$

Update: If $\delta(x^k, v_k) \leq \epsilon$ then

$v_{k+1} = \beta v_k + (1-\beta) c^T x^k$

else $v_{k+1} := v_k$

k:=k+1

UNTIL $c^T x^k - v_k < 2^{-L}$.

Let $v_k = v$ be the lower bound at an iteration in which the condition $\delta(x,v) < \epsilon$ is satisfied. Then the lower bound update will be given by

$$(c^T x - v') = \beta(c^T x - v).$$

Here β is not free, because $v' < \hat{v}$ is needed. A solution to this is provided by Lemma 4.5.1:

We make sure that

$$\beta(c^T x - v') \geq \Delta(x,v)$$

Using (4.13) with

$$e^T h \leq \sqrt{n}\,\|\bar{h}\| = \sqrt{n}\,\epsilon$$

$$\Delta(x,v) \leq \frac{n+\epsilon\sqrt{n}}{q}(c^T x - v).$$

The condition above is ensured by setting

$$\beta \geq \frac{n+\epsilon\sqrt{n}}{q}$$

The Choice of q

Gonzaga [44] suggest $q > 2n$. The reason for this choice lies in the condition number of $\nabla^2 f_v(x)$ near a central point $x(v)=e$ (after scaling). If this number approaches 1, SSD approaches to the behaviour of Newton's method, with the following consequences:(i) from the practical point of view, centering should be more efficient; (ii) from the theoretical point of view, the good conditioning allows the extension of the conclusions in Lemma 4.5.4 about proximity. Choosing q and β so that $\beta q \geq 2n$, the condition number of the Hessian near a central point will be better than 2, and the proximity becomes well defined. Now, for well chosen values of ϵ, the path-following algorithm achieves a complexity of $O(\sqrt{n}L)$ iterations whenever $1/\beta = 1+O(1/\sqrt{n})$, as we had in the penalty function

methods.

Large steps with complelxity of 0(nL) iterations are obtained by updating v at all iterations (not necessarily near central points), as we did in Karmakar's algorithm:

If $\tilde{v}(x^k) > v_k$ then set $v_{k+1} := \tilde{v}(x^k)$

else set $v_{k+1} := v_k$.

4.9.2 Ye's Approach

The primal-dual potential function was first studied by Todd and Ye [121] and an 0(n√L)-iteration algorithm was developed in Ye [129]. The function is described in the lemma 4.9.2 below.

Lemma 4.9.2

We consider the primal-dual potential function discussed in (4.32) with q=n and let $(x,z) \in S^0 \times Z^0$ be an interior feasible primal-dual pair then $F_n(x,z) \geq n\log n$ and equality occurs if and only if (x,z) is central.

Proof:

Let (x,z) be an interior feasible pair.

$$F_x(x,z) = n \log x^T z - \sum_{i=1}^{n} \log x_i z_i$$

$$= n \log \frac{x^T z}{\left(\prod_{i=1}^{n} x_i z_i\right)^{1/n}}$$

$$= n \log n + n \log \frac{\left(\sum_{i=1}^{n} x_i z_i\right)/n}{\left(\prod_{i=1}^{n} x_i z_i\right)^{1/n}}.$$

The fraction is the relation between the arithmetic and geometric means

of the positive numbers x_iz_i. This is greater than or equal to 1 with equality occuring if and only if for some $\theta>0$, $x_iz_i = \theta$, i=1,2,...,n. But this is precisely the centrality condition completing the proof.

Now, we consider a sequence of primal-dual pairs (x^k,z^k). We shall study sequences with the following characteristics for q>n:

(i) $F_q(x^o,z^o)$

(ii) For k=0,1,2,..., $F_q(x^{k+1},z^{k+1}) \leq F_q(x^k,z^k) - \theta$ where $\theta=0(1)$ is a constant.

Lemma 4.9.3

In a sequence (x^k,z^k) with the characteristics above, the condition $x^{kT}z^k \leq 2^{-L}$ is reached for k=0((q-n)L).

Proof:

For a sequence (x^k,z^k) with these characteristics and k>0

$F_q(x^k,z^{k+1}) \leq F_q(x^o,z^o) - k\theta$ using the definition of the potential function,

$(q-n)\log x^{kT}z^k + F_n(x^k,z^k) \leq F_q(x^o,z^o) - k\theta$.

From lemma 4.9.2, $F_n(x^k,z^k) \geq n\log n > 0$,

and hence $(q-n)\log x^{kT}z^k \leq F_x(x^o,z^o) - k\theta$

The condition $\log x^{kT}z^k \leq -L$ is ensured for $F_q(x^o,z^o) - k\theta \leq -(q-n)L$

or $k\theta \geq F_q(x^o,z^o)+(q-n)L=0((q-n)L)$ completing the proof.

We conclude that any algorithm that generates sequences with the characteristics above reaches the stopping condition $x^Tz \leq 2^{-L}$ in 0(q-n)L iterations. If $q=n+0(\sqrt{n})$, then the algorithm complexity will be of $0(\sqrt{n}L)$

iterations.

The condition (i) is not strong. It is satisfied for instance if x^o is nearly central for a penalty parameter $\alpha=0(\sqrt{n}2^{-L})$. An algorithm that allows $f(x^o,z^o)=0(2^L)$ and still has $0(\sqrt{n}L)$ - iteration complexity was reported in Mizuno [83].

Algorithm

It is important to note that the first version of Ye's Algorithm [129] is identical with the Algorithm 4.9.1 above, with the following updating rule:

Update: if $\delta(x^k,v_k) \leq \epsilon$ then

$v_{k+1}:=v(x^k,v_k)$ by (4.12)

else $v_{k+1}:=v_k$

The gap reduction is

$$c^T x^k - v_{k+1} = \Delta(x^k,v_k) = \frac{n-e^{\bar{Th}}(x^k,v^k)}{q}(c^T x^k - v_k),$$

similar to Agiroithm 4.9.1.

A second version of Ye's Algorithm is equivalent to the first one, but uses dual slack explicitly.

ALGORITHM 4.9.4 (Potential Reduction Path-Following Algorithm) Ye [129]:

Given $\epsilon \in (0,0.4)$, $x^0 \in S^0$, $z^0 \in Z^0$

k:=0

REPEAT

Direction: compute

$$\bar{h} := -P_{AX_k}\left(\frac{q}{z^{k^T}x^k} X_k z^k - e\right),$$

$$h := X_k \bar{h}$$
$$\delta := \| \bar{h} \|$$

Line search: $x^{k+1}:=x^k + \lambda h$

Update: if $\delta \leq \epsilon$ then

$$z^{k+1} := z(x^k,\alpha) = X_k^{-1}\frac{e-\bar{h}}{\alpha}, \alpha = \frac{q}{z^{k^T}x^k}$$

else $z^{k+1} := z^k$

k:=k+1

UNTIL $z^{k^T}x^k < 2^{-L}$.

We observe that this version is equivalent to the first, since for any $x \in S^0$, $x^T z(x^k,\alpha)=c^k x^k - v(x^k,\alpha)$.

The dual update keeps no memory of z^k, and thus the algorithm should be seen as a primal method.

The dual update uses the construction in (4.12) as

$$z^{k+1} := z(x^k,\alpha) = X_k^{-1}(\bar{c}_p + \frac{e-e_p}{\alpha}),$$

where $\bar{c}_p = P_{AX_k} X_k c$ and $e_p = P_{AX_k} e$.

A deeper update can be achieved by setting $z^{k+1} := \tilde{z}(x^k)$ as defined in (4.8). The gap associated with this slack will be

$\tilde{\Delta}(x^k) = \min\{x^{k^T} z(x^k,\alpha) \mid \alpha \geq 0, z(x^k,\alpha) \geq 0\}$, as in (4.14). This update, the deepest possible with dual variables in the format produces useless slack, since $\tilde{Z}(x^k) \notin Z^o$.

Instead of looking for the minimum gap using dual variables in this format, Ye minimized the primal-dual potential function, and obtained the final version of his algorithm as follows:

Update: if $\delta \leq \epsilon$ then

$$z^{k+1} := \underset{z}{argmin}\{F_q(x^k,z) \mid z = X^{-1}(\bar{c}_p + \mu(e - e_p)), \mu \in \mathbb{R}, z > 0\}$$

else $z^{k+1} := z^k$

Complexity

Ye proves that if $q \geq n+\sqrt{n}$, then both primal and dual iterations produces an improvement in the potential function better than a constant. Lemma 4.9.3 is then used to prove a complexity of $O(\sqrt{n}L)$ iterations for $q=n+O(\sqrt{n})$. If $q=n+\sqrt{n}$, as in Ye's original paper [130], the algorithm will evolve by short steps - the "largest short steps" with the enhacned lower bound computations. No way of making the SSD algorithm take a single iteration between dual updates for reasonable changes in v has been found up to now. This is due to the bad conditioning of the potential function.

4.9.3 Primal-Dual Algorithms

Lemmas 4.9.2 and 4.9.3 pave the way for Primal execute dual-Algorithms. Instead of taking dual updates in the format $z = X(\bar{c}_p + \mu(e - e_p))$, we can execute primal-dual iterations to reduce potential function $F_q(x,z)$.

Any of the primal-dual directions for penalty function methods can be immediately adapted to potential reduction algorithms by setting

$$\alpha = \frac{q}{x^{k^T} z^k}.$$

Two approaches have been used:

(i) Kojima, Mizuno and Yoshise [63] used the primal-dual scaling to generate from (x, z) the direction (h_x, h_z) in (4.17)

(ii) Gonzaga and Todd [48] used the independent SSD directions (4.16), but not simultaneously. The algorithm performs primal iterations while x^k is not nearly central ie., $\| X_k h_k \| > c$. If this condition fails, the algorithm switches to dual and stays dual while a similar condition stands for the dual points. It happens proved that $z \in Z^0$ and $x \in S^0$ can not be simultaneously nearly central for $q \geq n + \sqrt{n}$ and this drives the algorithm. The convergence proof is trivial, since a sound descent is guaranteed at primal or dual points that are not nearly central and Lemma 4.8.2 can be used.

The method can switch between primal and dual arbitrarily, as long as it avoids nearly central points.

Although, almost all potential reduction methods enjoy good to excellent performance guarantees i.e., complexity bounds but they have not received much attention in terms of computational testings.

Line Searches

In primal dual methods each iteration computes a primal-dual direction (h_x , h_z) and perform a line search to reduce $F_q (x^k+\lambda_x h_x, z^k+\lambda_z h_z)$.

If we follow the direction in $\mathbb{R}^{2n}$, then $\lambda_x = \lambda_z$. and the line search is just a normal one. But it seems convenient to use the extra degree of freedom and perform a bidirectional search varying λ_x and λ_z independently.

These primal-dual methods obtain a complexity of $0(\sqrt{n}L)$ iterations for $q=n+0(\sqrt{n})$. But they are not path-following algorithms. The first one does not mention the central path; the second one avoids jamming near the path, and makes no attempt to approach it.

This suggests the interpretations for the features that ensures low complexity: a good usage of primal-dual properties are careful steps. Staying near the central path is not necessary, but nice dual slacks can be found, there. In primal-only methods the vicinity of the central path is the only region where we know how to find these slacks.

4.10 Initialization and Termination

Initialization and Termination are two most important part of any algorithm. We now discuss these two issues in the context of path-following algorithms discussed in preceding sections of this thesis.

(i) Initialization:

As mentioned in the preceding section of the thesis, most of our discussion focuses on the path-following algorithms that generate sequences of strictly feasible solutions, and these methods clearly need such a solution to start; on the other hand, practical problems rarely provide a convenient initial point. Thus it seems highly desirable to devise some way to overcome this difficulties particularly when dealing

with practical problems. In order to initiate a path-following algorithm an initial feasible solution to linear programming problem can be obtained by the introduction of an extra variable w in the problem and writing the constraints as Ax-w (Ax^0 - b) = b with w^0 = 1 and an arbitrary x^0 >0. Now, the vector (x^0, w^0) is feasible and w nust be driven to zero to find a feasible solution for the original problem.

The classical approach associates with w a large cost M (big M-method). The algorithm starts by a phase 1, that terminates when w is reduced to zero. This has been used in Adler, Karmarkar, Resende and Veiga [2].

Phase 1 and Phase 2 methods are methods that simulteneously seek feasibility and optimality. The first such method was developed by de Gellaink and Vial [34] in the projective framework. Methods that work like optimization with multiple criteria were described by Anstreicher [5] and by Todd and Wang [120] for projective methods and by Anstreicher [5] using affine potential reduction methods. Freund [25,26] used a shifted barrier approach that allows the initialization from a point that satisfies only the equality constraints, but can have negative components.

Termination:

Termination of the algorithm requires finding a vertex solution from a near optimal solution. The use of a purification algorithm like in Kortanek and Zhu [67] amounts to performing simplex-like operations, which may be inefficient. Mehrotra [81] solves the problem by a random perturbation of the cost vector.

4.11 CONCLUDING REMARKS AND OPEN QUESTIONS

Interior point methods in mathematical programming have been the largest and most dramatic area of research in optimization since the development of the simplex method by G.B. Dantzig in 1947. In this thesis we have attempted to present a unified treatment of interior point path-following methods for linear programming that are based on central trajectory.

Infact, central path for a given linear programming problem (P:minimize $c^T x$, s.t. $Ax=b$, $x \geq 0$) is a curve along which the objective function $c^T x$ decreases. This path having the tendency to stay away from the boundary of the feasible region $S= (x \in \mathbb{R}^n : Ax = b, x \geq 0)$. Geometrical and analytical properties of the central path have been studied in Bayer and Lagarias [11] and Meggido [79]. The explications in the above two works have provided basic ingredients for the construction of all path-following methods. The trajectories generated by the vector fields of various path-following algorithms have been studied in Megiddo and Shub [80] and Adler and Monteiro [1]. All path-following methods explicated in this thesis are first order methods. The direction followed by all algorithms discussed in the thesis coincide at a point on the central path: all methods then take the direction tangent to the central path. Monteiro, Adler and Resende [87] modified the primal-dual affine scaling path - following algorithm to use power series extensions which achieved a substantial reduction in the number of iterations but it was forced to increase the time per iteration. A similar approach was taken by Mehrotra

[81] which was implemented by himself and Lustig, Marsten and Shano [74] that gave very encouraging results.

Apart from linear programming problem, path- following methods have also been developed for the other class of problems. Algorithms described in Mehrotra and Sun [82], Goldfarb and Liu [36], Manteiro and Adler [86] etc. are path-following methods for quadratic programming problems. Similarly path- following algorithms have also been extended to solve linear complementarity problems and convex programming problems.

Recent computational testings indicate that path-following methods enjoy superior performance than the simplex method for large linear programming problems whereas advanced implementation of simplex code shows sometimes comparable or sometimes superior performance for the linear programing problems of smaller size.

The degeneracy in linear programming does not have much influence on the computational performance of path-following methods, whereas the simplex method shows very poor performance for most of the degenerate linear programming problems.

OPEN QUESTIONS:

We know that the existence of strongly polynomial algorithm for linear programming problem is still an open problem, which can be tried in the case of various path following algorithms discussed in this thesis.

Because of results of Megiddo and Shub [80] on associated central

trajectories, it is not believed that the primal and dual affine scaling algorithms discussed in this thesis are polynomially bounded.

However, Monteiro et al. [85] have achieved a polynomial bound for one variant of this method under several restrictions. One can try with some hope of success to achieve polynomial bound on the number of iterations in general case or some less restrictive conditions.

Recently, theoretical research on complexity of solving linear programing has focussed on developing appropriate measures for adequately representing the "size" of a linear programming problem that are more relevant to computation than the traditional measures of "size" such as the dimension m, n or the bit size L of binary representation of linear programming problem. Recently, Vavasis and Ye [127] has developed an interior point method for linear programming whose running time depends on the dimension n and certain algebraic measure of the constraint matrix A. The significance of this result derives from the fact that problem data are not presumed to be integer (or rational) coefficients and price vector c and requirement vector b do not play any role in the complexity bound. One can examine the possibilities of achieving such complexity bounds in the case of various path- following algorithms discussed in this thesis.

REFERENCES

[1] ADLER, I. and MONTEIRO, R., *'Limiting Behaviour of the Affine Scaling Continuous Trajectories for Linear Programming Problems'*, Math Programming, 50 (1991), pp. 29-51.

[2] ADLER, I., RESENDE, M., VEIGAS, G. and KARMARKAR, N., *'An implmentation of Karmarkar's Algorithm for Linear Programming'*, Math Programming, 44(1989), pp.297-335. Errate in Math. Programming, 50(1991), p. 415.

[3] ANSTREICHER, K., *'Analysis of a modified Karmarkar Algorithm for Linear Programming'*. Tech. Rep. Series B 84, Yale School of Organization and Management, New Haven, CT, 1985.

[4] _____, *'A Monotonic Projective Algorithm for Fractional Linear Programming'*. Algorithmica, 1(1986), pp.483-498.

[5] _____, *'A Combined Phase I - Phase II Scaled Potential Algorithm for Linear Programming'* CORE discussion paper 8939, CORE, Universite Catholiqué de Louvain, Louvain, Belgium, (1989), Math. Programming.

[6] _____, *'The worst case step in Karmarkar's algorithm', Mathematics of Operation Research*', 14(2), (1989). pp. 294-302.

[7] _____, *On Long Step Path Following and SUMT for Linear and Quadrataic Programming'*, Department of Operation Research, Yale University, New Haven, CT. 1990.

[8] ANSTREICHER, K. and BOSCH, R., *'Long Steps in a $0(n^3L)$ Algorithm for Linear Programming'*, Yale School of Organization and Management, New Haven, CT. 1989.

[9] BARNES, E.R., *'A Variation on Karmarkar's Algorithm for Solving Linear Programming Problems'*. Math. Programming, 36(1986), pp. 174-182.

[10] BARNES, E., JENSEN, D. and CHOPRA, *'A Polynomial-Time Version of the Affine-Scaling Algorithm'*, New York University, New York, NY, 1988.

[11] BAYER, D. and LAGARIAS, J.C., *'The non-linear geometry of Linear Programming'*. i. Affine and Projective Scaling Trajectories, ii. Lagendre Transform Coordinates iii. Central Trajectories. AT & T Bell Laboratories, Murray Hill, NJ, 1986.

[12] BLAIR, C., *'The Iterative steps in Linear Programming Algorithm of N. Karmarkar'*, Arithmetica 1(1986), pp. 537-589.

[13] BLAND, R., GOLDFARB, D. and TODD, M., *'The Ellipsoid Method: A survey'*, Oper. Res., 29(1981), pp. 1039-1091.

[14] BORGWARDT, K., *'The Simplex Method: A Probablistic Analysis'*. Springer-Verlag, Berlin, 1987

[15] BORGWARDT, K.H., *'The average number of Pivot Steps Required by the Simplex-Method is Polynomial'*, Journal for Operations Research, 26(1982b), pp. 157-177.

[16] CHARNES, A., SONG, H., and WOLFE, P., *'Research Report CCS 501, CBS'*, The University of Texas (1984).

[17] DANTZIG, G.B., *'Maximization of a Linear function of Variables subject to Linear Inequalities'*, in Activity Analysis of Production and Allocation, T.C., Koopmans, ed., John Wiley, New York, 1951, pp. 339-347.

[18] DANTZIG, G.B., *'Linear Programming and Extensions'*, Princeton University Press, Princeton, NJ, 1963.

[19] DENNIS, J.E., and MORE, J.J, SIAM Rev. 19 (1977) p. 46

[20] DIKIN, I.I., *'Iterative Solution of problems of Linear and Quadratic Programming'*, Soviet Math. Dokl. 8(1967) pp. 674-675.

[21] _____, *'On the Speed of an Iterative Process'*. Upravlyaemye Sistemy, 12(1974), pp. 54-60.

[22] FARKAS, J. *'Uber die Theorie der einfachen Ungleichungen'*. J. für die Reine and Angewandte Mathematik. 124 (1902).

[23] FIACCO, A. and McCORMIC, G., *'Non linear Programming: Sequential Unconstrained Minimization Techniques'*, John Wiley, New York, 1968.

[23a] FOURIER, J, B. J., *'Solution d'une question particuliére du calcudes inégalités'* 1826, and extracts from 'Histoire de l'Académie', 1823, 1824, Oeuvres II, pp. 317 - 328.

[24] FRANKLIN, J., *'Convergence in Karmarkar's Algorithm'*, SIAM Journal of numerical Analysis, 24(4), (1987) pp. 929-945.

[25] FREUND, R.M., *'Polynomial-Time Algorithm for Linear Programming based only on Scaling and Projected Gradients of a Potential Function'*, Math. Programming, 51(1991), pp. 203-222.

[26] _____, *'A Potential-Function Reduction Algorithm for Solving a Linear Program Directly from and Infeasible 'Warm Start'*, Tech. Rep. 3079-89-MS, Sloan School of Management, MIT, Cambridge, MA, 1989.

[27] FRISCH, K.R., *'The Logarithmic Potential Method of Convex Programming'*, University Institute of Economics, Memorandum, Oslo, Norway, 1955.

[28] GACS, P. and LOVÁSZ, L., *"Kachian's Algorithm for Linear Programming'*, Math. Programming Study, 14(1981), pp. 61-68.

[29] GALE, D., *"Neighbouring Vertices on Convex Polyhedron"*, in H.W. Kuhn and A.W. Tucker (eds), Linear Inequalities and related systems. Annals of Mathematics study No.38, Princeton University Press, N.J. (1956), pp. 255-263.

[30] GALE, D., KUHN, H. and TUCKER, A., *"Linear Programming and Theory of Games"* in T.C. Koopman edited Activity Analysis of Production and allocation", John Wiley and Sons Inc. New Jersey (1951).

[31] GARFINKEL, R.S. and NEMHAUSER, G.L., *"Integer Programming"*, John Wiley and Sons, New York, 1972.

[32] GASS, S.L. *'Linear Programming: Methods and Applications'*, McGraw - Hill Book Company, NY (1958).

[33] GAY, D., *'A Variant of Karmarkar's Linear Programming Algorithm for Problems in Standard Form'*, Math. Programming, 37(1987), pp. 81-89.

[34] GHELLAINCK, G.D. and VIAL, J.P., *"A Polynomial Newton Method for Linear Programming'*, Algorithmica, 1(1986), pp. 425-453.

[35] GILL, P., MURRAY, W., SAUNDERS, M., TOMLIN, J. and WRIGHT, M., *'On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method'*, Math. Programming, 36(1986), pp. 183-209.

[36] GOLDFARB, D. and LIU, S., *'An $O(n^3L)$ Primal Interior Point Algorithm for Convex Quadratic Programming'*, Math. Programming, 49 (1990/91), pp. 325-340.

[37] GOLDFARB, D. and TODD, M. *'Modifications and Implementation of the Shor-Khachian Algorithm for Linear Programming'*, Technical report 406, Department of Computer Science, Cornell University, Ithaca, N.Y. (1980).

[38] GOLDFARB, D. and TODD, M., *'Linear Programming', in Handbooks in Operations Research and Management Science'*, Vol. I: Optimization, G. Nemhauser, A.R. Khan and M. Todd, eds. North-Holland, Amsterdam, 1989, Chap. 2.

[39] GOLDMAN, A.J. and TUCKER, A.W., *"Polyhedral Convex Cones"* Linear Programming and Extension (1956) pp. 19 - 39.

[40] GONZAGA, C., *'Search Directions for Interior Linear Programming Methods'.* Memorandum, UCB/ERL M87/44, Electronic Research Laboratory, University of California, Berkeley, CA, March 1987.

[41] _____, *'Conical Projection Algorithms for Linear Programming'* Math. Programming, 43 (1998), pp. 151 - 173.

[42] _____, *'An Algorithm for Solving linear Programming problems in 0 (n3L,) operation'*, in progress mathematical Programming-interior point and Related Methods', N. Megiddo, ed., springer-verlag, Berlin, 1989, Chap1.

[43] _____, *'Large Step Path-following Methods for Linear Programming'*, part I: Barrier function method SIAM J. OPTIMIZATION, 1 (1991), pp. 268-279.

[44] _____, *'Large step path-following methods for Linear Programming'*, part II: potential reduction method' SIAM J. OPTIMIZATION., 1 (1991), pp. 280-292.

[45] _____, *'Convergence of the large steps primal Affine scaling Algorithm for primal Non-degenerate Linear Programming'*, Report, COPPE-Federal UNiversity of Rio de Janeiro, Brasil, 1990.

[46] _____, *'On lower Bound updates in primal potential Reduction Methods for Linear Programming'*, Math. Programming, 52 (1991), pp. 415-428.

[47] _____, *'Polynomial Affine Algorithms for Linear Programming'*., Math. Programming, 49 (1990) pp.7-21.

[48] _____, GONZAGA, C. and TODD, M.J., *"An O (√nL) - iteration large-step primal-Dual Affire Algorithm for Linear Programming'*, Tech. Rep. 862, School of operations Research and Industrial Engineering, Cornell University, Ithaca, NY, (1989), SIAM. J. Optim, 2 (1992).

[49] GROTSCHEL, M., LOVASZ, L. and SCHRIJVER, A., *'The Ellipsoid Method and its Consequences in Combinatorial Optimization'*, combinatorica I (1981)pp.169-197.

[50] HERTOG,D.D., ROOS,C. and TERLARY,T., *'A potential reduction varian of Renegar's Short Path-Following Method for Linear Programming'*, linear Algebra Appl., 152 (1991), pp. 43-68.

[51] HERTOG, D.D., ROOS, C. and VIAL, J.P., *"Polynomial-time Long-steps Algorithms for linear Programming based on the use of the Logarithmic Barrier function'*, Faculty of Technical Mathematics and Informatics, Technische Universiteit Delft, Holland, 1990.

[52] HUARD, P., *'Resolution of Mathematical Programming with non-linear constraints by the methods of centers'*, in Non-linear Programming, J. Abadie, ed., North-Holland, Amsterdam, 1967.

[53] IMAI, H., *'On the Convexity of the Multiplicative Version of Karmarkar's Potential Function'*, Math. Programming, 40(1988), pp. 29-32.

[54] IUDIN, D.B. and NEMIROSVSKII, A.S., *'Evaluation of the Informational Complexity of Mathematical Programming Problems'*, Ekonomeka i Matematicheskiw methody 12 (Translations of Russian and East European Math Economics, 13 (1976) pp 3-25.

[55] IUDIN, D.B. and NEMIROVSKII, A.S. *'Informational Complexity and Effective Methods of Solution for Convex Externla problems'*, Translations of Russian and East European Math. Economics 13, (1976) pp. 25-45

[56] JONES, P.C. and MARWIL, E.S., *'A variant of Khachian's Algorithm for Linear Programming'*, G. & G Idaho, Inc. P.O.B. 1625, Idaho Falls, Idaho (1979).

[57] KANTROVICH, L., *'The Best use of Economic Resources'*. Oxford (1965).

[58] KARMARKAR, N., *'A New Polynomial-time Algorithm for Linear Programming'*, Combinatorica, 4 (1984), pp. 373-395.

[59] KHACHIAN, L.G., *'A Polynomial Algorithm for Linear Programming'*, Soviet Math. Dokl., 20 (1979). pp. 191-194.

[60] _____, *'Polynomial Algorithms in Linear Programming'*, USSR Computational Mathematics and Math. Phys., 20 (1980), pp. 53-72.

[61] KHACHIAN, L. and TODD, M.J., *'On the Complexity of Approximating the Maximal Inscribed Ellipsoid for a Polytope'*, Math. Programming. North-Holland Vol. 61(1993), pp. 137-159.

[62] KLEE, V. and MINTY, G., *"How good is the Simplex Algorithm?'*, in inequalities III, O. Sisha, ed., Academic Press, New York, NY, 1972.

[63] KOJIMA, M., MIZUNO, S. and YOSHISE, A., *'An O ($\sqrt{n}$ L) Iteration Potential Reduction Algorithm for Linear Complementary Problems'*, Math. Programming 50 (1991), pp. 331-342.

[64] _____, *'A polynomial-time Algorithm for a Class of Linear Complementary Problems'*, Math. Programming, 44 (1989), pp. 1-6.

[65] _____, *'A Primal-Dual Interior Point Method for Linear Programming'*, Math Programming-interior point and related methods, N. Megiddo, ed., springer-verslag, Berlin, 1989.

[66] KOJMA, M., SHINJI and YOSHISE, A., *'A Primal-Dual Interior Point Algorithm for Linear Programming'*, (ed. Megidd, N.) Springer-Verlag New York, Inc. (1989) pp. 30-47.

[67] KORTANEK, K. and ZHU, J., *'New Purification algorithms for linear programming'*. Naval Res. Logist. 35 (1988), pp. 571 - 583.

[68] KOZLOV, M., TARASOV, S. and KHACHIAN, L., *'Polynomial Solvability of Convex Quadratic Programming'* Dokl. Acad. Nauk, SSSR, 5 (1979), pp. 1051-1053.

[69] KROL, L. and MIRMAN, B., *'Some Practical Modifications of Ellipsoid Method for Linear Programming'*, Arcon, Inc., Boston (1980).

[70] KUHN, H. W., *'Duality in Mathematical Programming'*, Mathematical Systems Theory and Economics. Springer - Verlag. NY (1969), pp. 67 - 91.

[71] LEMKE, C.E., *'Bimatrix Equilibrium Points and Mathematical Programming'*, Management Sci. 11(1965) pp. 681-689.

[72] LEMKE, C.E. *"The Dual method of solving the linear Programming problem"*, Naval Research Logistics Quarterly 1, 1, 1954, pp. 36-47

[73] LUENBERGER, D.G., *"Introduction to linear Programming"*, Addison-Wesley Publishing company, Inc. Ma., 1973

[74] LUSTIG, I., MARSTEN, R. and SHANNO, D., *'On Implementing Mehrotra's predictor-corrector Interior Point Method for Linear Programming'*, Tech. Rep. Sor 90-03, Dept. of Civil Engineering and Operations Research, Princeton University, Princeton, NJ, 1990.

[75] MC DIARMID, C.J.H; *'On Improvement Per Iteration in Karmarkar's Method for Linear Programming'*, Math Programming 40 (1990) pp.290-320

[76] MC SHANE, K. MONMA, C. and SHANNO, D. *'An Implementation of a Primal-Dual Interior Point Method for Linear Programming'*, ORSA J. Comput., 1 (1989), pp. 70-83.

[77] MEGIDDO, N., *'Linear Programming in Linear Time when the Dimension is fixed'*, J. Assoc. Comput. Mach., 31 (1984), pp. 114-127.

[78] _____, *'On the Complexity of Linear Programming'*. in Advances in Economic Theory, T. Bewley ed., 1987, Cambridge University Press, Cambridge, MA, pp.225-268.

[79] _____, *'Pathways to the Optimal Set in Linear Programming'*, in Progress in Mathematical Programming - interior point and related methods, N. Megiddo, ed., Springer- verlag, Berlin, 1989, chap 8.

[80] MEGIDDO, N. AND SHUB, M., *'Boundary Behaviour of Interior Point Algorithms in Linear Programming'*, Math. of Oper. Res., 14 (1989), pp. 97-146.

[81] MEHROTRA, S., *'On Finding a Vertex Solution using Interior Point Methods'*, Linear Algebra and Appl. 152 (1991), pp. 233-253.

[82] MEHROTRA.S. and SUN, J., *'An Algorithm for Convex Quadratic Programming that requires O ($n^3$5L) Arithmetic operation'*, Math. Oper. Res. 15 (1990), pp.342-363.

[83] MIZUNO, S., *'O (n^0L) Iteration O (n^2 L) Potential Reduction Algorithms Linear Programming'*. Linear Algebra and Appl., 152 (1991), pp. (155-168).

[84] MONMA, C. and MORTON, A., *'Computational experience with a Dual Affine Variant of Karmarkar's Method for Linear Programming'*, Oper. Res. Lett., 6 (1987), pp. 271-267.

[85] MONTEIRO, R.C. and ADLER, I. *'Interior Path-following Primal-Dual Algorithm'*, Part I". Linear Programming, Math. Programming, 44(1989), pp. 27-41.

[86] MONTEIRO, R. and ADLER, I., *'An Extension of Karmarkar Type Algorithm to a Class of Convex Seperable Programming Problems with Global Linear rate of Convergence'*. Math. of Oper. Res 15 (1990), pp. 408-422.

[87] MONTEIRO, R.C., ADLER, I. and RESENDE, M., *'A Polynomial-time Primal-Dual Affine Scaling Algorithm for Linear and Convex Quadratic Programming and its Power Series Extension'*, Math. of Oper. Res., 15 (1990), pp. 191-214.

[88] MURTY, K.G., *'A New Interior Variant of the Gradient Projection Method for Linear Programming'*, Technical paper 85-18, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor (may, 1985).

[89] MYLENDER, W., HOLMES, R. and McCORMICK, *' A guide to SUMT-Version 4: The computer program implementing the sequential unconstrained minimization technique for non-linear programming'*, Research paper RAC-P- 63, Research Analysis Corporation McLean, VA, 1971.

[90] NAZARETH, J.L., *'Homotopy Techniques in Linear Programming'*, Algorithmica, 1 (1986), pp. 529-535.

[91] PADBERG, M.W., *'A Different Convergence Proof of the Projective Methods for Linear Programming'*, Operation Research Letters, 4 (1986), pp. 235-237.

[92] PADBERG, M.W. and RAO, M.R. *'The Russian method for linear inequalities'*, Graduate School of Business Admin. N.Y. Univ, N.Y. (1980).

[93] PADBERG, M.W. and RAO, M.R., *'The Russian Method for Inequalities II: Approximate Arithmetic'*, Graduate School of Business Admin., N.Y. Univ., N.Y. (1980).

[94] POLAK, E., *'Computational Methods in Optimization'*. Academic press, New York, NY, 1971.

[94a] POUSSIN, M. Ch. J. de la Vallee, *'Sur la methode de l' approximation minimum'*. Ann. Soc. Sci. de Bruxelles, 35 (1911), pp. 1 -16.

[95] RENEGAR, J., *'A Polynonial-time Algorithm Based on Newton's Method for Linear Programming'*, Math. Programming, 40 (1988), pp. 59-94.

[96] RENEGAR. J. and SHUB, M., *'Simplified Complexity Analysis for Newton LP Methods'*, Tech. Rep. 807, School of Operations Research and Industrial Engineering Cornell University, Ithaca, NY, 1988.

[97] ROOS, C. and VIAL, J.P., *'A Simple Polynomial Method of Approximate Centres for Linear Programming', Report*, Faculty of Technical Mathematics and Informatics, Technische Universiteit Delft Holland, 1988.

[98] _____, *'Long steps with the Logarithmic penalty Barrier function in Linear Programming'*, in Economic Decision-Making: Games, Economics and Optimization J. Gabszeywicz, J.F. Richard and L. wolsey, eds., Elserier Science Publishers B.V., Amsterdam, 1989, pp. 433-441.

[99] SCHRIJVER, A., *'Theory of Linear and Integer Programming'*, John Willey, New York, NY, 1986.

[100] SETHI, A.P. and THOMPSON, G.L. *"The Pivot and Probe Algorithm for Solving a Linear Program"*, Mathematical Programming, 29 (1984) North Holland, pp. 219-233.

[101] SETHI, A.P. and THOMPSON, G.L., *"The Non-candidate Constraint Method for Reducing the size of a Linear Program"*, in M.H. Karwan, V.Lofti, J. Telgen and S. Zoints, eds, Redundancy in Mathematical Programming 'Springer, Berlin, Heidelberg, NY, Tokyo (1983).

[102] SHAMIR, R., *'The Efficiency of the Simplex Method'*: A survey Management Sci., 33 (1987), pp. 301-334.

[103] SHOR, N., *'Utilization of the Operation of Space Dilation in the Minimization of Convex Functions'*, Kibernetika, 1 91970) pp. 6-12 ; English Translation: Cybernetics, 6,pp. 7-15.

[104] SHOR, N.Z., *'Utilization of the operation space Dilation in the minimization of Convex functions'*. Kibernetika 6 (2) 6-12 (translated in Cybetrnatics 6 (1), (1970) pp. 7-15.

[105] SHOR, N.Z., *'Cut-off Method with Space Extension in Convex Programming Problems'*, Kibernetika 13 (1) (translated in cybernetics 13 (1). pp 94-96). (1977).

[106] SHOR, N.Z. and GERSHOVICH, V.I., *'Family of Algorithms for solving Convex Programming problems'*. Cybernetics 15 (4), (1979), pp. 502 - 507.

[107] SINGH,, D. and SINGH,, J.N., *'Some Remarks on Karmarkar's Algorithm for Linear Programming'*, Journal of Bihar Math. Soc. India, 11 (1987/88), pp. 85-90.

[108] SINGH, J.N., *'The Ellipsoid Method and Karmarkar's Projective Algorithm"*, (Accepted for Fifth SIAM Conference on Applied Linear Algebra, Snowbord, Uttah, USA, June 15-18 (1994).

[109] SINGH,, J.N. *"A comparison of Two Recent Polynomial Algorithms for Linear Programming"*, Mathematics and Computer Education. Vol.... (1997) pp. 167-175.

[110] SINGH, J.N. and SINGH,, D., *'Some Remarks on the Choice of Steplengths in Karmarkar's Algorithm'*, Acta Cenecia, India, vol.-21(1)M (1995) pp. 53-58.

[111] SINGH, J.N. and SINGH, D., *"A Modified Termination Rule for Karmarkar's Algorithm'*, Presented at the fourth SIAM Conference on optimization, Chicago, May 11-13, 1992.

[112] SINGH, J.N. and SINGH,, D., *'Convergence in Karmarkar's Algorithm'; a Review'*, Int. Journal Math.Sci. technol (U.K.) 25, (Nov 3, 1994) pp. 333-341.

[113] SMALE, S., *"On Average Number of Steps of Simplex Method of Linear Programming"*, Mathematical Programing 27(1983), pp. 241-262.

[114] SONNEVEND, G., *'An Analytic Centre for Polyhedrons and New Classes of Global Algorithms for Linear (Smooth, Convex) Programming'*, in Lecture Notes Control Inform. Sci. 84, Springer-Verlag. New York, NY, 1985, pp. 866-876.

[115] STEGER, A., *'An Extension of Karmarkar's Algorithm for Bounded Linear Programming Problems', Master's Thesis'*, Department of Applied Mathematics, State University of New York, Story Brook, NY, August 1985. Condensed Version in Operations Research Proceedings 1987, H. Schellhaas, P. Van Beek, H. Isermann, R. Schmidt and M. Zijlstra, eds., Springer-verlag Berlin, west Germany, 1988, pp. 88-95

[116] TARDOS. E., *'A Strongly Polynomial Algorithm to solve Combinatorial Linear programs'*, Open. Res. 35 (1986). pp. 2250-256.

[117] TODD M.J. *'Improved Bounds and Containing Ellipsoids in Karmarkar's Linear Programming Algorithm'.* Manuscript, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York (1987).

[118] TODD, M.J. *'Polynomial Algorithm for Linear Programming'*, Advances in Optimization and Control, H.A. Eiseit and G. Pederzole (eds.) Springer-Verlag (1988).

[119] TODD, M. and BURRELL, B., *'An Extension of Karmarkar's Algorithm for Linear Programming using Dual Variables'*, Algorithmica, 1 (1986), pp. 409-424.

[120] TODD, M. J. and WANG, Y., *'On combined phase 1 - phase 2 projective methods for linear programming'.* Tech. Rep. 877, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, (1989).

[121] TODD M.J. and YE, Y., *'A Centered Projective Algorithm for Linear Programming'*, Math. of Oper. Res., 15 (1990), pp. 508-529.

[122] VAIDYA, P., *'An Algorithm for Linear Programming which requires* $((m+n)n^2 + (m+n)^{1.5}nL))$*'*. Math,. Programming, 47 (1990), pp. 175-201

[123] _____, *'A Locally Well-behaved Potential Function and a Simple Newton-tipe Method for Finding the Centre of a Polytope'*, in

progress in Mathematical Programming-Interior Point and Related Methods, N. Megiddo, ed., Springer-Verlag, Berlin, 1989, Chap. 6.

[124] _____, *' A New Algorithm for Minimizing a Convex Function over convex sets', in* proceedings of the 30th Annual Symposiom on Foundation of Computer Science, Research Triangle Park, NC 1989, IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 338-343.

[125] VANDERBEI, R.J. and LAGARIAS, J.C., 'I. *'Dikin's Convergence Results for the Affine-Scaling Algorithm',* in Mathematical Developments Arising from Linear Programming: Proceedings of a Joint Summer Research Conference held at Bowdoin College, Brunsweick, Maine, June/July 1988, J.C. Lagarias and M.J. Todd, eds., Contemp. Math., 114 American Mathematical Society, Providence, RI, 1990, pp. 109-119.

[126] VANDERBEI, R. J., MEKETON, M.J. and FREEDMAN B.A., *'A Modification of Karmarkar's Linear Programming Algorithm'*, Algorithmica, 1 (1986) pp. 395-407.

[127] VAVASIS, S and YE, Y., *'A Primal-dual Interior Point Method whose Running Time Depends only on the Constraint matrix",* Math. Programming 74(1996), pp. 79-120.

[128] VON NEUMANN, JOHN, *"On a Maximization Problem"*. (manuscript), Institute of Advanced study, Princton, New Jersey, USA (November, 1947).

[129] YE, Y., 'An O (n^3L) *'Potential Reduction Algorithm for Linear Programming'*, Math. Programming, 50 (1990), pp. 239-258.

[130] _____, *'Line Search in Potential Reduction Algorithms for Linear Programming',* Dept. of Management Sciences, The University of Iowa city, IA, 1989, Manuscript.

[131] YE, Y. and KOJIMA, M., *'Recovering Optimal Dual Solutions in Karmarkar's Polynomial Algorithm for Linear Programming',* Math. Programming 39 (1987), pp. 305-317.

[132] YE, Y. and TSE, E., *'Karmarkar's Algorithm and the Ellipsoid Method'* , Operations Research Letters 6(4) (1987), pp. 177-182.

APPENDIX

A computer program to implement modified Pivot and Probe Algorithm.

```
{=============================================================================}
{=============================================================================}
{ A COMPUTER PROGRAM TO IMPLEMENT MODIFIED VERSION OF PIVOT                   }
{ AND PROBE ALGORITHM (PAPA) CODED USING PASCAL PROGRAMMING                   }
{ LANGUAGE.                                                                   }
{=============================================================================}
{=============================================================================}
{ THE CONSTANTS AND VARIABLES USED HAVE CLEARLY BEEN SPECIFIED                }
{ AND DOCUMENTED WITHIN THE MODIFIED VERSION OF PAPA PROGRAM                  }
{ USERS OF MODIFIED PAPA PROGRAM CAN CHANGE THE CONST.                        }
{ m : Number of contraints and                                                }
{ n : Total number of variables                                               }
{ DEPENDING ON THE "Dimension OF matrix A" IN THE                             }
{ STANDARD FORM OF THE LINEAR PROGRAMMING PROBLEM.                            }
{                                                                             }
{    NOTE THAT LP PROBLEM MUST BE CONVERTED TO STANDARD                       }
{ FORM BEFORE ENTRING DATA USING THE FORM BELOW:                              }
{ a(1,1),a(1,2),-------,a(1,n),a(1,n+1),a(1,n+2),-------,a(1,n+m),b(1)        }
{  a(2,1),a(2,2),-------,a(2,n),a(2,n+1),a(2,n+2),------ ,a(2,n+m),b(2)       }
{   :                                  :     :                                }
{   :                                  :     :                                }
{  a(m,1),a(m,2),------,a(m,n),a(m,n+1),a(m,n+m),-------,a(m,n+m),b(m)        }
{ where,  a's are elements of matrix A and b's are elements of matrix B       }
{                                                                             }
{ next, the coeffs. of obj. funct.                                            }
{ c(1),c(2),-------,c(n),c(n+1),c(n+2),---------,c(n+m)                       }
{                                                                             }
{ next, the value of the obj. funct.( zo ) at  initial    iteration           }
{ and lastly                                                                  }
{ the baseindex number of starting BFS.                                       }
{=============================================================================}
{=============================================================================}
{ EXAMPLES SOLVED BY  MODIFIED VERSION OF PAPA  PROGRAM  AND THEIR            }
{ OUTPUT TABLEAUS APPEARED ON THE APPENDIX.                                   }
{=============================================================================}
{=============================================================================}
```

```
PROGRAM  modipapa (input, output);
uses dos, crt;
CONST
   m=4; n=6;       { m for Number of constraints and }
                   { n for total number of variables }
   fwt=8; dpt=2;   { output format constants for tableau values }
   fwi=1;          { output format constant for indies }
```

```
  largevalue = 1.0E20; smallvalue = 1.0E-10;


TYPE mrange = 1..m;  nrange = 1..n;
   matrix = ARRAY [ mrange, nrange ] OF real;
   column = ARRAY [ mrange] OF REAL;
   baseindex = ARRAY [ mrange ] OF integer;
   row = ARRAY [ nrange ] OF real;
   rowboolean = ARRAY [ nrange ] OF boolean;



VAR
  a : matrix;            { Matrix A in standard form of problem }
  b : column;            { vector b in standard form of problem }
  c : row;               { coefficients of objective function }
  basic : baseindex;     { Basic variables at each stage ofiteration}
  nonbasic : rowboolean; { Status indicators for variables }
  zo : real;             { Value of obj. funct. at current iteration}
  it : integer;          { Iteration counter}
  solution, unbounded : boolean; { iteration process terminators }
  r, s : integer;              { Row and column of pivot element }


PROCEDURE inputdata;
VAR i, j : integer;
BEGIN
   FOR i:= 1 TO m DO
   BEGIN  FOR j:= 1 TO n DO  READ(a[i,j]); READ(b[i]) END;
   FOR j:= 1 TO n DO  READ(c[j]);   READ(zo);
   FOR i:=1 TO m DO   READ(basic[i]);  { initial base }
END;                                { inputdata }


PROCEDURE initialise;
VAR i, j : integer;
BEGIN it:= 0; solution:=false;  unbounded:=false;
     FOR j:= 1 TO n DO nonbasic[j]:=true;
     FOR i:= 1 TO m DO nonbasic[basic[i]]:=false
END;                                { initialise }


PROCEDURE outputtableau;
VAR  i, j : integer;
```

```
BEGIN WRITELN; WRITELN('       ITERATION       ', it:2);
    WRITELN;
    WRITE('   B F S    ',' ':fwt-5, 'VALUE');
    FOR j:=1 TO n DO  WRITE(' ':fwt-fwi-1, 'X', j:fwi); WRITELN;
    FOR i:=1 TO m DO
    BEGIN
       WRITE(' ':8-fwi, 'X',  basic[i]:fwi, ' ':7,b[i]:fwt:dpt);
       FOR j:=1 TO n DO  WRITE(a[i,j]:fwt:dpt); WRITELN
    END;
    WRITE(' ':7, '-Z', ' ':7, zo:fwt:dpt);
    FOR j:=1 TO n  DO   WRITE(c[j]:fwt:dpt); WRITELN
END;                                { outputtableau }

PROCEDURE  nextbasicvariable  ( VAR r, s:integer);
VAR   i, j : integer; min : real;
BEGIN  min:=largevalue;                  { Find the variable, s, }
     FOR j:=1 TO n DO                  { to enter the basis.   }
       IF nonbasic[j]  THEN  IF c[j]<min THEN BEGIN min:=c[j]; s:=j END;
     solution := c[s] > -smallvalue;
     IF NOT  solution   THEN
     BEGIN unbounded:=true; i:=1;          { check that atleast one value }
       WHILE   unbounded  AND  (i<=m) DO { in column s is positive.    }
       BEGIN unbounded := a[i,s] < smallvalue; i:=i+1   END;
       IF NOT unbounded THEN
       BEGIN  min:=largevalue;          { Find the variable  basic[r] }
        FOR i:=1 TO m DO               { to leave the basis.        }
          IF a[i,s] > smallvalue  THEN
            IF  b[i]/a[i,s] < min THEN BEGIN min:=b[i]/a[i,s]; r:=i END;
        nonbasic[basic[r]]:=true; nonbasic[s]:=false; basic[r]:=s; WRITELN;
        WRITELN('  PIVOT IS AT  ROW ', r :fwi, '  COL ',s : fwi )

       END
     END
END;                                { nextbasicvariable }


PROCEDURE  transformtableau ( r, s:integer);
VAR  i, j : integer; pivot, savec : real; savecol : column;
BEGIN
    FOR i:=1 TO m DO savecol[i]:=a[i,s]; savec:=c[s]; pivot:=a[r,s];
    b[r]:=b[r]/pivot;
    FOR j:=1 TO n DO  a[r,j]:=a[r,j]/pivot;
    FOR i:=1 TO m DO
      IF i<>r THEN
```

```
      BEGIN b[i]:= b[i] - savecol[i]*b[r];
          FOR j:=1 TO n DO a[i,j] := a[i,j] - savecol[i]*a[r,j]
      END;
   FOR j:= 1 TO n DO c[j]:=c[j] - savec*a[r,j];
   zo:= zo - savec*b[r]; it:= it + 1
END;                                { transformtableau }


BEGIN   { main program }
CLRSCR;
TEXTBACKGROUND(blue);
CLRSCR;
TEXTCOLOR(white);
TEXTBACKGROUND(blue);
WRITELN;
WRITELN;
WRITELN('                                                    ');
WRITELN('                                                    ');
WRITELN('                                                    ');
WRITELN('                                                    ');
WRITELN('                                                    ');
WRITELN;
WRITELN;
WRITELN('              MODIFIED  PIVOT AND PROBE ALGORITHM (PAPA)');
WRITELN('              ----------------------------------------');
WRITELN('                    A  VARIANT  OF  SIMPLEX ALGORITHM');
WRITELN('                    ---------------------------------');
WRITELN;
WRITELN(' Enter data values ');
inputdata; initialise;
REPEAT
    outputtableau;
    nextbasicvariable(r,s);
    IF NOT ( solution OR unbounded ) THEN transformtableau(r,s)
UNTIL solution OR unbounded;
WRITELN;                            { Displaying results }
IF unbounded THEN WRITELN('    VARIABLE  ',s:fwi,' IS  UNBOUNDED ')
ELSE WRITELN ('     MINIMUM  Z = ', -zo:fwt:dpt)
END.                                {modipapa }
```



EXAMPLES: LINEAR PROGRAMMING PROBLEMS SOLVED BY THE ABOVE COMPUTER PROGRAMME.

Problem 1 (Example p-35):
Maximise $x_0 = x_1 + x_2$
such that, $-x_1 + 2x_2 \leq 12$
$2x_1 - x_2 \leq 12$
$x_1 + x_2 \leq 12$
$2x_1 + 3x_2 \geq 30$
$x_1, x_2 \geq 0$

```
Solution:
Enter data values:
-1
2
1
0
0
```

ITERATION 0

B F S	VALUE	X1	X2	X3	X4	X5	X6
X3	12.00	-1.00	2.00	1.00	0.00	0.00	0.00
X4	12.00	2.00	-1.00	0.00	1.00	0.00	0.00
X5	12.00	1.00	1.00	0.00	0.00	1.00	0.00
X6	30.00	2.00	3.00	0.00	0.00	0.00	1.00
-Z	0.00	-1.00	-1.00	0.00	0.00	0.00	0.00

PIVOT IS AT ROW 2 COL 1

ITERATION 1

B F S	VALUE	X1	X2	X3	X4	X5	X6
X3	18.00	0.00	1.50	1.00	0.50	0.00	0.00
X1	6.00	1.00	-0.50	0.00	0.50	0.00	0.00
X5	6.00	0.00	1.50	0.00	-0.50	1.00	0.00
X6	18.00	0.00	4.00	0.00	-1.00	0.00	1.00
-Z	6.00	0.00	-1.50	0.00	0.50	0.00	0.00

PIVOT IS AT ROW 3 COL 2

ITERATION 2

B F S	VALUE	X1	X2	X3	X4	X5	X6
X3	12.00	0.00	0.00	1.00	1.00	-1.00	0.00
X1	8.00	1.00	0.00	0.00	0.33	0.33	0.00
X2	4.00	0.00	1.00	0.00	-0.33	0.67	0.00
X6	2.00	0.00	0.00	0.00	0.33	-2.67	1.00
-Z	12.00	0.00	0.00	0.00	0.00	1.00	0.00

MINIMUM Z = -12.00

Hence, maximum x_0 = 12.00

problem 2:

Maximise $x_0 = 2x_1 + 5x_2 + 7x_3$
such that, $3x_1 + 2x_2 + 4x_3 \leq 100$
$x_1 + 4x_2 + 2x_3 \leq 100$
$x_1 + x_2 + 3x_3 \leq 100$
$x_1, x_2, x_3 \geq 0$

```
Solution:
Enter data values:
3
2
4
1
0
0
100
1
4
2
0
1
0
```

ITERATION 0

B F S	VALUE	X1	X2	X3	X4	X5	X6
X4	100.00	3.00	2.00	4.00	1.00	0.00	0.00
X5	100.00	1.00	4.00	2.00	0.00	1.00	0.00
X6	100.00	1.00	1.00	3.00	0.00	0.00	1.00
-Z	0.00	-2.00	-5.00	-7.00	0.00	0.00	0.00

PIVOT IS AT ROW 1 COL 3

ITERATION 1

B F S	VALUE	X1	X2	X3	X4	X5	X6
X3	25.00	0.75	0.50	1.00	0.25	0.00	0.00
X5	50.00	-0.50	3.00	0.00	-0.50	1.00	0.00
X6	25.00	-1.25	-0.50	0.00	-0.75	0.00	1.00
-Z	175.00	3.25	-1.50	0.00	1.75	0.00	0.00

PIVOT IS AT ROW 2 COL 2

ITERATION 2

B F S	VALUE	X1	X2	X3	X4	X5	X6
X3	16.67	0.83	0.00	1.00	0.33	-0.17	0.00
X2	16.67	-0.17	1.00	0.00	-0.17	0.33	0.00
X6	33.33	-1.33	0.00	0.00	-0.83	0.17	1.00
Z	200.00	3.00	0.00	0.00	1.50	0.50	0.00

MINIMUM Z = -200

Hence, maximum x_0 = 200

Problem 3:

Minimise $Z = -x_1 + 2x_2$

such that, $-x_1 + 3x_2 \leq 10$

$x_1 + x_2 \leq 6$

$x_1 - x_2 \leq 2$

$x_1, x_2 \geq 0$

Solution:

```
Enter data values:
 1
3
1
0
0
10
1
1
0
1
0
6
```

ITERATION 0

B F S	VALUE	X1	X2	X3	X4	X5
X3	10.00	-1.00	3.00	1.00	0.00	0.00
X4	6.00	1.00	1.00	0.00	1.00	0.00
X5	2.00	1.00	-1.00	0.00	0.00	1.00
-Z	0.00	-1.00	2.00	0.00	0.00	0.00

PIVOT IS AT ROW 3 COL 1

ITERATION 1

B F S	VALUE	X1	X2	X3	X4	X5
X3	12.00	0.00	2.00	1.00	0.00	1.00
X4	4.00	0.00	2.00	0.00	1.00	-1.00
X1	2.00	1.00	-1.00	0.00	0.00	1.00
Z	2.00	0.00	1.00	0.00	0.00	1.00

MINIMUM Z = -2.00