

Two dimensional search algorithms for linear programming

by

Fabio Torres Vitor

B.S., Maua Institute of Technology, Brazil, 2013

M.S., Kansas State University, 2015

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Industrial and Manufacturing Systems Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2019

Abstract

Linear programming is one of the most important classes of optimization problems. These mathematical models have been used by academics and practitioners to solve numerous real world applications. Quickly solving linear programs impacts decision makers from both the public and private sectors. Substantial research has been performed to solve this class of problems faster, and the vast majority of the solution techniques can be categorized as one dimensional search algorithms. That is, these methods successively move from one solution to another solution by solving a one dimensional subspace linear program at each iteration. This dissertation proposes novel algorithms that move between solutions by repeatedly solving a two dimensional subspace linear program. Computational experiments demonstrate the potential of these newly developed algorithms and show an average improvement of nearly 25% in solution time when compared to the corresponding one dimensional search version.

This dissertation's research creates the core concept of these two dimensional search algorithms, which is a fast technique to determine an optimal basis and an optimal solution to linear programs with only two variables. This method, called the slope algorithm, compares the slope formed by the objective function with the slope formed by each constraint to determine a pair of constraints that intersect at an optimal basis and an optimal solution.

The slope algorithm is implemented within a simplex framework to perform two dimensional searches. This results in the double pivot simplex method. Differently than the well-known simplex method, the double pivot simplex method simultaneously pivots up to two basic variables with two nonbasic variables at each iteration. The theoretical computational complexity of the double pivot simplex method is identical to the simplex method. Computational results show that this new algorithm reduces the number of pivots to solve benchmark instances by approximately 40% when compared to the classical implementation of the simplex method, and 20% when compared to the primal simplex implementation of

CPLEX, a high performance mathematical programming solver. Solution times of some random linear programs are also improved by nearly 25% on average.

This dissertation also presents a novel technique, called the ratio algorithm, to find an optimal basis and an optimal solution to linear programs with only two constraints. When the ratio algorithm is implemented within a simplex framework to perform two dimensional searches, it results in the double pivot dual simplex method. In this case, the double pivot dual simplex method behaves similarly to the dual simplex method, but two variables are exchanged at every step.

Two dimensional searches are also implemented within an interior point framework. This dissertation creates a set of four two dimensional search interior point algorithms derived from primal and dual affine scaling and logarithmic barrier search directions. Each iteration of these techniques quickly solves a two dimensional subspace linear program formed by the intersection of two search directions and the feasible region of the linear program. Search directions are derived by orthogonally partitioning the objective function vector, which allows these novel methods to improve the objective function value at each step by at least as much as the corresponding one dimensional search version. Computational experiments performed on benchmark linear programs demonstrate that these two dimensional search interior point algorithms improve the average solution time by approximately 12% and the average number of iterations by 15%.

In conclusion, this dissertation provides a change of paradigm in linear programming optimization algorithms. Implementing two dimensional searches within both a simplex and interior point framework typically reduces the computational time and number of iterations to solve linear programs. Furthermore, this dissertation sets the stage for future research topics in multidimensional search algorithms to solve not only linear programs but also other critical classes of optimization methods. Consequently, this dissertation's research can become one of the first steps to change how commercial and open source mathematical programming software will solve optimization problems.

Two dimensional search algorithms for linear programming

by

Fabio Torres Vitor

B.S., Maua Institute of Technology, Brazil, 2013

M.S., Kansas State University, 2015

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Industrial and Manufacturing Systems Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2019

Approved by:

Major Professor
Todd Easton

Copyright

© Fabio Torres Vitor 2019.

Abstract

Linear programming is one of the most important classes of optimization problems. These mathematical models have been used by academics and practitioners to solve numerous real world applications. Quickly solving linear programs impacts decision makers from both the public and private sectors. Substantial research has been performed to solve this class of problems faster, and the vast majority of the solution techniques can be categorized as one dimensional search algorithms. That is, these methods successively move from one solution to another solution by solving a one dimensional subspace linear program at each iteration. This dissertation proposes novel algorithms that move between solutions by repeatedly solving a two dimensional subspace linear program. Computational experiments demonstrate the potential of these newly developed algorithms and show an average improvement of nearly 25% in solution time when compared to the corresponding one dimensional search version.

This dissertation's research creates the core concept of these two dimensional search algorithms, which is a fast technique to determine an optimal basis and an optimal solution to linear programs with only two variables. This method, called the slope algorithm, compares the slope formed by the objective function with the slope formed by each constraint to determine a pair of constraints that intersect at an optimal basis and an optimal solution.

The slope algorithm is implemented within a simplex framework to perform two dimensional searches. This results in the double pivot simplex method. Differently than the well-known simplex method, the double pivot simplex method simultaneously pivots up to two basic variables with two nonbasic variables at each iteration. The theoretical computational complexity of the double pivot simplex method is identical to the simplex method. Computational results show that this new algorithm reduces the number of pivots to solve benchmark instances by approximately 40% when compared to the classical implementation of the simplex method, and 20% when compared to the primal simplex implementation of

CPLEX, a high performance mathematical programming solver. Solution times of some random linear programs are also improved by nearly 25% on average.

This dissertation also presents a novel technique, called the ratio algorithm, to find an optimal basis and an optimal solution to linear programs with only two constraints. When the ratio algorithm is implemented within a simplex framework to perform two dimensional searches, it results in the double pivot dual simplex method. In this case, the double pivot dual simplex method behaves similarly to the dual simplex method, but two variables are exchanged at every step.

Two dimensional searches are also implemented within an interior point framework. This dissertation creates a set of four two dimensional search interior point algorithms derived from primal and dual affine scaling and logarithmic barrier search directions. Each iteration of these techniques quickly solves a two dimensional subspace linear program formed by the intersection of two search directions and the feasible region of the linear program. Search directions are derived by orthogonally partitioning the objective function vector, which allows these novel methods to improve the objective function value at each step by at least as much as the corresponding one dimensional search version. Computational experiments performed on benchmark linear programs demonstrate that these two dimensional search interior point algorithms improve the average solution time by approximately 12% and the average number of iterations by 15%.

In conclusion, this dissertation provides a change of paradigm in linear programming optimization algorithms. Implementing two dimensional searches within both a simplex and interior point framework typically reduces the computational time and number of iterations to solve linear programs. Furthermore, this dissertation sets the stage for future research topics in multidimensional search algorithms to solve not only linear programs but also other critical classes of optimization methods. Consequently, this dissertation's research can become one of the first steps to change how commercial and open source mathematical programming software will solve optimization problems.

Table of Contents

List of Figures	xi
List of Tables	xii
Acknowledgements	xiv
Dedication	xv
1 Introduction	1
1.1 Research Motivation and Objective	3
1.2 Research Contributions	4
1.2.1 Theoretical Contributions	4
1.2.2 Algorithmic Contributions	5
1.2.3 Computational Contributions	6
1.3 Dissertation Outline	7
2 An Overview of Advancements in Linear Programming	9
2.1 Advancements in a Simplex Framework	12
2.1.1 Update the Basis Inverse	13
2.1.2 Degeneracy	15
2.1.3 Pivoting Rules	17
2.1.4 Multiple Pivots	19
2.2 Advancements in an Interior Point Framework	20
2.2.1 Affine Scaling Methods	22
2.2.2 Barrier Methods	23
2.2.3 Path Following Algorithms	25

2.2.4	Infeasible Interior Point Algorithms	26
2.2.5	Multidimensional Search Techniques	27
3	Two Dimensional Searches in Linear Programming	30
3.1	One Dimensional vs. Two Dimensional Searches	31
3.2	The Slope Algorithm	35
3.3	The Ratio Algorithm	49
4	Two Dimensional Searches in a Simplex Framework	56
4.1	The Primal and Dual Simplex Frameworks	56
4.2	The Double Pivot Simplex Method	61
4.2.1	The Double Pivot Simplex Method and Degeneracy	69
4.2.2	Computational Study	72
4.2.2.1	Preliminary Implementation and Results	72
4.2.2.2	CPLEX's Implementation Details	76
4.2.2.3	CPLEX's Results and Analysis	77
4.3	The Double Pivot Dual Simplex Method	82
5	Two Dimensional Searches in an Interior Point Framework	89
5.1	One Dimensional Search Interior Point Algorithms	90
5.1.1	One Dimensional Search Primal Affine Scaling Interior Point Algorithm	90
5.1.2	One Dimensional Search Dual Affine Scaling Interior Point Algorithm	93
5.1.3	One Dimensional Search Primal Logarithmic Barrier Interior Point Algorithm	94
5.1.4	One Dimensional Search Dual Logarithmic Barrier Interior Point Algorithm	96
5.2	Two Dimensional Search Interior Point Algorithms	97
5.2.1	Principles for Determining Two Search Directions	97

5.2.2	Two Dimensional Search Primal Affine Scaling and Logarithmic Barrier Interior Point Algorithms	102
5.2.3	Two Dimensional Search Dual Affine Scaling and Logarithmic Barrier Interior Point Algorithms	106
5.3	Computational Study	108
5.3.1	Implementation Details	108
5.3.2	Instances and Computational Results	110
5.3.3	Analysis of Results	115
6	Conclusions and Future Research	121
6.1	Conclusions	121
6.2	Future Research	125
6.2.1	Future Simplex Framework Research	125
6.2.2	Future Interior Point Framework Research	126
	Bibliography	128

List of Figures

2.1	Graphical representation of a simplex and an interior point framework	11
3.1	Graphical representation of one and two dimensional searches in a simplex framework	32
3.2	Graphical representation of one and two dimensional searches in an interior point framework	33
3.3	Sample α_j for eight classes of constraints of an SA2VLP	37
3.4	Graphical representation of conditions <i>i</i>), <i>ii</i>), and <i>iii</i>) from Theorem 3.2.1 . .	40
3.5	Graphical representation of Example 3.2.1	48
4.1	Graphical representation of Example 4.2.2	71
5.1	Graphical representation of Example 5.2.1	101

List of Tables

4.1	Double pivot simplex tableau from Example 4.2.1	68
4.2	Improvement in the number of pivots and solution time of DPSM over SM (Dense Random LPs)	74
4.3	Improvement in the number of pivots and solution time of DPSM over SM (Sparse Random LPs, $\xi = 0.75$)	74
4.4	Improvement in the number of pivots and solution time of DPSM over SM (Sparse Random LPs, $\xi = 0.90$)	75
4.5	Improvement in the number of pivots of DPSM over SM and CPLEX's primal simplex algorithm (Netlib)	78
4.6	Improvement in the number of pivots of DPSM over SM and CPLEX's primal simplex algorithm (Netlib) - continued	79
4.7	Improvement in the number of pivots of DPSM over SM and CPLEX's primal simplex algorithm (MIPLIB)	80
4.8	Double pivot dual simplex tableau from Example 4.3.1	87
5.1	Number of constraints, variables, and nonzero elements in instances tested from Netlib	111
5.2	Improvement in the number of iterations and solution time of $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ over $1\text{DIM}_{\text{aff}}^{\text{P}}$ and $1\text{DIM}_{\text{log}}^{\text{P}}$	113
5.3	Improvement in the number of iterations and solution time of $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ over $1\text{DIM}_{\text{aff}}^{\text{P}}$ and $1\text{DIM}_{\text{log}}^{\text{P}}$ - continued	114
5.4	Improvement in the number of iterations and solution time of $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ over $1\text{DIM}_{\text{aff}}^{\text{D}}$ and $1\text{DIM}_{\text{log}}^{\text{D}}$	116

5.5	Improvement in the number of iterations and solution time of $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ over $1\text{DIM}_{\text{aff}}^{\text{D}}$ and $1\text{DIM}_{\text{log}}^{\text{D}}$ - continued	117
5.6	Average relative improvement in the objective function value per iteration $(\bar{\Delta})$	119

Acknowledgments

First of all, I would like to express my sincere gratitude to my advisor Dr. Todd Easton for his valuable guidance, understanding, and support throughout my career as a graduate student. He has inspired me with his optimization view to be a better researcher and a better teacher every single day. Dr. Easton, thank you for this incredible mentorship. It was a privilege to work with you in exciting research ideas, and also learn all the details about lecture based tutoring. I will never forget your advices.

I would like to thank my Ph.D. supervisory committee members Dr. Jessica Heier Stamm, Dr. Chih-Hang Wu, Dr. Nathan Albin, Dr. Dong Lin, and Dr. Christopher Vahl for their time and guidance during my graduate studies. Their insightful comments and suggestions strengthened this dissertation's research substantially.

I thank Dr. Bradley Kramer for his constant financial support and opportunity to develop my research and teaching skills. I also offer my sincere appreciation to all faculty and staff from the department of Industrial and Manufacturing Systems Engineering at Kansas State University for their support, knowledge, and assistance.

Last but not least, I would like to thank my family, friends, and colleagues who helped me during my academic and professional journey. Their encouragement, support, and willingness also made this work possible.

Dedication

This dissertation is dedicated to my parents, Jose Marcos and Maria Helena, and my brother, William, for their unconditional love, encouragement, and continuing support of my education.

Chapter 1

Introduction

Linear programming is an important class of optimization models. For decades, linear programs have improved complex systems from both the public and private sectors. Quickly finding an optimal solution to these mathematical models is vital to modern society, and numerous researchers have created results that decrease the computational time to solve linear programs. This dissertation's research advances this field by creating several two dimensional search algorithms that solve linear programs faster than the corresponding one dimensional search version.

Linear programming problems seek to either maximize or minimize an objective function denoted by a linear cost function in terms of decision variables. This objective function is restricted to a set of linear inequality and/or equality constraints. Typically, decision variables in a linear program may assume any nonnegative real value. When a linear program is optimally solved, the values attributed to the decision variables represent a solution that satisfies all constraints and provides the best possible objective function value.

Recent applications demonstrate the vital importance of linear programming. For instance, research in the logistics industry uses linear programs to optimize supply chain operations planning, intermodal transportation, and coordinated production and logistics planning (García et al., 2013; Kunnumkal et al., 2012; Lawrence and Burbidge, 2007; Spitter et al., 2005). Examples of linear programs applied to financial systems include asset

management, insurance, risk in portfolio optimization, and real option management of commodity storage ([Chalermkraivuth et al., 2005](#); [Mansini et al., 2007](#); [Nadarajah et al., 2015](#)).

Manufacturing businesses have also benefited from linear programming. Optimizing steel making and casting production, irregular strip packing, and scrap charge in steel production are some of the examples in this industry ([Gomes and Oliveira, 2006](#); [Rong and Lahdelma, 2008](#); [Tang et al., 2000](#)). In health care and medicine, linear programs have optimized high-dose-rate brachytherapy dose distributions, radiation therapy treatment planning, and diets ([Alterovitz et al., 2006](#); [Easton, 2009](#); [Lee et al., 2003](#); [Romeijn et al., 2006](#); [van Dooren, 2018](#)). Other than the applications in the private industry, linear programming has established governmental policies in the wood-fiber market, water in farming systems, and energy saving ([Bartolini et al., 2007](#); [Gautier et al., 2000](#); [Zhou and Ang, 2008](#)).

Besides these applications, [Williams \(2013\)](#) presents several other industries and problems where linear programming is applicable or accepted as a solution approach. Some of these applications include the oil and mining industries, agriculture, food, chemical, manpower planning, military, and energy systems. Other types of linear programming problems discussed by [Williams \(2013\)](#) are the network models, and some of them include the classical transportation, transshipment, assignment, minimum cost, maximum flow, and shortest path problems.

These practical applications are only a small sample of linear programming's ability to improve complex real world systems. However, the application and importance of this class of mathematical models goes beyond that. Linear programs are frequently the base to solve other classes of optimization problems as well. For instance, integer and mixed integer programming problems are typically solved by the branch and bound technique ([Land and Doig, 1960](#)). Each node of the branch and bound tree denotes a linear relaxation problem. That is, a linear program where the integrality restrictions are eliminated. Consequently, one could view the solution process of integer and mixed integer programs as solving numerous linear programs.

Another example involves applying successive linear programming algorithms to solve nonlinear optimization problems ([Palacios-Gomez et al., 1982](#)). These techniques, as the

name suggests, determine an approximate optimal solution to nonlinear programs via a sequence of linear programs. Other examples of optimization models where linear programming and its theory are vital include semidefinite programming (Wolkowicz et al., 2000) and complementary problems (Hu et al., 2012).

Either as a modeling approach or as a technique to solve other classes of mathematical programming models, linear programs play a critical role in optimization. Even though polynomial time methods exist to solve linear programming problems (Gondzio, 2012), a substantial amount of time may still be required to find an optimal solution to these models. This is because real world linear programming applications, and also other aforementioned classes of optimization problems where linear programs are used as a solution procedure, are frequently large and sparse.

Furthermore, the complexity of current real world applications has escalated substantially over the years. The amount of data available to model optimization problems has also drastically increased due to the discovery of many efficient data analytics techniques. All these and other facts combined frequently create optimization models that cannot be solved quickly. In this case, decision makers must sacrifice the quality of the optimization model in order to obtain solutions within a reasonable amount of time.

1.1 Research Motivation and Objective

Because of the importance of quickly solving linear programming problems, numerous advancements to state-of-the-art commercial and open source mathematical programming solvers are developed every year. These advancements are theoretical, algorithmic, and computational in nature. Consequently, finding novel techniques to solve linear programming models faster than the existing methods is the primary motivation of this dissertation.

The majority of these advancements can be categorized as one dimensional search techniques. For instance, the well-known simplex method (Dantzig, 1947) and the majority of interior point methods (Kojima et al., 1993) are one dimensional search techniques. That is, these algorithms solve linear programs by analyzing a one dimensional subspace problem

at each iteration. While there exist methods to solve linear programs by analyzing multi-dimensional subspace problems at each step (see Chapter 2), the amount of work done in this topic is either minimal or does not change this one dimensional search paradigm established in optimization methods. Consequently, both the lack of research and the lack of a broader view with respect to multidimensional search methods to solve linear programs is a motivation to this dissertation's research.

The objective of this dissertation is to advance the discipline of multidimensional searches in linear programming. More specifically, this dissertation focuses on two dimensional search techniques. The goal is to create novel two dimensional search algorithms that can solve linear programming problems faster than the corresponding one dimensional search version. Furthermore, this dissertation also aims to establish a new research area in optimization methods by motivating other researchers to study and pursue cutting-edge multidimensional search solution methods.

1.2 Research Contributions

This dissertation presents a variety of contributions that advance the topic of two dimensional searches in linear programming. These contributions are theoretical, algorithmic, and computational in nature. Details about each of these contributions are explained in the following sections.

1.2.1 Theoretical Contributions

This dissertation's theoretical contributions are demonstrated through a set of theorems, lemmas, and corollaries along with their corresponding proofs and arguments. The dissertation provides several theoretical results to guarantee that unboundedness, feasibility, and optimality conditions are satisfied when two dimensional searches are performed. Sufficient conditions are also presented to guarantee not only an optimal solution to these two dimensional searches, but also an optimal simplex basis. Moreover, this dissertation's research

proves that if all these conditions are implemented within a simplex framework, then an optimal solution to nondegenerate linear programming problems can be obtained within a finite number of steps.

This dissertation also argues the theoretical benefit of two dimensional search directions over a single one dimensional search direction. Under broad conditions, two dimensional searches are proved to improve the objective function value by at least as much as one dimensional searches. This theoretical result leads to one of the principles for determining two search directions in an interior point framework.

1.2.2 Algorithmic Contributions

This dissertation's algorithmic contributions are eight new two dimensional search algorithms for linear programming. These techniques are formalized from the theoretical foundation established in this dissertation's research. These algorithms and their corresponding contributions are as follows.

- i) The slope algorithm (SA) finds an optimal basis and an optimal solution to linear programs with two variables.
- ii) The ratio algorithm (RA) determines an optimal basis and an optimal solution of linear programs with two constraints.
- iii) The double pivot simplex method (DPSM) is a primal simplex framework algorithm where up to two basic leaving variables are exchanged with two nonbasic entering variables at each step. The slope algorithm performs two dimensional searches and defines the two basic leaving variables.
- iv) The double pivot dual simplex method (DPDSM) is a dual simplex framework algorithm where two variables can be exchanged simultaneously at each iteration. The ratio algorithm determines both nonbasic entering variables.

- v) The two dimensional search primal affine scaling interior point algorithm ($2\text{DIM}_{\text{aff}\perp}^{\text{P}}$) determines successive feasible and improved interior solutions by using two primal affine scaling search directions. The two search directions are derived from orthogonal vectors with respect to the objective function.
- vi) The two dimensional search primal logarithmic barrier interior point algorithm ($2\text{DIM}_{\text{log}\perp}^{\text{P}}$) considers two primal logarithmic barrier search directions at each iteration. Each of these two search directions are composed of a primal affine scaling search direction along with a centering component.
- vii) The two dimensional search dual affine scaling interior point algorithm ($2\text{DIM}_{\text{aff}\perp}^{\text{D}}$) has two dual affine scaling search directions at each step. These search directions are derived from orthogonal vectors with respect to the right-hand side vector.
- viii) The two dimensional search dual logarithmic barrier interior point algorithm ($2\text{DIM}_{\text{log}\perp}^{\text{D}}$) determines an improved feasible interior solution at each step by using two dual logarithmic barrier search directions.

1.2.3 Computational Contributions

This dissertation’s primary computational contribution is the computational results obtained with the implementation and comparison of the aforementioned two dimensional search algorithms to the corresponding one dimensional search version. Some of these algorithms were also compared to the implementation of CPLEX ([IBM, 2016](#)), a high performance mathematical programming solver. Computational experiments tested these methods with random linear programs and instances from well-known benchmark libraries such as Netlib ([Gay, 1985](#)) and MIPLIB ([Koch et al., 2011](#)).

The double pivot simplex method, when compared to the simplex method over random dense linear programs, improves the number of pivots and solution time by an average of approximately 17%. If random sparse instances are considered, this improvement becomes 30%. The number of pivots are also improved by nearly 41% when tested over benchmark

instances from Netlib and MIPLIB. Furthermore, the number of pivots are reduced by about 22%, on average, when tested with both sets of benchmark problems and compared to the primal simplex implementation of CPLEX.

The two dimensional search interior point algorithms, when contrasted to the corresponding one dimensional search version over benchmark linear programs from Netlib, improve the number of iterations by nearly 15%, on average. Computational results also show an overall improvement in solution time of approximately 12%. Additionally, the average relative improvement in objective function value per iteration is around 23%.

1.3 Dissertation Outline

The remainder of this dissertation is organized as follows. Chapter 2 describes some of the main contributions in the field of linear programming so that the reader can understand the difference between this dissertation's research and these other contributions. In addition, this chapter gives a chance for the reader visualize how this research advances the knowledge in linear programming. These advancements are discussed within the context of a simplex framework and an interior point framework.

Chapter 3 discusses two dimensional searches in linear programming. The chapter introduces the main difference between one and two dimensional search algorithms. Benefits and requirements for an efficient implementation of two dimensional searches are discussed. With the primary considerations established, the slope algorithm is presented. This technique finds an optimal basis and an optimal solution to linear programs with only two variables. Theoretical and algorithmic results are presented along with an example to show its implementation. Chapter 3 also presents the ratio algorithm, a method to find an optimal basis and an optimal solution to linear programs with exactly two constraints.

Chapter 4 demonstrates how the slope algorithm and the ratio algorithm can be implemented within a simplex framework. The chapter provides the mathematical concept of a simplex framework. The double pivot simplex method, an advancement to the simplex method, is presented. Theoretical results and an example support the understanding of this

new algorithm. The chapter also discusses the double pivot simplex method with respect to degeneracy. Computational experiments compare the double pivot simplex method with the simplex method, and also with the primal simplex implementation of CPLEX. Details on the implementation, and the comparison of results on random and benchmark linear programs are presented. The chapter concludes with the double pivot dual simplex method, a novel dual simplex framework where the ratio algorithm performs two dimensional searches. Theoretical results and an example assist the reader to understand this algorithm.

Chapter 5 extends the concept of two dimensional searches to an interior point framework. The chapter begins with the theoretical foundation of four classical one dimensional search interior point algorithms: primal affine scaling, dual affine scaling, primal logarithmic barrier, and dual logarithmic barrier. Principles for determining two search directions are discussed with the support of theoretical results and an example. The chapter presents novel two dimensional search primal and dual affine scaling and logarithmic barrier interior point algorithms. Implementation details and results of a computational study that compare each of these methods with the corresponding one dimensional search version are described.

Chapter 6 concludes the dissertation and provides a summary of the main results. This dissertation's research establishes a change of paradigm in optimization methods, which may result in important future research. Consequently, the chapter also discusses theoretical, algorithmic, and computational future research topics that may improve the efficiency of state-of-the-art commercial and open source mathematical programming solvers. If such a claim is confirmed, numerous industries in both the public and private sectors would benefit substantially.

Chapter 2

An Overview of Advancements in Linear Programming

Linear programming is one of the most important classes of optimization problems. Formally, a linear program (LP) with n variables and r constraints takes the form of:

$$\begin{aligned} & \text{maximize } z = c^T x \\ & \text{subject to } Ax \leq b \\ & x \geq 0, \end{aligned}$$

where n and r are positive integers, $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{r \times n}$, and $b \in \mathbb{R}^r$. Denote $N = \{1, 2, \dots, n\}$ to be the set of variable indices and $R = \{1, 2, \dots, r\}$ to be the set of constraint indices. The feasible region of an LP is denoted by $S = \{x \in \mathbb{R}_+^n : Ax \leq b\}$ and the optimal solution of an LP is (z^*, x^*) , where $x^* \in S$ and $z^* = c^T x^* \geq c^T x'$ for all $x' \in S$.

Every LP has a corresponding dual problem. Mathematically, define an LP's corresponding dual linear program (DLP) as:

$$\begin{aligned} & \text{minimize } z = b^T w \\ & \text{subject to } A^T w \geq c \\ & w \geq 0, \end{aligned}$$

where $w \in \mathbb{R}^r$. Observe that in this case, N becomes the set of constraint indices and R becomes the set of variable indices. The feasible region of a DLP can be defined as $T = \{w \in \mathbb{R}_+^r : A^T w \geq c\}$. Furthermore, the optimal solution of a DLP is (z^*, w^*) such that $w^* \in T$ and $z^* = b^T w^* \leq b^T w'$ for all $w' \in T$.

Some of the earliest works in linear programming were developed by Nobel Laureates [Kantorovich \(1939\)](#) and [Koopmans \(1949\)](#), although one could credit the origins of linear programming to [Fourier \(1826\)](#) and [de la Vallée Poussin \(1911\)](#) (see [Grattan-Guinness \(1970\)](#) and [Farebrother \(2006\)](#) for additional discussion). The work of [Kantorovich](#) describes mathematical methods to organize and plan production processes within an optimization perspective, while the paper from [Koopmans](#) focuses on the optimal utilization of transportation systems. Credit must also be given to [Leontief \(1936\)](#), who invented the input-output model in economics, and [von Neumann \(1928\)](#), who was critical to the development of duality theory in linear programming. These researchers all influenced [Dantzig \(1982, 1991\)](#), who can be considered the “father” of linear programming. This is because [Dantzig](#) developed the general formulation of linear programming problems and invented the simplex method ([Dantzig, 1947](#)), the first algorithm to optimally solve LPs. Additional details on the discovery of linear programming can also be found in [Dorfman \(1984\)](#).

Linear programming rapidly became popular not only in the academic community, but also in the public and private sectors. Currently, LPs dramatically impact society by improving complex systems from numerous industries (recall the examples of current applications discussed in [Chapter 1](#)). Consequently, it suffices to state that millions of LPs are solved every day, which helps create a more efficient world.

More quickly solving linear programming problems also became a priority over the decades. This is because solving LPs faster enable decision makers to more efficiently improve their systems. Additionally, LPs also play an important role in other classes of optimization problems such as integer and nonlinear programming. The practical importance of solving linear programs faster has motivated researchers to pursue numerous advancements. While an enormous number of advancements in linear programming have been developed throughout the years, all of them can be divided in two major categories: advancements within the

context of a simplex framework and advancements within the context of an interior point framework.

Figure 2.1(a) depicts the concept of a simplex framework. These algorithms begin from an initial solution. This solution corresponds to an extreme point (or a corner point), which is defined by the intersection of n or more constraints of the LP. These algorithms successively move to another solution, also defined by an extreme point, until an optimal solution is obtained. Define an extreme point in a simplex framework as a basic solution. A basic solution is denoted as a set with basic variables (values are nonzero assuming a nondegenerate basis) and nonbasic variables (values equal to zero). If a basic solution satisfies all the constraints of the LP, then this solution corresponds to a basic feasible solution. Observe that one should not view a simplex framework as moving from an extreme point to an adjacent extreme point. Instead, a simplex framework simply moves from one extreme point to another extreme point.

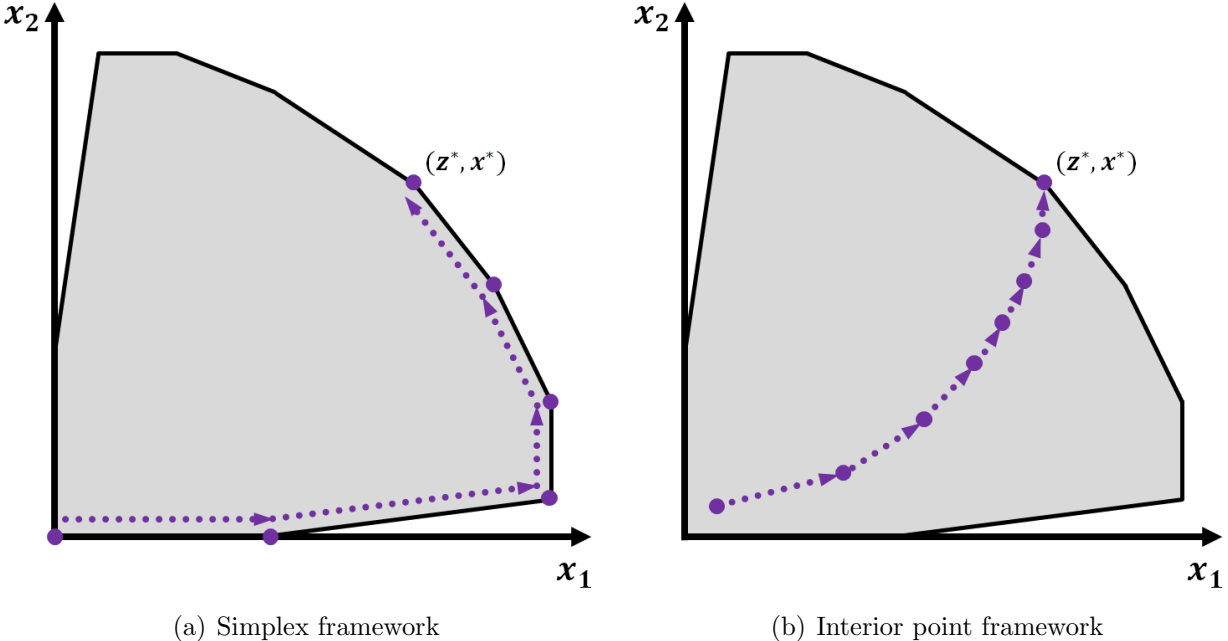


Figure 2.1: Graphical representation of a simplex and an interior point framework

Figure 2.1(b) presents the concept of an interior point framework. The primary premise is that these techniques begin from an initial solution that is contained in the interior of the LP’s feasible region. The algorithms then move to another solution, also contained

in the interior of the LP's feasible region, until they converge to an optimal solution. This movement typically occurs by following one or multiple search directions. Because an interior point framework requires every solution to be within the interior of the LP's feasible region, these methods never achieve an exact optimal solution. Instead, these techniques achieve a solution that is sufficiently close to an optimal solution.

Research performed on both frameworks are theoretical and computational in nature. This includes different strategies to improve the movement between solutions, techniques to avoid or skip stationary solutions, or methods to more quickly solve their subproblems or update solutions. The following sections describe some of these advancements within both frameworks since the late 1940s.

2.1 Advancements in a Simplex Framework

Even though other contributions to linear programming appeared before 1947, the simplex method created by [Dantzig \(1947\)](#) is what made linear programming possible. The simplex method is one of the most famous and important developments in science, and was recognized by the *Journal of Computing in Science & Engineering* as one of the top 10 algorithms of the 20th century ([Dongarra and Sullivan, 2000](#)). The simplex method has a worst-case running time of $O(2^n)$ ([Klee and Minty, 1972](#)). However, [Spielman and Teng \(2004\)](#) demonstrate that the simplex method may run in polynomial time when instances are randomly generated and slightly perturbed.

While the simplex method from [Dantzig](#) was used to solve LPs for many years, its dual version, first introduced by [Lemke \(1954\)](#), is mostly used in state-of-the-art commercial and open source solvers ([Koberstein and Suhl, 2007](#)). Not only the computational performance of the dual simplex method is shown to be superior to the simplex method ([Bixby, 2002](#)), but it also offer other benefits. For instance, the dual simplex method can be used to find an initial feasible solution instead of the well-known Big- M or two-phase simplex implementations ([Bazaraa et al., 2010](#)). This is because the dual simplex method starts super optimal and moves toward feasibility. Furthermore, the dual simplex method is used to solve the linear

relaxation problem of each child from a branch and bound tree (Land and Doig, 1960). The branch and bound technique typically solves integer and mixed integer programs (see other methods for integer programming in Marchand et al. (2002), Jünger et al. (2010), Vitor (2015), Bailey et al. (2018) Vitor and Easton (2016, 2019b)).

Currently, efficient implementations of the simplex and dual simplex method are available in commercial and open source solvers. However, the original implementation proposed by Dantzig was very inefficient and its practical application was not promising by that time (Orchard-Hays, 1990). This is because each iteration of the simplex method updated all the elements of the simplex tableau, including the reduced costs, right-hand side values, basis inverse, and other constraint values.

Numerous researchers developed theoretical and computational improvements to the simplex method. The author believes that two major advancements made the simplex method substantially more efficient over the last decades. First, the advancements in computer machinery throughout the years. Second, and probably the most critical one, is the effective methods to update the basis inverse. Other contributions such as techniques to handle degenerate LPs, pivoting rules, and methods to perform multiple pivots should also be considered.

Observe that the history of theoretical and computational advancements in LP solvers is relatively long. Trying to describe all these advancements in a single chapter of a dissertation is not only unfair to the researchers that made it possible, but also not applicable. The reader is encouraged to read the work from Orchard-Hays (1968, 1978a,b,c, 1984, 1990), Dantzig (1982, 1991), Shamir (1987), Bixby (1994, 2002, 2012), and Darby-Dowman and Wilson (2002) for additional details on some of these advancements.

2.1.1 Update the Basis Inverse

The simplex method and other simplex frameworks are completely dependent upon the inverse of the basis matrix, which is determined by the set of basic variables. In the simplex method, each iteration determines an entering nonbasic variable and a leaving basic variable.

Thus, a column of the basis matrix is replaced and the updated inverse matrix is required for the following iteration.

Observe that solving the inverse of the basis matrix requires the solution of a system of linear equations and its transpose. Even though some well-known techniques exist to obtain the basis inverse matrix using LU factorization via Gaussian elimination ([Elble and Sahinidis, 2012a](#); [Suhl and Suhl, 1990](#); [Tomlin, 1972](#)), substantial research has been performed over the last decades to more efficiently obtain this invertible matrix. Recently, [Elble and Sahinidis \(2012b\)](#) provided a detailed mathematical review of several of the methods discussed in this section, and their work is suggested for additional background information.

The first technique was proposed by [Dantzig and Orchard-Hays \(1954\)](#) and it follows the product form of the inverse. The core concept of this method is to represent the inverse of the basis matrix by the product of elementary matrices. Each elementary matrix corresponds to the identity matrix with one of its columns replaced by a set of elements determined by constraint values defined by the entering nonbasic variable and the leaving basic variable. These columns are well-known in the literature as the eta-vectors and the elementary matrices are referred to as the eta-matrices.

Each update in the basis inverse proposed by [Dantzig and Orchard-Hays](#) consists to the product of the current basis inverse with a set of eta-matrices. In contrast, [Bartels and Golub \(1969\)](#) derived a method where the factors from the lower and upper triangular matrices of the LU decomposition are updated using a similar idea as the eta matrix. To maintain the structure of the upper and lower triangular matrices, the [Bartels and Golub](#) update performs a column permutation and a series of elementary transformations. Observe that [Bartels \(1971\)](#) also provides an interesting discussion about the benefits obtained with respect to round-off errors when similar strategies are implemented.

Later on, [Forrest and Tomlin \(1972\)](#) proposed a technique that was mostly designed to solve sparse LPs. The [Forrest and Tomlin](#) update is still considered one of the most effective methods ([Huangfu and Hall, 2015](#)). Differently from the [Bartels and Golub](#) update, the [Forrest and Tomlin](#) update maintain triangularity by performing both column and row permutations before a series of elementary transformations. One can see that [Reid \(1982\)](#) also

uses both column and row permutations to restore triangularity, but additional permutations are considered at each step.

Other contributions in this topic also exist such as the update methods from [Gill et al. \(1984\)](#), [Fletcher and Matthews \(1984\)](#), [Tolla \(1986\)](#), and [Eldersveld and Saunders \(1992\)](#). Another widely used effective method was discovered by [Suhl and Suhl \(1993\)](#). This technique is a modification from the [Forrest and Tomlin](#) update in which the position of the permuted columns plays an important role. That is, the [Forrest and Tomlin](#) update permutes the income vector to the last position of the matrix while the [Suhl and Suhl](#) update permutes this vector to the last nonzero.

Recently, [Huangfu and Hall \(2015\)](#) proposed a set with three novel update techniques for practical sparse LPs. The alternate product form update and the middle product form update are advancements to the product form of the inverse from [Dantzig and Orchard-Hays](#). A third technique involves an advancement to the [Forrest and Tomlin](#) update. In this case, the so-called collective Forrest and Tomlin update allows multiple [Forrest and Tomlin](#) simultaneous updates to obtain the inverse of the basis matrix. Computational experiments on a high performance solver ([Huangfu and Hall, 2018](#)) indicates that the last technique is as efficient as performing [Forrest and Tomlin](#) updates independently.

2.1.2 Degeneracy

One of the biggest issues within a simplex framework is degeneracy ([Greenberg, 1986](#); [Megiddo, 1986](#)). Degeneracy occurs when a basic feasible solution has at least one basic variable that equals zero. Degenerate linear programming problems may force simplex framework algorithms to complete extra operations by performing multiple iterations at the same feasible solution. That is, these techniques may visit different basic feasible solutions with an identical objective function value. Additionally, degenerate LPs may cycle, which prohibits these algorithms from terminating. During several years, researchers have investigated and discovered techniques to prevent degeneracy and avoid cycling.

Some of these techniques involve pivoting strategies, and the first method was created by [Bland \(1977\)](#). When [Bland's](#) rule is implemented, the simplex method is guaranteed to terminate within a finite number of steps. This rule is simple and consists of selecting an improving entering nonbasic variable with the lowest index with respect to the columns, and a leaving basic variable that satisfies the minimum ratio test and has the lowest index with respect to the rows. Observe that [Bland](#) also proposes a recursive rule in addition to his index rule.

Similarly, [Fukuda \(1982\)](#) also describes a pivoting rule that theoretically avoids cycling ([Clausen, 1987](#)). This technique is mostly known as the Edmonds-Fukuda rule, and involves selecting an entering variable, also following index rules, from a set that may contain both basic and nonbasic variables. On the theoretical survey presented by [Terlaky and Zhang \(1993\)](#), the paper discusses the relation between the “last in first out” and the “most often selected” rules to finite pivoting rules, including the techniques from [Bland](#) and [Fukuda](#), and also other pivoting rules such as the finite crisscross method ([Terlaky, 1987](#)), Jensen’s general recursion rule ([Jensen, 1985](#)), and the lexicographic method ([Todd, 1985](#)).

Another class of techniques to handle degenerate LPs is the so-called perturbation method. The core concept of this strategy is to slightly perturb the right-hand side values of the basic variables that are degenerate so that they become nonzero. One of the greatest benefits with this strategy is that a perturbation is only required after a degenerate basis has been found. On the other hand, this technique increases the number of basic feasible solutions, and these solutions have the potential to improve the objective function value by only a small amount ([Elhallaoui et al., 2010a](#)). Some of these perturbation methods were developed by researchers such as [Charnes \(1952\)](#), [Wolfe \(1963\)](#), [Ryan and Osborne \(1988\)](#), and [Pan \(1999, 2000\)](#).

An approach suggested by [Pan \(1998\)](#) involves the idea of a reduced basis. From a basic feasible solution, the method proposed by [Pan](#) identifies a set of degenerate basic variables. Columns and rows associated with these variables are removed so that a reduced basis matrix is obtained. This subproblem is solved and its dual values help determine the reduced costs of all the variables from the original problem. Observe that [Pan](#) also uses the

idea of compatible and incompatible variables ([Raymond et al., 2010](#)) to remove variables from the original problem or add constraints to the subproblem.

Other researchers expanded the method of [Pan](#). For instance, [Elhallaoui et al. \(2005\)](#) presents a dynamic constraint aggregation method to obtain subproblems associated with degenerate basic variables. Similarly, [Elhallaoui et al. \(2008\)](#) improves the previous technique by reducing the size of both the original problem and the subproblem. Further improvements to both methods are also presented ([Elhallaoui et al., 2010b](#)). Improved simplex primal algorithms for degenerate LPs were also derived from the combination of previous discussed methods ([Elhallaoui et al., 2010a](#); [Raymond et al., 2010](#)).

2.1.3 Pivoting Rules

The simplex method, as initially created by [Dantzig \(1947\)](#), has a simple and straightforward pivoting rule. That is, an improving entering nonbasic variable is selected as the variable with the most negative reduced cost, while the leaving basic variable is determined by the minimum ratio test (usually selected arbitrarily when a tie is detected). Pivoting rules have a high correlation to the number of iterations performed by the simplex method and other simplex frameworks, and consequently to the performance of these algorithms. Research have investigated several pivoting strategies to improve the simplex method and other simplex framework algorithms.

Overall, there are three types of pivoting rules: finite, full pricing, and partial pricing. The first one is related to the termination of the algorithm in a finite number of steps, and usually avoids or resolve degeneracy. Full pricing denotes techniques that evaluate all nonbasic variables at each step to find an entering variable. In contrast, partial pricing only evaluates a set of nonbasic variables at each iteration. [Terlaky and Zhang \(1993\)](#) discuss several pivoting strategies discovered until the early 1990s such as finite rules, pivoting strategies related to parametric programming, and pivoting rules inspired by interior point methods. Recent surveys on pivoting rules also exist ([Pan, 2014](#); [Ploskas and Samaras, 2017](#)).

Observe that the strategies discussed in Chapter 2.1.2 to handle degenerate linear programming problems (Bland, 1977; Fukuda, 1982; Jensen, 1985; Terlaky, 1987; Todd, 1985) are all finite pivoting rules. Other strategies also include for example the admissible pivot method from Lim and Park (2004), the s -monotone index selection pivoting rule from Csizmadia et al. (2012), the improvement to Bland’s rule proposed by Liao (2013), and the optimal pivot rule from Etoa (2016).

Full pricing rules have also been investigated. One can see that Dantzig pivoting technique can be classified as full pricing. Some commercial and open source solvers have widely used the steepest-edge (Forrest and Goldfarb, 1992; Goldfarb and Reid, 1977) and Devex (Harris, 1973) pivoting rules to solve LPs (Bixby, 2002; Pan, 2010). The steepest-edge rule selects an improving entering nonbasic variable by analyzing the normalized vector of reduced costs formed by the set of candidate entering nonbasic variables. The Devex rule is similar to the steepest-edge pivoting rule, but this strategy considers an approximation of the normalized vector of reduced costs.

One technique derived from the idea of normalized reduced costs is the largest-distance pivoting rule (Pan, 2008a). Even though this rule is applied in the primal space, its interpretation is easily obtained when analyzing the dual space. The largest-distance pivoting rule considers “fixed normalized factors”, and computational experiments show an improved performance over the Devex rule.

When it comes to partial pricing strategies, Pan (2008b) proposes a nested pricing pivoting rule. Given a subset of all reduced costs, the method selects one that is sufficiently negative. If none exists, then the algorithm redefines the subset of reduced costs until one sufficiently negative reduced cost is found or optimality is proven. The paper describes a nested Dantzig rule, a nested steepest-edge rule, and a nested Devex rule. Observe that Pan (2010) reports efficient computational results when the largest-distance and nested pricing pivoting rules are combined. Other partial pricing techniques also exist (Benichou et al., 1977; Greenberg, 1978; Maros, 2003a,b).

2.1.4 Multiple Pivots

Practical LPs are usually large and sometimes complex to solve. Solution times to solve these LPs may be relatively long, even though polynomial time algorithms exist to solve this critical class of optimization problems ([Gondzio, 2012](#)). A common technique to decrease the solution time of LPs, which highly motivated this research, is to solve the problem on a subset of variables and/or constraints. An optimal basis from this subproblem is used to identify a new subset of variables and/or constraints, and the process repeats until an optimal basis to a subproblem identifies an optimal basis to the original instance. This general methodology when utilized within a simplex framework is called in this dissertation as multiple pivot techniques. Some of the methods developed throughout the years that follow this strategy are decomposition methods, column generation, and block pivots.

Decomposition methods were first introduced by [Dantzig and Wolfe \(1960, 1961\)](#) and these techniques are sometimes referred to as the decomposition principle. The core concept of decomposition methods is the partitioning of the original problem into two parts. The first part consists of a general structure while the second part denotes a special structure ([Bazaraa et al., 2010](#)). The general structure usually has constraints that cannot be decomposed, while the special structure contains those constraints that follow a similar framework and its solution can be easily obtained when decomposed. Each structure is successively solved as an LP, and necessary information is exchanged between both LPs until an optimal solution to the original instance is finally obtained.

Other than the classical [Dantzig and Wolfe](#) decomposition method, [Benders \(1962\)](#) proposed another classical decomposition framework that can be applied to linear programming, and is often used in stochastic programming. Observe that the idea of a reduced basis can also be considered a decomposition method ([Elhallaoui et al., 2005, 2008, 2010a,b](#); [Pan, 1998](#); [Raymond et al., 2010](#)). Recently, [Gauthier et al. \(2018\)](#) also created a vector space decomposition framework that is guided by dual optimality considerations. Other decomposition methods can also be found in [Conejo et al. \(2006\)](#).

Another multiple pivot technique is column generation. This method was initially proposed by [Ford and Fulkerson \(1958\)](#). The goal is to solve LPs that have numerous variables, and the number of variables is substantially greater than the number of constraints ([Nemhauser, 2012](#)). Column generation begins with a subset of these variables (usually enough variables to form an initial basis). This subproblem is solved and the algorithm analyzes if adding one of the excluded variables (new column) improves the current solution. This process is repeated until an optimal solution to the original LP is achieved. Observe that column generation techniques are sometimes applied to decomposition methods to handle the interaction between the general and special LPs. Column generation has been used for example to solve cutting-stock problems ([Gilmore and Gomory, 1961, 1963](#)), shipping scheduling problems ([Appelgren, 1969](#)), and machine learning related topics ([Demiriz et al., 2002](#)). Column generation has also widely been used in integer and mixed integer programming ([Lübbecke and Desrosiers, 2005; Nemhauser, 2012](#)).

Block pivot techniques are another class of multiple pivots. While the simplex method exchanges exactly one basic variable with one nonbasic variable at each iteration, block pivot methods exchange a set of these variables at each step. [Bazaraa et al. \(2010\)](#) provide three conditions that must be satisfied to have successful block pivots: basis structure maintained, feasibility, and no worse objective function values. [Bazaraa et al.](#) argue that block pivots are not often explored because checking for feasibility is either difficult or time consuming. The reader will see later in this dissertation that some of the algorithms created by this research are block pivot techniques, and correctly checking for feasibility can be quickly performed. Block pivots are also discussed by [Howard \(1960\)](#), [Padberg \(1999\)](#), and [Ye \(2011\)](#).

2.2 Advancements in an Interior Point Framework

Even though simplex framework algorithms can efficiently solve linear programming problems, no polynomial time pivoting rule is known to date, and some researchers believe that one probably does not exist ([Gondzio, 2012](#)). This fact, associated with the importance of solving LPs faster, motivated extensive research to various other techniques. This investiga-

tion resulted in the discovery of numerous algorithms within the context of an interior point framework. Interior point methods are one of the most important breakthroughs in linear programming, and were a complete change of paradigm to the field.

Substantial interior point algorithm results for linear programming appeared in the early 1980s, but credit must be given to [Frisch \(1955\)](#), [Carroll \(1961\)](#), and [Fiacco and McCormick \(1968\)](#), who developed critical interior point algorithm results for nonlinear programming. Currently, there still exists a debate as to whether or not interior point algorithms are computationally faster than the simplex method ([Bertsimas and Tsitsiklis, 1997](#); [Gondzio, 2012](#); [Illés and Terlaky, 2002](#); [Terlaky and Zhang, 1993](#); [Todd, 2002](#)). Based on these discussions, interior point methods frequently solve large and sparse LPs faster than the simplex method.

The first promising interior point algorithm for LPs was proposed by [Khachiyan \(1980\)](#). This method, mostly known as the [Khachiyan](#)'s ellipsoid interior point algorithm, determines a sequence of ellipsoids which contains a solution that satisfy the LP's feasible region. When one of these ellipsoids has its center violating at least one of the constraints, then the algorithm constructs a smaller ellipsoid containing this constraint and the process is repeated. [Khachiyan](#)'s ellipsoid interior point algorithm was the first known polynomial time method to solve LPs. However, this techniques was proved to be ineffective in practice because of its high dependency on the input data ([Goldfarb and Todd, 1989](#)).

The first interior point algorithm that was efficient in practice was created by [Karmarkar \(1984\)](#). Similar to [Khachiyan](#)'s technique, [Karmarkar](#)'s projective interior point algorithm can solve LPs in polynomial time. From a feasible interior solution, this method computes a feasible and improving search direction at each step, and moves along this search direction to an improved feasible interior solution. In this case, [Karmarkar](#)'s technique computes search directions by using a projective transformation to place feasible interior solutions near the center of the LP's feasible region in the transformed space.

Given the impressive results with [Karmarkar](#)'s method, numerous other researchers have studied and proposed alternate interior point algorithms. Some of the major results can be classified as affine scaling methods, barrier methods, path following algorithms, infeasible interior point algorithms, and multidimensional search techniques. Additional details about

the history or background information of interior point methods is found in the well-known papers of [Lustig et al. \(1994a\)](#), [Shanno \(2012\)](#), and [Gondzio \(2012\)](#). Other algorithmic and theoretical results can also be found in [Fang and Puthenpura \(1993\)](#), [Bazaraa et al. \(2010\)](#), and [Vanderbei \(2014\)](#).

2.2.1 Affine Scaling Methods

Affine scaling interior point algorithms are similar to [Karmarkar's](#) projective interior point algorithm. The main difference between both techniques is on the computation of search directions. That is, affine scaling methods place feasible interior solutions near the center of the LP's feasible region by using an affine scaling transformation instead of the projective transformation from [Karmarkar's](#) algorithm. Another difference is that affine scaling methods can be directly applied to LPs in standard form, while [Karmarkar's](#) algorithm requires additional transformations to the problem. On the other hand, theoretical convergence results are usually easier to obtain in projective algorithms than in affine scaling techniques ([Vanderbei and Lagarias, 1990](#)).

The first results on affine scaling methods were published by [Dikin \(1967, 1974\)](#). However, [Dikin's](#) work was noticed only after [Karmarkar](#) created its projective algorithm. From the work of [Karmarkar](#), other researchers developed critical results to affine scaling interior point algorithms. For instance, [Barnes \(1986\)](#), [Vanderbei et al. \(1986\)](#), and [Adler et al. \(1989\)](#) independently developed algorithmic and theoretical results on affine scaling methods. Only after a few years of these discoveries, [Bayer and Lagarias \(1989a\)](#) and [Vanderbei and Lagarias \(1990\)](#) more clearly presented the idea of [Dikin](#), including a detailed review of [Dikin's](#) theoretical convergence result. In fact, several other researchers developed convergence results to affine scaling methods ([Güler and Ye, 1993](#); [Monteiro et al., 1993](#); [Saigal, 1996a](#); [Tseng and Luo, 1992](#); [Tsuchiya, 1992](#)).

Affine scaling methods have three basic variants: primal, dual, and primal-dual. The primal affine scaling interior point algorithm optimizes over the primal form of LPs (see Chapter [5.1.1](#) for the mostly used formulation). The algorithm begins with a primal feasible

interior solution and moves toward optimality while maintaining primal feasibility. The dual affine scaling interior point algorithm considers the a dual LP formulation (see Chapter 5.1.2). From an initial dual feasible interior solution, the algorithm maintains dual feasibility during the entire solution process while it achieves optimality. Computationally speaking, the dual affine scaling method tends to converge faster than the primal affine scaling technique, and it is less sensitive to numerical errors (Fang and Puthenpura, 1993). Other variants of the classical primal and dual affine scaling methods have also been developed (Hager and Zhang, 2014; Liao, 2014; Nayak et al., 2012; Pan, 2013; Saigal, 1996b).

The primal-dual affine scaling interior point algorithm optimizes over the primal and dual LP formulations simultaneously. The algorithm uses a logarithmic barrier function to accomplish all three Karush-Kuhn-Tucker (KKT) conditions (primal feasibility, dual feasibility, and complementary slackness) during the optimality process. Because of this additional feature, and other such variants explored throughout the years, the primal-dual affine scaling interior point algorithm is usually not viewed as an affine scaling method. Instead, this techniques is simply categorized in the broad class of primal-dual interior point methods.

The first primal-dual framework was proposed by Megiddo (1989). Furthermore, both Kojima et al. (1989) and Monteiro and Adler (1989) discovered polynomial time primal-dual algorithms for LPs. The history of primal-dual methods is relatively long, and the work of Wright (1997) is suggested for a complete study. Recent research on primal-dual methods also exists (Bai et al., 2008; Baryamureeba and Steihaug, 2006; Benson and Shanno, 2007; Edlund et al., 2009; Takács and Darvay, 2018).

2.2.2 Barrier Methods

Barrier interior point algorithms utilize some type of barrier function to push successive feasible interior solutions away from the boundary of the LP's feasible region. These algorithms have a barrier function associated with a weight parameter included in the objective function. The basic idea is that this additional term must approach infinity (or negative infinity depending on the optimization direction) as the solution approaches the boundary of the

LP's feasible region. Consequently, the barrier function prevents solutions from becoming sufficiently close or violating constraints. One can see that barrier interior point algorithms convert a constrained LP to an unconstrained problem.

Observe that the algorithms from [Megiddo \(1989\)](#), [Kojima et al. \(1989\)](#), and [Monteiro and Adler \(1989\)](#) can also be classified as barrier interior point algorithms, and are an advancement to the initial research in logarithmic barrier methods. Overall, the majority of barrier techniques use a logarithmic function as the barrier term. This idea comes from nonlinear programming ([Fiacco and McCormick, 1968](#); [Frisch, 1955](#)), but was first explored in linear programming by [Gill et al. \(1986\)](#) and [Gay \(1987\)](#). In fact, [Gill et al.](#) show that the primal logarithmic search direction and [Karmarkar's](#) search direction are identical for some positive weight parameter. Later on, [Gonzaga \(1989a, 1992\)](#) and [Shanno and Bagchi \(1990\)](#) showed that [Karmarkar's](#) projective algorithm and other such variants ([Renegar, 1988](#)) can be easily represented as logarithmic barrier methods.

The work of [Nesterov and Nemirovskii \(1994\)](#) discusses the benefits of logarithmic functions to barrier methods. [Nesterov and Nemirovskii](#) introduce the concept of self-concordant functions and their properties. These type of functions, which are preserved under affine transformations, describe a suitable relationship between the second and third derivatives of a nonlinear function with respect to the convergence of Newton's method. In this case, [Nesterov and Nemirovskii](#) show that logarithmic functions are self-concordant. Other research on interior point methods derived from logarithmic barrier functions also exist ([El Ghami et al., 2008](#); [Jansen et al., 1994](#); [Komodakis and Pesquet, 2015](#)).

Even though the majority of results in barrier methods involve logarithmic functions, other researchers have developed different approaches to this class of interior point methods. For instance, researchers such as [Bai et al. \(2003, 2004\)](#), [El Ghami and Roos \(2008\)](#), [El Ghami et al. \(2009\)](#), and [Darvay and Takács \(2018\)](#) all provide studies and/or barrier interior point algorithms based on kernel functions. Similarly, [El Ghami et al. \(2012\)](#) and [Bouafia et al. \(2016\)](#) consider kernel functions with a trigonometric barrier term, and both show improvements in the iteration bound for large and small updates.

2.2.3 Path Following Algorithms

Numerous interior point algorithms can be categorized in the class of so-called path following methods. These techniques limit successive feasible interior solutions to a neighborhood defined by a path, which is usually referred to as the central path (Bayer and Lagarias, 1989a,b; Megiddo, 1989). The main idea is to center search directions so that they more closely follow the trajectory of the central path.

Common path following methods typically include a logarithmic barrier function so that the neighborhood defined by the central path does not contain solutions close to the boundary of the LP's feasible region. Observe that the aforementioned initial primal-dual frameworks developed by Megiddo (1989), Kojima et al. (1989), and Monteiro and Adler (1989) are also path following methods. Path following algorithms are dependent on a centering parameter that ranges between 0 and 1. Consequently, these methods are usually categorized as short-step, long-step, and predictor-corrector.

Short-step path following methods choose a centering parameter close to 1. In this case, the movement between solutions stays within the neighborhood defined by the central path. While feasibility is guaranteed and solutions stay away from the boundary of the LP's feasible region, only small progress is made toward the direction of the optimal solution. Consequently, short-step path following methods perform short movements until optimality is achieved, as its name suggests. This strategy was not deeply explored in the literature, but some studies are included in Renegar (1988), Gonzaga (1989b), Gonzaga (1992), and Shaw and Goldfarb (1994).

Long-step path following algorithms determine a centering parameter farther from 1 and closer to 0. Consequently, larger steps toward the direction of the optimal solution are performed, but solutions usually stay away from the central path and its neighborhood. Because long-step techniques perform less conservative movements, its iteration bound is smaller, and long-step path following methods are typically preferred rather than short-step methods. Examples of research developed using long-step strategies include the work of Gonzaga (1991b,c), Anstreicher and Bosch (1992), Anstreicher (1996), and Wu (1998).

The most useful and explored path following implementation involves the predictor-corrector strategy. Predictor-corrector path following algorithms balance both the central path and the direction of the optimal solution. That is, the centering parameter is alternated between 0 and 1 at successive iterations. One can view an iteration where the centering parameter equals 0 as the predictor step, while the other is denoted as the corrector step. The first predictor-corrector algorithm was proposed by [Mehrotra \(1992\)](#), which was further explored by [Lustig et al. \(1992, 1994b\)](#). Other predictor-corrector path following methods also exist ([Colombo and Gondzio, 2008](#); [Gondzio, 1996](#); [Haeberly et al., 1999](#); [Ma and Liu, 2017](#); [Winternitz et al., 2012](#); [Yang, 2013](#)).

2.2.4 Infeasible Interior Point Algorithms

Several proposed interior point methods require an initial feasible interior solution, which is not always simple to obtain. Even though modifying the problem being solved may lead to such a solution ([Kojima et al., 1989](#); [Megiddo, 1989](#); [Monteiro and Adler, 1989](#)), this modification increases the size of the instance and may result into numerical issues ([Lustig, 1990](#); [Lustig et al., 1991](#); [McShane et al., 1989](#)). Infeasible interior point methods, as the name suggests, do not require an initial strictly feasible solution. In this case, the algorithm can solve an LP in its standard form and no artificial problem is necessary. Basically, all that is needed are strictly positive decision and slack variables. Search directions in an infeasible interior point method are modified such that the algorithm moves toward feasibility and consequently, to an optimal solution. Currently, infeasible primal-dual methods are considered one of the computationally fastest algorithms to solve linear programming problems ([Gondzio, 2012](#)).

The first infeasible primal dual interior point algorithm was presented by [Kojima et al. \(1993\)](#), who also proved its global convergence. Primal, dual, and slack search directions are determined by solving a system of linear equations that are dependent on the residuals with respect to the current solution. One can view the residuals as the “amount of infeasibility” in the primal and dual spaces given by the current solution. [Kojima et al.](#) also defines a

rule to calculate distinct step sizes in both spaces, which allows primal and dual/slack search directions to maximize the movement in both spaces. Additionally, [Zhang \(1994\)](#) provided a polynomial time long-step path following interior point method from the work developed by [Lustig et al. \(1991\)](#).

Other variants of infeasible primal dual interior point methods can be found in the papers of [Potra \(1994\)](#), [Ye et al. \(1994\)](#), [Freund \(1996\)](#), and [Potra \(1996\)](#). Recently, other researchers have also proposed infeasible interior point methods for LPs. For instance, [Gu et al. \(2010\)](#), [Mansouri and Zangiabadi \(2013\)](#), [Ahmadi et al. \(2014\)](#), [Mansouri et al. \(2015\)](#), and [Roos \(2015\)](#) all created improved full Newton step infeasible interior point methods. For each of these algorithms, one feasibility step and several centering steps are performed at each iteration. Furthermore, [Asadi and Roos \(2016\)](#) and [Liu et al. \(2011\)](#) design infeasible large neighborhood techniques inspired by the full Newton step methods and second order corrector algorithms. Another example is the arc-search infeasible interior point algorithms from [Yang et al. \(2016\)](#) and [Yang \(2018\)](#), which have a wider infeasible central path neighborhood than the existing methods.

2.2.5 Multidimensional Search Techniques

The core concept of multidimensional search interior point algorithms involves the idea of two or more search directions in contrast to a single search direction (see Chapter 3.1 for additional details). In this case, each step of an interior point algorithm searches over a multidimensional subspace instead of a line. While algorithms with a single search direction have been thoroughly explored over the decades, only a minimal amount of work has been done on algorithms with multiple search directions.

[Karmarkar and Ramakrishnan \(1985\)](#) gave a talk in the ORSA/TIMS Joint National Meeting, and were the first to discuss the concept and potential benefits of multidimensional search interior point algorithms for LPs. Later on, [Gonzaga \(1991a\)](#) presented the first preliminary theoretical results on multidimensional search procedures. Gonzaga compares the trust region of many one dimensional search methods and provides an expanded trust

region when two dimensional searches are performed over a two dimensional region. Gonzaga also discusses the use of cost improvement search directions along with recentering search directions on a two dimensional search framework.

The work of [Karmarkar and Ramakrishnan](#) and [Gonzaga](#) provided a transformative change of paradigm, but no algorithmic results were presented. The first promising results on two dimensional searches appeared in the work of [Boggs et al. \(1989\)](#), which developed the first two dimensional search interior point algorithm to solve LPs. From a feasible interior solution, their technique defines one of its search directions as a dual affine direction. The other search direction is determined by either the so called rank-one update direction or a Newton recentering direction. The rank-one update search direction is derived by the first constraint found by the dual affine search direction. [Boggs et al.](#) argue that the rank-one update search direction is either improving or dominated by the dual affine search direction, which guarantees an improving movement at each iteration. The Newton recentering search direction is implemented only at the final iterations when the rank-one update search direction is less productive due to the large number of active constraints.

The method of [Boggs et al.](#) was implemented only at Phase 2 (when an initial feasible interior solution has been already obtained) and two dimensional subspace linear programs were solved with the general implementation of the simplex method. The algorithm was tested on a small subset of benchmark linear programs against the classical dual affine scaling interior point algorithm. The paper of [Boggs et al.](#) does not report any solution times, but claim an improvement in the number of iterations.

The above two dimensional search interior point algorithm was extended to a three dimensional search version by [Domich et al. \(1991\)](#). The method is similar to the one developed by [Boggs et al.](#), and a third search direction is derived from [Huard's](#) method of centers ([Huard, 1967](#)). Computational experiments showed not only an improvement in the number of iterations but also an improvement in solution times. Other multidimensional search techniques not related to interior point methods or linear programming include the work of [Megiddo \(1984\)](#), [Dyer \(1986\)](#), [Dyer and Frieze \(1989\)](#), [Seidel \(1991\)](#), [Agarwala and Fernández-Baca \(1996\)](#), and [Dyer and Sen \(2000\)](#).

To the best of the author's knowledge, the algorithms of [Boggs et al.](#) and [Domich et al.](#) are the only multidimensional search interior point algorithms developed for linear programming. Even though both showed theoretical and practical benefits on this new advancement to interior point methods, this topic has not been explored by other researchers. Furthermore, all developed multiple pivot techniques discussed in [Chapter 2.1.4](#) were created more from an application perspective rather than a fundamental change of paradigm. That is, the majority of these methods more efficiently solve specific applications or classes of problems that are hard to solve using the classical techniques. However, if applied to solve any other general problem, these methods may not necessarily be as effective as the classical one dimensional search algorithms.

As discussed in [Chapter 1.1](#), this lack of research along with the lack of a broader view with respect to multidimensional search methods is one of the motivations to this dissertation's research. The following chapters provide novel multidimensional search techniques within both a simplex and interior point frameworks.

Chapter 3

Two Dimensional Searches in Linear Programming

This chapter presents newly developed two dimensional search techniques for linear programming. Chapter 3.1 discusses the concept of one dimensional and two dimensional searches implemented to both a simplex and an interior point framework. Chapter 3.2 describes the slope algorithm, a technique to find both an optimal basis and an optimal solution to LPs with only two variables. This method is also expanded in Chapter 3.3 to create the ratio algorithm, a technique to determine an optimal basis and an optimal solution to LPs with only two constraints. These methods are the foundation for the two dimensional search algorithms presented in Chapters 4 and 5.

The primary content of this chapter is based on the journal paper, “The Double Pivot Simplex Method”, published in *Mathematical Methods of Operations Research* (Vitor and Easton, 2018b) and the conference paper, “The Ratio Algorithm to Solve the Optimal Basis of Two Constraint Linear Programs”, published in the *Proceedings of the 2018 IISE Annual Conference* (Vitor, 2018). Some fundamental insights are also retrieved from the manuscript, “Projected Orthogonal Vectors in Two Dimensional Search Interior Point Algorithms for Linear Programming”, which is currently under peer-review (Vitor and Easton, 2019a).

3.1 One Dimensional vs. Two Dimensional Searches

Numerous techniques, as discussed in Chapter 2, have been developed to solve LPs. These methods are iterative algorithms that solve linear programming problems using a master-subproblem strategy. To solve the original LP (master problem), the algorithms successively move from one solution to another solution by solving a smaller LP (subproblem) at each iteration. This strategy is commonly used not only to solve LPs, but also to solve many other classes of optimization problems such as integer and nonlinear programming.

The vast majority of these linear programming algorithms can be viewed as one dimensional search techniques. That is, these methods successively improve the objective function value by solving a one dimensional subspace linear program at each step. Given a solution, the subproblem determines how far to move along a single feasible improving search direction to a new solution.

In contrast, two dimensional search algorithms solve a two dimensional subspace linear program at each iteration. This subproblem requires two search directions instead of one. In this case, at least one of them must be a feasible improving search direction. The subproblem determines the next solution by searching over a two dimensional subspace defined by the nonnegative linear combination of both search directions and the LP's feasible region.

Figure 3.1(a) geometrically demonstrates, in a 3-dimensional LP sketch, the concept of one dimensional searches when applied to a simplex framework. From a basic feasible solution x_{1D}^k , a feasible improving search direction d_{1D}^k is selected. This search direction, when intersected with the LP's feasible region, creates a one dimensional subspace defined by the line from x_{1D}^k to x_{1D}^{k+1*} . Optimally solving $x_{1D}^k + \lambda_{1D}^k d_{1D}^k$ along with $\lambda_{1D}^k \geq 0$ forms a one dimensional subspace linear program. An optimal solution to this subproblem, λ_{1D}^{k*} , maximizes the movement from x_{1D}^k along search direction d_{1D}^k to x_{1D}^{k+1*} . This new and improved basic feasible solution is defined as $x_{1D}^{k+1*} = x_{1D}^k + \lambda_{1D}^{k*} d_{1D}^k$.

Figure 3.1(b) geometrically presents the primary concept of two dimensional searches in a simplex framework. From a basic feasible solution x_{2D}^k , two feasible search directions $d_{2D}^{k'}$ and $d_{2D}^{k''}$ are selected where at least one of them must be improving. Observe that a

nonnegative linear combination of $d_{2D}^{k'}$ and $d_{2D}^{k''}$ defines a plane. This plane, when intersected with the LP's feasible region, creates a two dimensional subspace defined in Figure 3.2(b) by Π . Optimally solving $x_{2D}^k + \lambda_{2D}^{k'} d_{2D}^{k'} + \lambda_{2D}^{k''} d_{2D}^{k''}$ with $\lambda_{2D}^{k'} \geq 0$ and $\lambda_{2D}^{k''} \geq 0$ creates a two dimensional subspace linear program. An optimal solution to this subproblem, $(\lambda_{2D}^{k'*}, \lambda_{2D}^{k''*})$, determines the new and improved basic feasible solution $x_{2D}^{k+1*} = x_{2D}^k + (\lambda_{2D}^{k'*} d_{2D}^{k'} + \lambda_{2D}^{k''*} d_{2D}^{k''})$.

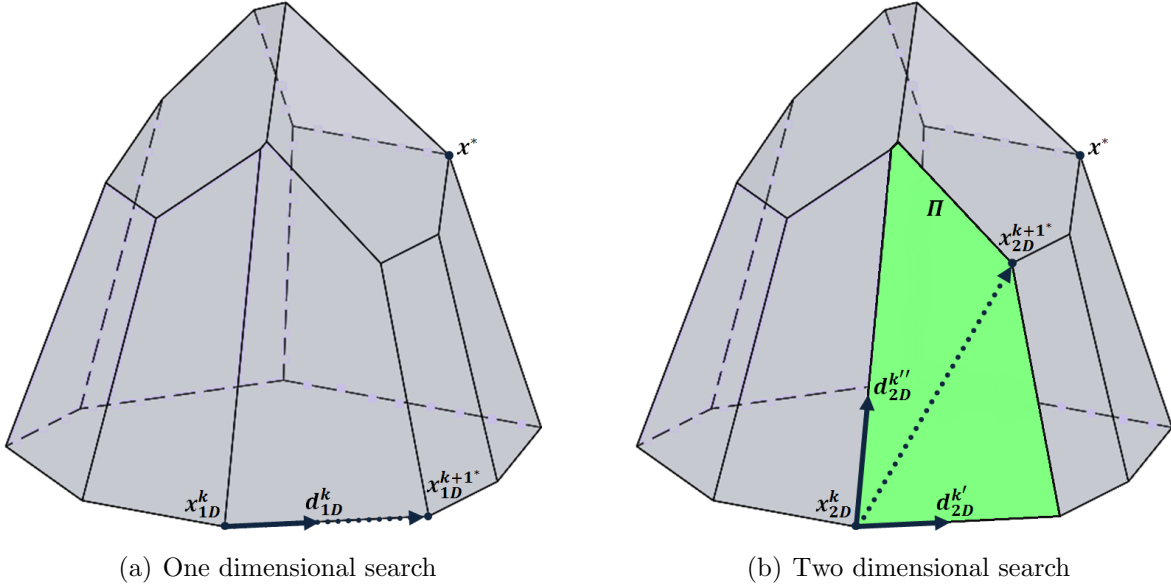


Figure 3.1: Graphical representation of one and two dimensional searches in a simplex framework

Observe that one and two dimensional searches behave similarly in an interior point framework. Figure 3.2(a) presents the concept for one dimensional searches. In this case, x_{1D}^k must be a feasible interior solution as well as x_{1D}^{k+1*} . Since x_{1D}^{k+1*} is at the boundary of the LP's feasible region, x_{1D}^{k+1*} is backed off by a constant $\alpha \in (0, 1)$. Thus, the new and improved feasible interior solution becomes $x_{1D}^{k+1} = x_{1D}^k + \alpha \lambda_{1D}^{k*} d_{1D}^k$. Furthermore, Figure 3.2(b) describes two dimensional searches where x_{2D}^k is feasible and in the interior of the LP's feasible region, and x_{2D}^{k+1*} is backed off so that $x_{2D}^{k+1} = x_{2D}^k + \alpha(\lambda_{2D}^{k'*} d_{2D}^{k'} + \lambda_{2D}^{k''*} d_{2D}^{k''})$ for some $\alpha \in (0, 1)$ is also a feasible interior solution.

The real benefit obtained with these two dimensional search methods can be attributed to the addition of a second search direction. Observe in Figures 3.1(a) and 3.2(a) that either in a simplex or interior point framework, finding a new and improved feasible solution

at each iteration is limited by all the solutions contained in a single ray. When a second search direction is added in Figures 3.1(b) and 3.2(b), the number of potential solutions is drastically increased, which allows these methods to search over a broader space.

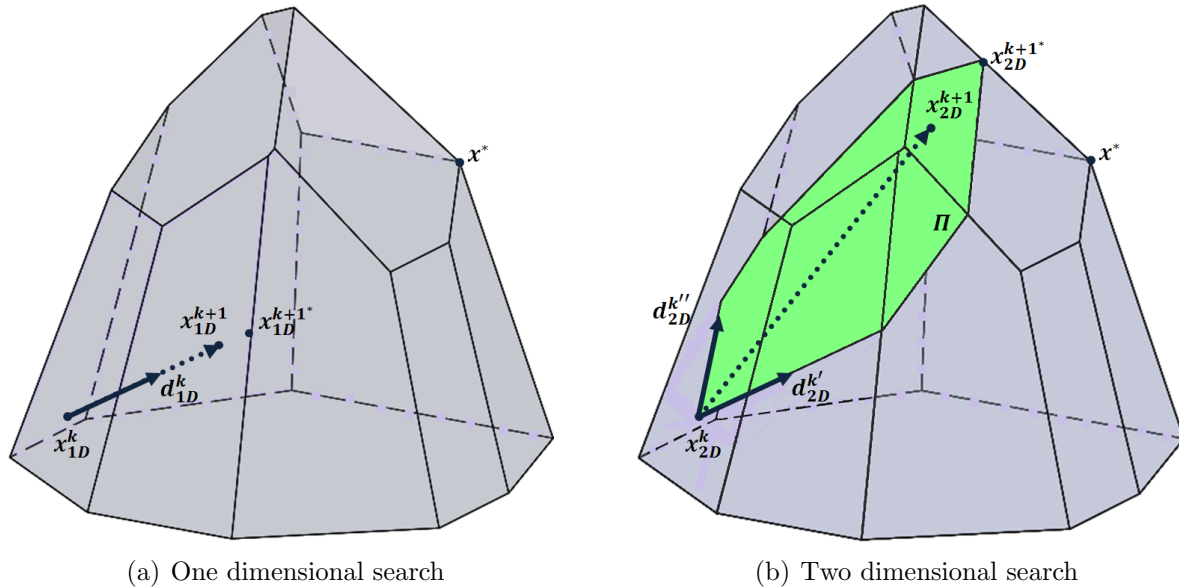


Figure 3.2: Graphical representation of one and two dimensional searches in an interior point framework

Another benefit is the quality of solutions obtained with two dimensional search algorithms. This is because there are an infinity number of one dimensional search directions that can be represented as a nonnegative linear combination of other two search directions. Therefore, if these two search directions are appropriately selected, then these techniques have the potential to improve the object function value at each iteration by more than the corresponding one dimensional search version.

Because two dimensional search techniques examine a broader space to search for a new and improved feasible solution at each step, these methods have the potential to improve the objective function value by more than one dimensional search techniques. Thus, one would expect these algorithms to perform fewer iterations. On the other hand, one could claim that solving a two dimensional subspace linear program may require more computational effort than solving a one dimensional subspace linear program. Therefore, two dimensional search algorithms may not necessarily perform faster than the corresponding one dimensional search

version. Consequently, effective methods should be implemented to solve their subproblems in order to counteract this additional effort. This requires using other algorithms to solve these subproblems rather than the traditional techniques such as the classical simplex method or interior point methods.

Observe that a two dimensional subspace linear program has only two decision variables, and is one of the simplest LPs to solve. These two variable linear programs can be easily solved by the graphical method, which is commonly presented in courses teaching linear programming. Many authors have proposed effective algorithms to solve such simple problems. For instance, [Shamos and Hoey \(1976\)](#) present a $O(r \log r)$ algorithm that can solve two variable linear programs while [Megiddo \(1983\)](#) and [Dyer \(1984\)](#) present linear time algorithms to solve two or three variable linear programs.

Even though these methods can rapidly find an optimal solution to a two variable linear program, such algorithms do not necessarily determine an optimal basis. That is, an optimal basic feasible solution from where there does not exist any feasible improving search directions. For instance, if the two variable linear program has three or more constraints intersecting at an optimal solution, then these methods do not necessarily obtain an optimal basis. If on the other hand, exactly two constraints intersect at an optimal solution, then these algorithms can be easily modified to return an optimal basis.

Since an interior point framework only requires an optimal solution to its subproblems, implementing the method of [Shamos and Hoey](#), [Megiddo](#), or [Dyer](#) to perform two dimensional searches is sufficient. On the other hand, both an optimal basis and an optimal solution is necessary in a simplex framework. This is because selecting a non-optimal basis to its subproblems may result to an unchanged basis of the master problem, and the algorithm may never terminate.

The following section presents the slope algorithm, a newly developed technique to identify both an optimal basis and an optimal solution of two variable linear programs. The subsequent section describes the ratio algorithm, which finds an optimal basis and an optimal solution of LPs with only two constraints. Both techniques can be implemented within

a simplex and an interior point framework, and are the foundation for the two dimensional search algorithms developed for this dissertation's research.

3.2 The Slope Algorithm

The slope algorithm (SA) is designed to solve two variable linear programs within a simplex framework. Thus, these subproblems satisfy three conditions: the cost coefficient of both variables are positive, both of the variables have nonnegativity constraints, and the right-hand side value of every constraint is nonnegative. Formally, let $c_1, c_2 \in \mathbb{R}_+ \setminus \{0\}$, $A \in \mathbb{R}^{r \times 2}$, and $b \in \mathbb{R}_+^r$. Define a two variable linear program (2VLP) as:

$$\begin{aligned} & \text{maximize } z = c_1x_1 + c_2x_2 \\ & \text{subject to } \quad a_{j,1}x_1 + a_{j,2}x_2 \leq b_j \quad \forall j \in R \\ & \quad \quad \quad x_1, x_2 \geq 0. \end{aligned}$$

Denote $S^2 = \{x \in \mathbb{R}_+^2 : a_{j,1}x_1 + a_{j,2}x_2 \leq b_j \forall j \in R\}$ as the feasible region of a 2VLP. To implement SA, the nonnegativity constraints are converted into less than or equal to constraints. Define a slope algorithm two variable linear program (SA2VLP) as:

$$\begin{aligned} & \text{maximize } z = c_1x_1 + c_2x_2 \\ & \text{subject to } \quad a_{j,1}x_1 + a_{j,2}x_2 \leq b_j \quad \forall j \in R', \end{aligned}$$

where $R' = \{1, \dots, r + 2\}$, $c_1 > 0$, $c_2 > 0$, $b_j \geq 0$ for all $j \in R'$, $a_{r+1,2} = a_{r+2,1} = 0$, $a_{r+1,1} = a_{r+2,2} = -1$, and $b_{r+1} = b_{r+2} = 0$.

The slope algorithm evaluates the angle or ‘‘slope’’ of each constraint, and contrasts these values with the slope formed by the cost coefficients c_1 and c_2 . Since the slope of each constraint may not be well defined depending on whether the constraint coefficients are positive, negative, or zero, denote:

$$M > \max \left\{ \max_{j \in R'} \left\{ \left| \frac{a_{j,1}}{a_{j,2}} \right| : a_{j,2} \neq 0 \right\}, \max_{j \in R'} \left\{ \left| \frac{a_{j,2}}{a_{j,1}} \right| : a_{j,1} \neq 0 \right\}, \frac{c_2}{c_1} \right\}.$$

Each constraint $j \in R'$ of an SA2VLP is partitioned into one of nine sets $R'_{\psi\omega}$ where ψ and ω denotes whether $a_{j,1}$ and $a_{j,2}$ are < 0 , $= 0$, or > 0 . Each set assigns a value for a slope coefficient, α_j , as follows:

$$\alpha_j = \begin{cases} -2M & \text{If } j \in R'_{=<} \text{ where } R'_{=<} = \{j \in R' : a_{j,1} = 0, a_{j,2} < 0\} \\ -M + \frac{a_{j,2}}{a_{j,1}} & \text{If } j \in R'_{><} \text{ where } R'_{><} = \{j \in R' : a_{j,1} > 0, a_{j,2} < 0\} \\ -M & \text{If } j \in R'_{>=} \text{ where } R'_{>=} = \{j \in R' : a_{j,1} > 0, a_{j,2} = 0\} \\ \frac{a_{j,2}}{a_{j,1}} & \text{If } j \in R'_{>>} \text{ where } R'_{>>} = \{j \in R' : a_{j,1} > 0, a_{j,2} > 0\} \\ M & \text{If } j \in R'_{=>} \text{ where } R'_{=>} = \{j \in R' : a_{j,1} = 0, a_{j,2} > 0\} \\ M - \frac{a_{j,1}}{a_{j,2}} & \text{If } j \in R'_{<>} \text{ where } R'_{<>} = \{j \in R' : a_{j,1} < 0, a_{j,2} > 0\} \\ 2M & \text{If } j \in R'_{<=} \text{ where } R'_{<=} = \{j \in R' : a_{j,1} < 0, a_{j,2} = 0\} \\ 3M & \text{If } j \in R'_{==} \text{ where } R'_{==} = \{j \in R' : a_{j,1} = 0, a_{j,2} = 0\} \\ 3M & \text{If } j \in R'_{<<} \text{ where } R'_{<<} = \{j \in R' : a_{j,1} < 0, a_{j,2} < 0\}. \end{cases}$$

Due to the large value of M , viewing the constraints in a non-descending order according to the values of the α_j 's creates a counterclockwise orientation of the constraints' slopes starting from the x_1 axis. Figure 3.3 depicts the constraints with their respective α_j values. Observe that only eight out of the nine constraints are viewable because $R'_{==}$ defines the entire two dimensional space.

The first step in creating SA is to determine whether or not an SA2VLP is unbounded. The following lemma provides a relationship between the coefficients from two constraints, which helps derive conditions of an unbounded SA2VLP.

Lemma 3.2.1. *If SA2VLP has j and $k \in R'$ such that $\alpha_j < \alpha_k$ and $\alpha_k \leq -M$, then $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.*

Proof. Assume an SA2VLP has j and $k \in R'$ such that $\alpha_j < \alpha_k$ and $\alpha_k \leq -M$. If $\alpha_j = -2M$, then $a_{j,1} = 0$, $a_{j,2} < 0$, and $-2M < \alpha_k \leq -M$. Thus, $a_{k,1} > 0$. Consequently, $a_{j,2}a_{k,1} < 0$, $a_{k,2}a_{j,1} = 0$, and $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.

If $-2M < \alpha_j < -M$, then $a_{j,1} > 0$, $a_{j,2} < 0$, and $\alpha_j < \alpha_k \leq -M$. If $\alpha_k < -M$, then $-M + \frac{a_{j,2}}{a_{j,1}} < -M + \frac{a_{k,2}}{a_{k,1}}$, which results in $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. If $\alpha_k = -M$, then $a_{k,1} > 0$ and $a_{k,2} = 0$. Thus, $a_{j,2}a_{k,1} < 0$ and $a_{k,2}a_{j,1} = 0$. Consequently, $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. \square

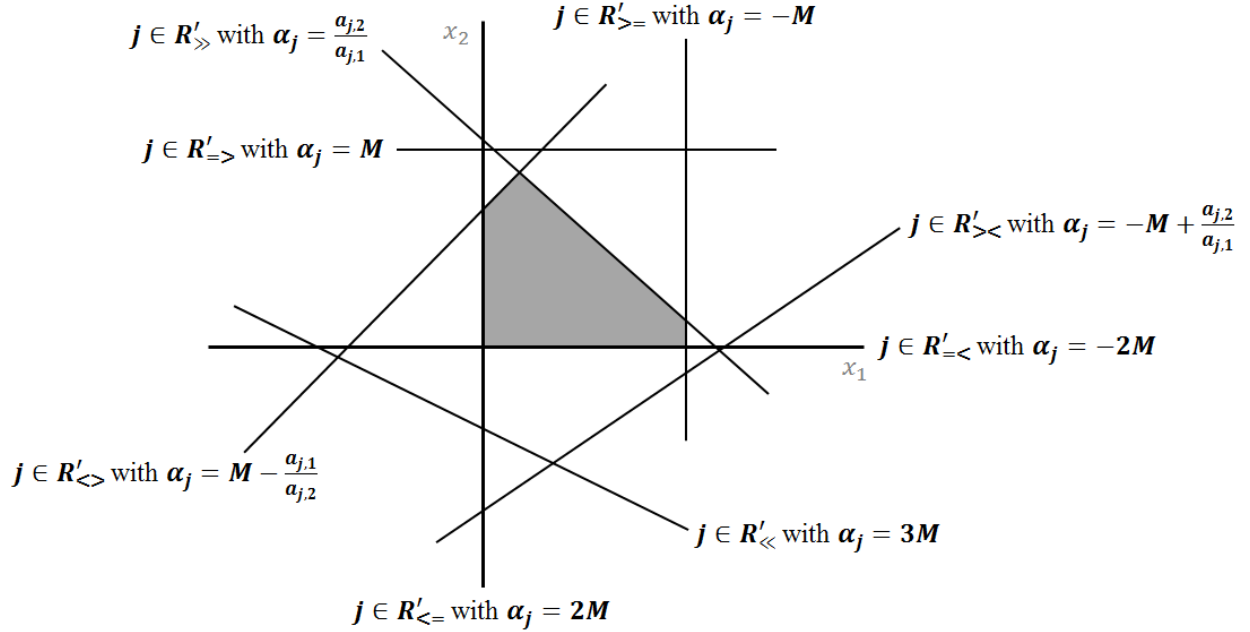


Figure 3.3: Sample α_j for eight classes of constraints of an SA2VLP

Since the point $(0, 0)$ is feasible, $c_1 > 0$, $c_2 > 0$, $x_1 \geq 0$, and $x_2 \geq 0$ for every SA2VLP, thus any feasible ray $d = (d_1, d_2) \in \mathbb{R}^2$ implies that SA2VLP is unbounded. Therefore, if there exists a $d = (d_1, d_2) \in \mathbb{R}_+^2$ such that $a_{j,1}d_1 + a_{j,2}d_2 \leq 0$ for all $j \in R'$, then SA2VLP is unbounded. The following theorem provides necessary and sufficient conditions for an unbounded SA2VLP.

Theorem 3.2.1. *An SA2VLP is unbounded, if and only if, the following three conditions hold:*

- i) $R'_{>>} = \emptyset$;
- ii) If $R'_{>=} \neq \emptyset$, then $R'_{>} = \emptyset$ and $R'_{<>} = \emptyset$;
- iii) If $R'_{><} \neq \emptyset$, then $R'_{>} = \emptyset$ and $\frac{a_{j,2}}{a_{j,1}} \leq \frac{a_{k,2}}{a_{k,1}}$ for every $j \in R'_{><}$ and every $k \in R'_{<>}$.

Proof. Assume an SA2VLP is unbounded. Thus, there exists a ray $d = (d_1, d_2)$ such that $a_{j,1}d_1 + a_{j,2}d_2 \leq 0$ for all $j \in R'$ and $c_1d_1 + c_2d_2 > 0$. Since $x_1 \geq 0$, $x_2 \geq 0$, $c_1 > 0$, and $c_2 > 0$, the unbounded ray must satisfy $d_1 \geq 0$, $d_2 \geq 0$, and $d_1 + d_2 > 0$.

Assume there exists a $j \in R'_{>>}$. Then $a_{j,1} > 0$ and $a_{j,2} > 0$. Evaluating d on the j^{th} constraint results in $a_{j,1}d_1 + a_{j,2}d_2 > 0$, contradicting d being a feasible ray. Thus, $R'_{>>} = \emptyset$ and $i)$ is satisfied.

Assume there exists a $j \in R'_{>=}$. Therefore, $a_{j,1} > 0$ and $a_{j,2} = 0$. Since $d_1 \geq 0$, $d_1 = 0$ or d is not a feasible ray for the j^{th} constraint. Thus, $d = (0, d_2)$ where $d_2 > 0$. If any $k \in R'$ has $a_{k,2} > 0$, then d is not a feasible direction for the k^{th} constraint. Thus, $R'_{>=} = \emptyset$ and $R'_{<>} = \emptyset$, which satisfies $ii)$.

Assume there exists a $j \in R'_{><}$. Consequently, $a_{j,1} > 0$ and $a_{j,2} < 0$. For contradiction, assume $k \in R'_{>=}$, which implies $a_{k,1} = 0$ and $a_{k,2} > 0$. In order for d to be an improving direction that is feasible on the k^{th} constraint, $d_2 = 0$ and $d_1 > 0$. Thus, $a_{j,1}d_1 + a_{j,2}d_2 > 0$, contradicting d being a feasible direction for the j^{th} constraint. Therefore, $R'_{>=} = \emptyset$ and the first condition of $iii)$ is satisfied.

Assume there exists $j \in R'_{><}$ and $k \in R'_{<>}$. This implies that $a_{j,1} > 0$, $a_{j,2} < 0$, $a_{k,1} < 0$, and $a_{k,2} > 0$. Since d is an improving direction and feasible on the j^{th} and k^{th} constraints, $d_1 > 0$, $d_2 > 0$, $a_{j,1}d_1 + a_{j,2}d_2 \leq 0$, and $a_{k,1}d_1 + a_{k,2}d_2 \leq 0$. Therefore, $d_1 \leq -\frac{a_{j,2}d_2}{a_{j,1}}$. Substituting into the second inequality results in $a_{k,1}\left(-\frac{a_{j,2}d_2}{a_{j,1}}\right) + a_{k,2}d_2 \leq 0$, which implies $d_2\left(-\frac{a_{k,1}a_{j,2}}{a_{j,1}} + a_{k,2}\right) \leq 0$. Since $d_2 > 0$, $-\frac{a_{k,1}a_{j,2}}{a_{j,1}} + a_{k,2} \leq 0$, implying that $\frac{a_{j,2}}{a_{j,1}} \leq \frac{a_{k,2}}{a_{k,1}}$. Therefore, the second condition of $iii)$ is satisfied.

Conversely, assume an SA2VLP satisfies conditions $i)$, $ii)$, and $iii)$. Let $j^* \in R'_{<=} \cup R'_{><} \cup R'_{>=}$ such that $\alpha_{j^*} \geq \alpha_j$ for all $j \in R'_{<=} \cup R'_{><} \cup R'_{>=}$. The claim is that the $j^{*\text{th}}$ constraint defines a ray of unboundedness, which is $d = (-a_{j^*,2}, a_{j^*,1})$. Trivially, $c_1(-a_{j^*,2}) + c_2(a_{j^*,1}) > 0$ for any $j^* \in R'_{<=} \cup R'_{><} \cup R'_{>=}$. Since the point $(0, 0)$ is feasible, it suffices to show that d is a feasible direction for each $k \in R'$.

Let $k \in R'$ such that $\alpha_k < \alpha_{j^*}$. The conditions of Lemma 3.2.1 are satisfied and $a_{k,2}a_{j^*,1} < a_{k,1}a_{j^*,2}$. This implies that $a_{k,2}a_{j^*,1} - a_{k,1}a_{j^*,2} < 0$. If $k \in R'$ such that $\alpha_k = \alpha_{j^*}$, then the $j^{*\text{th}}$ and $k^{*\text{th}}$ constraints are parallel and $a_{k,2}a_{j^*,1} - a_{k,1}a_{j^*,2} = 0$. Consequently, d is a feasible

direction for every such constraint and the remainder of the cases need only to consider $\alpha_k > \alpha_{j^*}$.

If $j^* \in R'_{=<}$, then $R'_{><} = \emptyset$ and $R'_{>=} = \emptyset$ because $\alpha_{j^*} \geq \alpha_j$ for all $j \in R'_{=<} \cup R'_{><} \cup R'_{>=}$. Since $R'_{>>} = \emptyset$, $a_{k,1} \leq 0$ for all $k \in R'$. Thus, $a_{k,1}(-a_{j^*,2}) + a_{k,2}(0) \leq 0$ for all $k \in R'$ and d is a feasible improving ray.

If $j^* \in R'_{><}$, then $a_{j^*,1} > 0$, $a_{j^*,2} < 0$, $R'_{>>} = \emptyset$, $R'_{=>} = \emptyset$, and $R'_{>=} = \emptyset$ due to *i*), the first condition of *iii*), and α_{j^*} being the maximum α_j for all $j \in R'_{=<} \cup R'_{><} \cup R'_{>=}$. For any $k \in R'_{=<} \cup R'_{==} \cup R'_{<<}$, $a_{k,1}(-a_{j^*,2}) \leq 0$ and $a_{k,2}(a_{j^*,1}) \leq 0$, so $a_{k,1}(-a_{j^*,2}) + a_{k,2}(a_{j^*,1}) \leq 0$. If $k \in R'_{<>}$, $a_{k,1} < 0$. The second condition of *iii*), $\left(\frac{a_{j^*,2}}{a_{j^*,1}} \leq \frac{a_{k,2}}{a_{k,1}} \text{ for all } k \in R'_{<>}\right)$, implies $a_{j^*,2}a_{k,1} \geq a_{k,2}a_{j^*,1}$. Therefore, $a_{k,1}(-a_{j^*,2}) + a_{k,2}(a_{j^*,1}) \leq 0$. Consequently, d is a feasible direction for each $k \in R'$ such that $\alpha_k > \alpha_{j^*}$. Thus, d is a ray of unboundedness.

If $j^* \in R'_{>=}$, then $R'_{>>} = \emptyset$, $R'_{=>} = \emptyset$, and $R'_{<>} = \emptyset$ according to *i*) and *ii*). Therefore, $a_{k,2} \leq 0$ and $a_{k,1}(0) + a_{k,2}(a_{j^*,1}) \leq 0$ for all $k \in R'$ such that $\alpha_k > \alpha_{j^*}$. Consequently, d is a feasible improving ray. Since all cases have an improving ray, SA2VLP is unbounded. \square

The three graphs in Figure 3.4 illustrate the conditions of Theorem 3.2.1. The j^* constraint is labeled in each figure and is represented by a solid line. This constraint identifies a ray of unboundedness as shown in the theorem. The dashed lines demonstrate constraints that cannot exist for SA2VLP to be unbounded.

Since the point $(0, 0)$ is feasible, an optimal solution to SA2VLP exists whenever SA2VLP is bounded. The following lemma and two corollaries provide other useful relationships between the coefficients of two constraints and also the objective function coefficients c_1 and c_2 . From any SA2VLP, define SA2VLP $_{j,k}$ to be an SA2VLP with only four constraints. The constraints are the two nonnegativity constraints and the j^{th} and k^{th} constraints from SA2VLP.

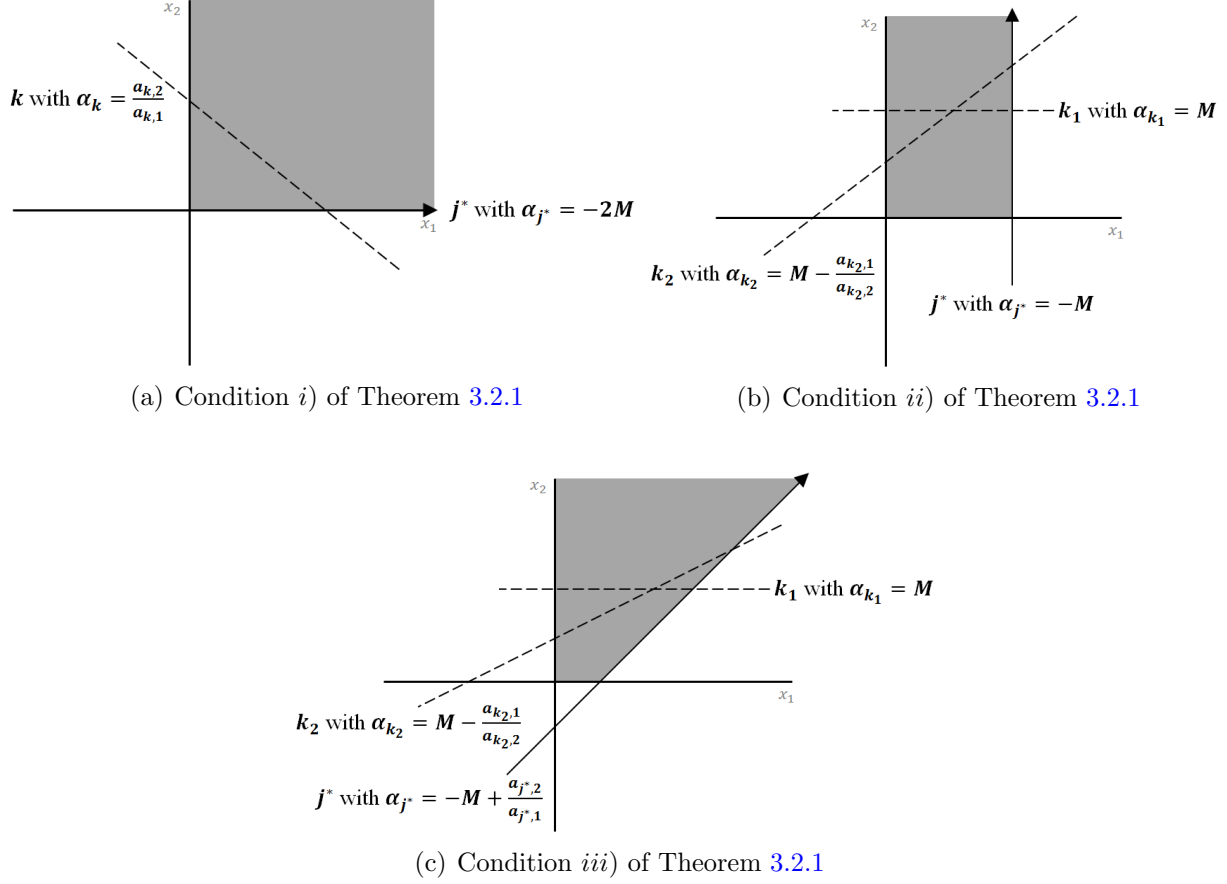


Figure 3.4: Graphical representation of conditions *i*), *ii*), and *iii*) from Theorem 3.2.1

Lemma 3.2.2. *If an SA2VLP has j and $k \in R'$ such that $SA2VLP_{j,k}$ is bounded, $\alpha_j < M$, $-M < \alpha_k \leq 2M$, and $\alpha_j < \alpha_k$, then $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.*

Proof. Assume an SA2VLP has j and $k \in R'$ such that $SA2VLP_{j,k}$ is bounded, $\alpha_j < M$, $-M < \alpha_k \leq 2M$, and $\alpha_j < \alpha_k$. From Theorem 3.2.1, the possible combinations for the j and k constraints are limited. Therefore, the proof shows that $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$ for all possible values of α_j .

If $\alpha_j = -2M$, then $a_{j,1} = 0$ and $a_{j,2} < 0$. Since $SA2VLP_{j,k}$ is bounded and from the conditions of Theorem 3.2.1, then $\alpha_k < M$. Thus, $a_{k,1} > 0$ and $a_{k,2} > 0$. Consequently, $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.

If $-2M < \alpha_j < -M$, then $a_{j,1} > 0$, $a_{j,2} < 0$, and $\alpha_k < 2M$ or $SA2VLP_{j,k}$ is unbounded from Theorem 3.2.1. If $\alpha_k \leq M$, then $a_{k,1} > 0$ and $a_{k,2} \geq 0$. Thus, $a_{j,2}a_{k,1} < 0$ and

$a_{j,1}a_{k,2} \geq 0$, which implies $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. If $\alpha_k > M$ and because SA2VLP $_{j,k}$ is bounded, then $\frac{a_{j,2}}{a_{j,1}} > \frac{a_{k,2}}{a_{k,1}}$ by the results from Theorem 3.2.1. Hence, $a_{j,2}a_{k,1} < a_{k,2}a_{j,1}$.

If $\alpha_j = -M$, then $a_{j,1} > 0$, $a_{j,2} = 0$, and $\alpha_k < 2M$ or SA2VLP $_{j,k}$ is unbounded. Thus, $a_{k,2} > 0$, $a_{j,2}a_{k,1} = 0$, and $a_{j,1}a_{k,2} > 0$. Consequently, $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$.

If $-M < \alpha_j < M$, then $a_{j,1} > 0$ and $a_{j,2} > 0$. If $\alpha_k < M$, then $\frac{a_{j,2}}{a_{j,1}} < \frac{a_{k,2}}{a_{k,1}}$ and $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$, because $\alpha_j < \alpha_k$. If $M \leq \alpha_k < 2M$, then $a_{k,1} \leq 0$ and $a_{k,2} > 0$. Consequently, $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. If $\alpha_k = 2M$, then $a_{k,1} < 0$ and $a_{k,2} = 0$, and so $a_{j,2}a_{k,1} < a_{j,1}a_{k,2} = 0$. \square

Observe that the right-hand side b_j and b_k are not contained in the proofs or statements of Lemmas 3.2.1 and 3.2.2. These lemmas are based solely on the slopes, represented by α_j and α_k . However, these relationships remain between a constraint and the cost coefficients c_1 and c_2 . Since $c_1 > 0$ and $c_2 > 0$, the α value of the objective function is equivalent to $\frac{c_2}{c_1}$. Since $-M < \frac{c_2}{c_1} < M$, the following two corollaries are direct applications of Lemma 3.2.2.

Corollary 3.2.1. *If SA2VLP has a $j \in R'$ such that $\alpha_j < \frac{c_2}{c_1}$, then $a_{j,2}c_1 < a_{j,1}c_2$.* \square

Corollary 3.2.2. *If SA2VLP has a $k \in R'$ such that $\frac{c_2}{c_1} \leq \alpha_k \leq 2M$, then $c_2a_{k,1} \leq c_1a_{k,2}$.*

Proof. Assume an SA2VLP has a $k \in R'$ such that $\frac{c_2}{c_1} < \alpha_k$. From Lemma 3.2.2, $c_2a_{k,1} < c_1a_{k,2}$. If $\frac{c_2}{c_1} = \alpha_k$, then $a_{k,1} > 0$ and $a_{k,2} > 0$. Thus, $\frac{c_2}{c_1} = \frac{a_{k,2}}{a_{k,1}}$ and $c_2a_{k,1} = c_1a_{k,2}$, which implies $c_2a_{k,1} \leq c_1a_{k,2}$. \square

There are typically an infinite number of points in \mathbb{R}^2 that satisfy both the j^{th} and k^{th} constraints. This dissertation defines the intersection point of the j^{th} and k^{th} constraints as the unique extreme point of the feasible region defined by only these two constraints. If no such extreme point exists, then the j^{th} and k^{th} constraints are parallel, $a_{j,1}a_{k,2} - a_{j,2}a_{k,1} = 0$, and the constraints are said to be nonintersecting. Consequently, any intersecting constraints satisfy $a_{j,1}a_{k,2} - a_{j,2}a_{k,1} \neq 0$, and the intersection point is given by $\bar{x} = (\bar{x}_1, \bar{x}_2)$, where:

$$\bar{x}_1 = \frac{b_j a_{k,2} - b_k a_{j,2}}{a_{j,1} a_{k,2} - a_{j,2} a_{k,1}} \quad \text{and} \quad \bar{x}_2 = \frac{a_{j,1} b_k - b_j a_{k,1}}{a_{j,1} a_{k,2} - a_{j,2} a_{k,1}}.$$

Theorem 3.2.2 identifies nonsupportive constraints in S^2 by evaluating \bar{x} on a particular constraint. A constraint is said to support S^2 if there exists a point in S^2 that meets the constraint at equality.

Theorem 3.2.2. *If a bounded SA2VLP has $i, j,$ and $k \in R'$ such that $\alpha_i < \alpha_j < \alpha_k \leq 2M$, $\alpha_j < M$, and the intersection point of the j^{th} and k^{th} constraints violates the i^{th} constraint, then the j^{th} constraint does not support S^2 .*

Proof. Assume an SA2VLP is bounded and there exists constraints $i, j,$ and $k \in R'$ such that $\alpha_i < \alpha_j < \alpha_k \leq 2M$, $\alpha_j < M$, and the intersection point of the j^{th} and k^{th} constraints, \bar{x} , violates the i^{th} constraint. The points in \mathbb{R}^2 that meet the j^{th} constraint at equality can be expressed as $\bar{x} + \rho(a_{j,2}, -a_{j,1})$ and $\bar{x} + \lambda(-a_{j,2}, a_{j,1})$ where $\rho \geq 0$ and $\lambda > 0$.

Evaluating $\bar{x} + \rho(a_{j,2}, -a_{j,1})$ for all $\rho \geq 0$ on the i^{th} constraint results in $a_{i,1}\bar{x}_1 + a_{i,2}\bar{x}_2 + \rho(a_{i,1}a_{j,2} - a_{i,2}a_{j,1})$. Since \bar{x} violates the i^{th} constraint, $a_{i,1}\bar{x}_1 + a_{i,2}\bar{x}_2 > b_i$. The i^{th} and j^{th} constraints satisfy either the conditions of Lemma 3.2.1 or Lemma 3.2.2 and $a_{i,2}a_{j,1} < a_{i,1}a_{j,2}$. Therefore, $a_{i,1}\bar{x}_1 + a_{i,2}\bar{x}_2 + \rho(a_{i,1}a_{j,2} - a_{i,2}a_{j,1}) > b_i$ for all $\rho \geq 0$, and none of these points are contained in S^2 .

Evaluating $\bar{x} + \lambda(-a_{j,2}, a_{j,1})$ for all $\lambda > 0$ on the k^{th} constraint results in $a_{k,1}\bar{x}_1 + a_{k,2}\bar{x}_2 + \lambda(-a_{k,1}a_{j,2} + a_{k,2}a_{j,1})$. Since \bar{x} satisfies the k^{th} constraint, $a_{k,1}\bar{x}_1 + a_{k,2}\bar{x}_2 = b_k$. The j^{th} and k^{th} constraints satisfy either the conditions of Lemma 3.2.1 or Lemma 3.2.2 and $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. Thus, $a_{k,1}\bar{x}_1 + a_{k,2}\bar{x}_2 + \lambda(-a_{k,1}a_{j,2} + a_{k,2}a_{j,1}) > b_k$ for all $\lambda > 0$, and none of these points are contained in S^2 . Consequently, the j^{th} constraint does not support S^2 . \square

Corollary 3.2.3 trivially extends this result to constraints with larger slope values. The proof is obtained by simply swapping x_1 and x_2 and applying Theorem 3.2.2.

Corollary 3.2.3. *If a bounded SA2VLP has $j, k,$ and $l \in R'$ such that $\alpha_j < \alpha_k < \alpha_l \leq 2M$, $-M < \alpha_k$, and the intersection point of the j^{th} and k^{th} constraints violates the l^{th} constraint, then the k^{th} constraint does not support S^2 .* \square

The next two corollaries identify the intersection points of constraints and apply Theorem 3.2.2 and Corollary 3.2.3 to determine nonsupportive constraints. Combining these results

enables the creation of a linear time procedure to identify a feasible extreme point of S^2 , assuming constraints are sorted in a non-descending order according to their α_j values.

Corollary 3.2.4. *If a bounded SA2VLP has h, i, j , and $k \in R'$ such that $\alpha_h < \alpha_i < \alpha_j < \alpha_k \leq 2M$, $\alpha_j < M$, and the intersection point of the j^{th} and k^{th} constraints satisfies the i^{th} constraint but violates the h^{th} constraint, then the i^{th} and j^{th} constraints do not support S^2 .*

Proof. Assume a bounded SA2VLP has constraints h, i, j , and $k \in R'$ such that $\alpha_h < \alpha_i < \alpha_j < \alpha_k \leq 2M$, $\alpha_j < M$, the intersection point of the j^{th} and k^{th} constraints, \bar{x} , satisfies the i^{th} constraint, and violates the h^{th} constraint. Let the intersection point of the i^{th} and k^{th} constraints be $\bar{\bar{x}}$. Since \bar{x} satisfies the i^{th} constraint, thus $\bar{\bar{x}} = \bar{x} + \lambda(a_{k,2}, -a_{k,1})$ for some $\lambda \geq 0$. Evaluating $\bar{\bar{x}}$ on the h^{th} constraint results in $a_{h,1}\bar{\bar{x}}_1 + a_{h,2}\bar{\bar{x}}_2 + \lambda(a_{h,1}a_{k,2} - a_{h,2}a_{k,1})$. Since \bar{x} violates the h^{th} constraint, then $a_{h,1}\bar{x}_1 + a_{h,2}\bar{x}_2 > b_h$. The h^{th} and k^{th} constraints satisfy either the conditions of Lemma 3.2.1 or Lemma 3.2.2 and $a_{h,2}a_{k,1} < a_{h,1}a_{k,2}$. Thus, $a_{h,1}\bar{\bar{x}}_1 + a_{h,2}\bar{\bar{x}}_2 + \lambda(a_{h,1}a_{k,2} - a_{h,2}a_{k,1}) > b_h$. Consequently, $\bar{\bar{x}}$ violates the h^{th} constraint and the conditions of Theorem 3.2.2 are satisfied, which implies that the i^{th} and j^{th} constraints do not support S^2 . \square

Corollary 3.2.5. *If a bounded SA2VLP has j, k, l , and $m \in R'$ such that $\alpha_j < \alpha_k < \alpha_l < \alpha_m \leq 2M$, $-M < \alpha_k$, and the intersection point of the j^{th} and k^{th} constraints satisfies the l^{th} constraint, but violates the m^{th} constraint, then the l^{th} and k^{th} constraints do not support S^2 .* \square

The slope algorithm identifies an optimal solution to SA2VLP by finding constraints j and $k \in R'$ such that $\alpha_j < \frac{c_2}{c_1} \leq \alpha_k$, SA2VLP $_{j,k}$ is bounded, the intersection point of the j^{th} and k^{th} constraints is feasible, and $\alpha_k - \alpha_j$ is minimized. The following theorem formalizes that the first three conditions are sufficient to identify an optimal solution to SA2VLP.

Theorem 3.2.3. *An optimal solution to an SA2VLP occurs at the intersection point of the j^{th} and k^{th} constraints if the following three conditions hold:*

i) $\alpha_j < \frac{c_2}{c_1} \leq \alpha_k \leq 2M$;

ii) $\text{SA2VLP}_{j,k}$ is bounded;

iii) The intersection point of the j^{th} and k^{th} constraints is feasible.

Proof. Assume SA2VLP has j and $k \in R'$ such that $\text{SA2VLP}_{j,k}$ is bounded, $\alpha_j < \frac{c_2}{c_1} \leq \alpha_k \leq 2M$, and the intersection point of the j^{th} and k^{th} constraints, \bar{x} , is feasible. The proof shows that every direction from \bar{x} is either infeasible or nonimproving. Partitioning all possible directions in $\mathbb{R}^2 \setminus \{(0,0)\}$ results in the following four sets:

- $D^1 = \{d \in \mathbb{R}^2 \setminus \{(0,0)\} : d = \beta(a_{j,2}, -a_{j,1}) + (1 - \beta)(a_{k,2}, -a_{k,1}) \ \forall \beta \in [0, 1]\}$;
- $D^2 = \{d \in \mathbb{R}^2 \setminus \{(0,0)\} : d = \beta(-a_{j,2}, a_{j,1}) + (1 - \beta)(a_{k,2}, -a_{k,1}) \ \forall \beta \in (0, 1]\}$;
- $D^3 = \{d \in \mathbb{R}^2 \setminus \{(0,0)\} : d = \beta(-a_{j,2}, a_{j,1}) + (1 - \beta)(-a_{k,2}, a_{k,1}) \ \forall \beta \in (0, 1]\}$;
- $D^4 = \{d \in \mathbb{R}^2 \setminus \{(0,0)\} : d = \beta(a_{j,2}, -a_{j,1}) + (1 - \beta)(-a_{k,2}, a_{k,1}) \ \forall \beta \in [0, 1]\}$.

Since $-M < \frac{c_2}{c_1} \leq \alpha_k$, the j^{th} and k^{th} constraints satisfy Lemma 3.2.2 and $a_{j,2}a_{k,1} < a_{j,1}a_{k,2}$. Evaluating any $d \in D^1$ on the j^{th} constraint results in $\beta(a_{j,1}a_{j,2}) + (1 - \beta)(a_{j,1}a_{k,2}) - \beta(a_{j,2}a_{j,1}) - (1 - \beta)(a_{j,2}a_{k,1}) = (1 - \beta)(a_{j,1}a_{k,2} - a_{j,2}a_{k,1}) > 0$ for all $\beta \in [0, 1]$. Thus, $\bar{x} + \lambda d$ violates the j^{th} constraint for every $\lambda > 0$, and D^1 is a set of infeasible directions from \bar{x} .

Evaluating any $d \in D^2$ on the j^{th} constraint results in $-\beta(a_{j,1}a_{j,2}) + (1 - \beta)(a_{j,1}a_{k,2}) + \beta(a_{j,2}a_{j,1}) - (1 - \beta)(a_{j,2}a_{k,1}) = (1 - \beta)(a_{j,1}a_{k,2} - a_{j,2}a_{k,1}) > 0$ for all $\beta \in (0, 1]$. Thus, $\bar{x} + \lambda d$ violates the j^{th} constraint for every $\lambda > 0$, and D^2 is a set of infeasible directions from \bar{x} .

Evaluating any $d \in D^3$ on the k^{th} constraint results in $-\beta(a_{k,1}a_{j,2}) - (1 - \beta)(a_{k,1}a_{k,2}) + \beta(a_{k,2}a_{j,1}) + (1 - \beta)(a_{k,2}a_{k,1}) = \beta(-a_{k,1}a_{j,2} + a_{k,2}a_{j,1}) > 0$ for all $\beta \in (0, 1]$. Thus, $\bar{x} + \lambda d$ violates the k^{th} constraint for every $\lambda > 0$, and D^3 is a set of infeasible directions from \bar{x} .

Evaluating any $d \in D^4$ on the objective function results in $\beta(c_1a_{j,2} - c_2a_{j,1}) + (1 - \beta)(-c_1a_{k,2} + c_2a_{k,1})$. Since $\alpha_j < \frac{c_2}{c_1}$, the conditions of Corollary 3.2.1 are satisfied and

$a_{j,2}c_1 < a_{j,1}c_2$. Thus, $\beta(c_1a_{j,2} - c_2a_{j,1}) \leq 0$ for all $\beta \in [0, 1]$. Since $\frac{c_2}{c_1} \leq \alpha_k$, $c_2a_{k,1} \leq c_1a_{k,2}$ by Corollary 3.2.2 and $(1-\beta)(-c_1a_{k,2} + c_2a_{k,1}) \leq 0$ for all $\beta \in [0, 1]$. Consequently, every $d \in D^4$ is a nonimproving direction. Since every direction from \bar{x} is infeasible or nonimproving and SA2VLP is a linear convex problem, \bar{x} is an optimal solution. \square

The above results enable the creation of SA, which optimally solves 2VLPs. The slope algorithm is shown in Algorithm 3.1. The input to SA is a 2VLP and a sufficiently large positive number M . The slope algorithm returns that either 2VLP is unbounded or an optimal solution (z^*, x^*) along with $j^* \in R'$, $k^* \in R'$, α_{j^*} , and α_{k^*} . The j^* and k^* represent the intersecting constraints that provide an optimal basis and an optimal solution to 2VLP. Even though j^* , k^* , α_{j^*} , and α_{k^*} are not part of 2VLP's optimal solution, this information is necessary to identify an optimal basis. The proof that j^* and k^* identifies an optimal basis of a 2VLP is presented in Chapter 4.2.

The slope algorithm correctly solves any SA2VLP. The check for unboundedness follows the conditions of Theorem 3.2.1 when the constraints are viewed from their α values. If SA2VLP is bounded, then the algorithm returns an x^* at the intersection point of two constraints, j^* and k^* . Clearly, such a j^* and k^* exist due to the nonnegativity constraints. One constraint has an α value less than $\frac{c_2}{c_1}$ and the other constraint has an α value greater than or equal to $\frac{c_2}{c_1}$. The point is validated against all constraints according to Theorem 3.2.2 and Corollaries 3.2.3, 3.2.4, and 3.2.5. From Theorem 3.2.3, x^* is an optimal solution to SA2VLP.

To determine the theoretical running time of SA, observe that the algorithm first calculates every element of the array α in $O(r)$ and sorts this array (lines 3-4). There are numerous sorting algorithms and let $S(r)$ be the effort required by the selected algorithm to sort r elements. The main step of SA (lines 5-28) first determines two intersecting constraints in $O(r)$. The check for unboundedness is performed in $O(1)$. If SA2VLP is bounded, then SA calculates \bar{x} in $O(1)$.

Algorithm 3.1 : The Slope Algorithm (SA)

```
1: begin
2:   From a 2VLP, create the corresponding SA2VLP;
3:   Calculate  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_{r+2})$ ;
4:   Let  $P = (\rho_1, \dots, \rho_{r+2})$  be a list of sorted constraint indices such that  $\alpha_{\rho_j} \leq \alpha_{\rho_{j+1}}$ 
    $\forall j \in \{1, \dots, r+1\}$ ;
5:   Determine  $j' \in R'$  such that  $\alpha_{\rho_{j'}} < \frac{c_2}{c_1} \leq \alpha_{\rho_{j'+1}}$ ;
6:    $k' \leftarrow j' + 1$ ;
7:   if  $\left( \alpha_{\rho_{j'}} = -2M \text{ and } \alpha_{\rho_{k'}} \geq M \right)$  or  $\left( -2M < \alpha_{\rho_{j'}} < -M \text{ and } \alpha_{\rho_{k'}} = 2M \right)$  or
    $\left( \alpha_{\rho_{j'}} = -M \text{ and } \alpha_{\rho_{k'}} = 2M \right)$  or
    $\left( -2M < \alpha_{\rho_{j'}} < -M \text{ and } M < \alpha_{\rho_{k'}} < 2M \text{ and } \frac{a_{\rho_{j'},2}}{a_{\rho_{j'},1}} \leq \frac{a_{\rho_{k'},2}}{a_{\rho_{k'},1}} \right)$  then
8:     return 2VLP is unbounded;
9:   else
10:     $j \leftarrow j'$ ;
11:     $k \leftarrow k'$ ;
12:    Calculate  $\bar{x} = (\bar{x}_1, \bar{x}_2)$  from constraints  $\rho_{j'}$  and  $\rho_{k'}$ ;
13:    while  $j > 1$  or  $k < r + 2$  do
14:      if  $j > 1$  then  $j \leftarrow j - 1$ ;
15:      if  $a_{\rho_j,1}\bar{x}_1 + a_{\rho_j,2}\bar{x}_2 > b_{\rho_j}$  then
16:         $j' \leftarrow j$ ;
17:         $k' \leftarrow k$ ;
18:        Calculate  $\bar{x} = (\bar{x}_1, \bar{x}_2)$  from constraints  $\rho_{j'}$  and  $\rho_{k'}$ ;
19:      if  $k < r + 2$  then  $k \leftarrow k + 1$ ;
20:      if  $a_{\rho_k,1}\bar{x}_1 + a_{\rho_k,2}\bar{x}_2 > b_{\rho_k}$  then
21:         $k' \leftarrow k$ ;
22:         $j' \leftarrow j$ ;
23:        Calculate  $\bar{x} = (\bar{x}_1, \bar{x}_2)$  from constraints  $\rho_{j'}$  and  $\rho_{k'}$ ;
24:       $z^* \leftarrow c_1\bar{x}_1 + c_2\bar{x}_2$ ;
25:       $x^* \leftarrow \bar{x}$ ;
26:       $j^* \leftarrow \rho_{j'}$ ;
27:       $k^* \leftarrow \rho_{k'}$ ;
28:      return  $z^*, x^*, j^*, k^*, \alpha_{j^*}$ , and  $\alpha_{k^*}$ ;
29: end
```

Each iteration of the while loop either validates that the current \bar{x} is feasible on up to two constraints in $O(1)$, or that \bar{x} violates a constraint. If a violation occurs, a new \bar{x} is calculated. From Corollaries 3.2.4 and 3.2.5, if \bar{x} violates a constraint, then every constraint between j' and j or k' and k is nonsupportive in SA2VLP. Consequently, the while loop is repeated at most $O(r)$ times. Thus, the entire main step requires $O(r)$ effort. Furthermore, SA requires

$O(1)$ to report a solution to SA2VLP or that SA2VLP is unbounded. Consequently, SA requires $O(S(r))$ effort and the most time consuming step is sorting the array α . From merge sort (Sedgewick and Wayne, 2011), $S(r) = r \log r$ and SA runs in $O(r \log r)$ time.

Example 3.2.1 demonstrates the implementation of SA to solve a 2VLP. In addition, Figure 3.5 presents a graphical representation of the corresponding SA2VLP in \mathbb{R}^2 .

Example 3.2.1. *Consider the following 2VLP.*

$$\text{maximize } z = 2x_1 + x_2$$

$$\text{subject to } -3x_1 - 5x_2 \leq 0 \quad (1)$$

$$4x_1 + 3x_2 \leq 100 \quad (2)$$

$$2x_1 - x_2 \leq 20 \quad (3)$$

$$x_1 + x_2 \leq 9 \quad (4)$$

$$-2x_1 + x_2 \leq 6 \quad (5)$$

$$x_2 \leq 6 \quad (6)$$

$$3x_1 + x_2 \leq 37 \quad (7)$$

$$x_1 \leq 9 \quad (8)$$

$$x_1 - x_2 \leq 3 \quad (9)$$

$$x_1, x_2 \geq 0$$

The first step of SA converts 2VLP into an SA2VLP by changing the nonnegativity conditions into constraints $-x_1 \leq 0$ (10), $-x_2 \leq 0$ (11), and assigning $R' = \{1, 2, \dots, 11\}$. The slope algorithm calculates $\alpha = (3M, \frac{3}{4}, -M - \frac{1}{2}, 1, M + 2, M, \frac{1}{3}, -M, -M - 1, 2M, -2M)$ and then sorts the indices of the constraints in a non-descending order according to these values, which results in $P = (11, 9, 3, 8, 7, 2, 4, 6, 5, 10, 1)$. The algorithm identifies $j' = 5$ and $k' = 6$ since $\alpha_{\rho_{j'}} = \frac{1}{3} < \frac{c_2}{c_1} = \frac{1}{2} \leq \alpha_{\rho_{j'+1}} = \frac{3}{4}$. Because $-M < \alpha_{\rho_{j'}}$, none of the conditions for an unbounded SA2VLP are satisfied.

The slope algorithm continues by setting $j = 5$, $k = 6$, and calculating the intersection point of the ρ_5 and ρ_6 constraints, (7) and (2). This intersection point is $\bar{x} = (\frac{11}{5}, \frac{152}{5})$,

represented by \bar{x}^1 in Figure 3.5. The slope algorithm assigns $j = 4$ and the feasibility of \bar{x} is validated on the ρ_4 constraint, (8), because $\frac{11}{5} < 9$. Next, $k = 7$ and the point is tested on the ρ_7 constraint, (4). This point is infeasible because $\frac{11}{5} + \frac{152}{5} > 9$. From Corollary 3.2.3, (2) does not support S^2 and SA assigns $k' = 7$ and j returns to 5.

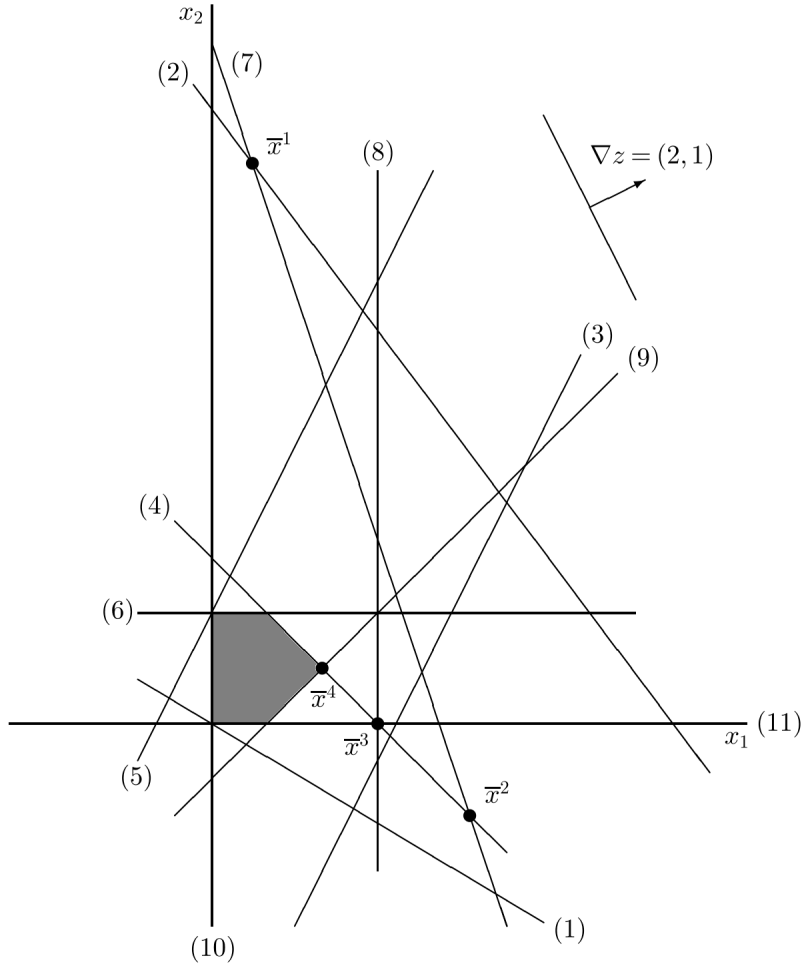


Figure 3.5: Graphical representation of Example 3.2.1

From SA, \bar{x} becomes $(14, -5)$, which is the intersection point of (7) and (4) and is represented by \bar{x}^2 in Figure 3.5. The slope algorithm updates $j = 4$ and \bar{x} is evaluated on (8), which indicates an infeasibility since $14 > 9$. Thus, $j' = 4$ and $k = 7$. The updated \bar{x} occurs at $(9, 0)$, the intersection of (8) and (4), and is represented by \bar{x}^3 in Figure 3.5. The algorithm assigns $k = 8$ and \bar{x} does not violate (6), (3), and (5), but it does violate (9). From Theorem 3.2.2 and Corollary 3.2.4, both (8) and (3) do not support S^2 .

The algorithm assigns $j' = 2$ and $k = 7$. The new $\bar{x} = (6, 3)$ is the intersection point of (9) and (4), and is identified by \bar{x}^4 in Figure 3.5. The slope algorithm follows with $k = 8$ and this point does not violate (6), (11), (5), (10), and (1). Thus, $\bar{x} = (6, 3)$ satisfies all the constraints and by Theorem 3.2.3, this point is an optimal solution to 2VLP. The algorithm reports $z^* = 15$, $x^* = (6, 3)$, $j^* = 9$, $k^* = 4$, $\alpha_9 = -M - 1$, and $\alpha_4 = 1$.

The slope algorithm is a newly created technique to solve simple two variable linear programs with fairly stringent assumptions. Since this dissertation aims to implement SA within a simplex framework to perform two dimensional searches, these assumptions are always true. However, SA can be easily modified to solve any two variable linear program. The author believes that this modification is straightforward, and is left as an exercise. The following section extends the applicability of SA to solve other two dimensional search linear programming problems.

3.3 The Ratio Algorithm

This dissertation's research uses the knowledge from SA to create the ratio algorithm (RA), a novel technique to find an optimal basis and an optimal solution to LPs with only two constraints. Such an optimal basis is required if RA is implemented within a dual simplex framework. Since SA generates an optimal basis and an optimal solution to LPs with only two variables, one could obtain an optimal basis and an optimal solution to two constraint linear programs by solving its corresponding dual problem with SA and applying complementary slackness. The ratio algorithm facilitates this process by implementing the same logic used in SA. In this case, RA compares the ratio defined by each variable with the ratio of the right-hand side values.

Recall from the discussion in Chapter 3.1 that the algorithms of Shamos and Hoey (1976), Megiddo (1983), and Dyer (1984) all solve a two variable linear program. However, these techniques do not necessarily identify an optimal basis. Thus, solving the corresponding dual problem and applying complementary slackness does not guarantee an optimal basis to LPs with only two constraints.

Formally, let $c \in \mathbb{R}_+^n$, $A \in \mathbb{R}^{2 \times n}$, and $b_1, b_2 \in \mathbb{R}_+ \setminus \{0\}$. Define a two constraint linear program (2CLP) as:

$$\begin{aligned} & \text{minimize } z = c_i x_i \\ & \text{subject to } \quad a_{1,i} x_i \geq b_1 \\ & \quad \quad \quad a_{2,i} x_i \geq b_2 \\ & \quad \quad \quad x_i \geq 0 \end{aligned}$$

for all $i \in N = \{1, \dots, n\}$. Denote $\bar{S}^2 = \{x \in \mathbb{R}_+^n : a_{1,i} x_i \geq b_1, a_{2,i} x_i \geq b_2 \forall i \in N\}$ as the feasible region of a 2CLP. The standard form of a two constraint linear program (S2CLP) adds a surplus variable to each of the two constraints. Formally, an S2CLP takes the form:

$$\begin{aligned} & \text{minimize } z = c_i x_i \\ & \text{subject to } \quad a_{1,i} x_i = b_1 \\ & \quad \quad \quad a_{2,i} x_i = b_2 \\ & \quad \quad \quad x_i \geq 0 \end{aligned}$$

for all $i \in N' = \{1, \dots, n + 2\}$, where $c_{n+1} = c_{n+2} = 0$, $a_{1,n+1} = a_{2,n+2} = -1$, and $a_{1,n+2} = a_{2,n+1} = 0$. Furthermore, assume $c_i \geq 0$ for all $i \in N'$, $b_1 > 0$, and $b_2 > 0$. Similarly to SA, these assumptions are considered to facilitate the presentation of the ratio algorithm, but one can easily modify the proposed method to solve any two constraint linear programming problem.

The ratio algorithm calculates the ratio of each variable defined by both constraint values and contrasts these ratios to the ratio of the right-hand side values b_1 and b_2 . Let

$$M > \max \left\{ \max_{i \in N'} \left\{ \left| \frac{a_{1,i}}{a_{2,i}} \right| : a_{2,i} \neq 0 \right\}, \max_{j \in N'} \left\{ \left| \frac{a_{2,j}}{a_{1,j}} \right| : a_{1,j} \neq 0 \right\}, \frac{b_2}{b_1} \right\}$$

be the sufficiently large positive number in this case. Each variable of an S2CLP is partitioned into one of nine sets and the ratio β_i for each variable $i \in N'$ is calculated as follows:

$$\beta_i = \begin{cases} -2M & \text{If } i \in N'_{=<} \text{ where } N'_{=<} = \{i \in N' : a_{1,i} = 0, a_{2,i} < 0\} \\ -M + \frac{a_{2,i}}{a_{1,i}} & \text{If } i \in N'_{><} \text{ where } N'_{><} = \{i \in N' : a_{1,i} > 0, a_{2,i} < 0\} \\ -M & \text{If } i \in N'_{>=} \text{ where } N'_{>=} = \{i \in N' : a_{1,i} > 0, a_{2,i} = 0\} \\ \frac{a_{2,i}}{a_{1,i}} & \text{If } i \in N'_{>>} \text{ where } N'_{>>} = \{i \in N' : a_{1,i} > 0, a_{2,i} > 0\} \\ M & \text{If } i \in N'_{= >} \text{ where } N'_{= >} = \{i \in N' : a_{1,i} = 0, a_{2,i} > 0\} \\ M - \frac{a_{1,i}}{a_{2,i}} & \text{If } i \in N'_{<>} \text{ where } N'_{<>} = \{i \in N' : a_{1,i} < 0, a_{2,i} > 0\} \\ 2M & \text{If } i \in N'_{<=} \text{ where } N'_{<=} = \{i \in N' : a_{1,i} < 0, a_{2,i} = 0\} \\ 3M & \text{If } i \in N'_{==} \text{ where } N'_{==} = \{i \in N' : a_{1,i} = 0, a_{2,i} = 0\} \\ 3M & \text{If } i \in N'_{<<} \text{ where } N'_{<<} = \{i \in N' : a_{1,i} < 0, a_{2,i} < 0\}. \end{cases}$$

Algorithm 3.2 presents RA, and its input is a 2CLP and an appropriate M . The output to RA is either that the corresponding 2CLP is infeasible or an optimal solution (z^*, x^*) , variable indices that define an optimal basis i^* and j^* , and ratios β_{i^*} and β_{j^*} . In this case, denote $\bar{x} = (\bar{x}_i, \bar{x}_j)$ as the solution of the following system with two equations:

$$\begin{aligned} a_{1,i}x_i + a_{1,j}x_j &= b_1 \\ a_{2,i}x_i + a_{2,j}x_j &= b_2, \end{aligned}$$

where

$$\bar{x}_i = \frac{b_1 a_{2,j} - a_{1,j} b_2}{a_{1,i} a_{2,j} - a_{1,j} a_{2,i}} \quad \text{and} \quad \bar{x}_j = \frac{a_{1,i} b_2 - b_1 a_{2,i}}{a_{1,i} a_{2,j} - a_{1,j} a_{2,i}}.$$

The ratio algorithm begins by calculating a ratio β_i for every variable index $i \in N'$. Each variable index is then sorted in a non-descending order according to their β_i values, and two variable indices i and $i + 1$ are determined such that $\beta_i < \frac{b_2}{b_1} \leq \beta_{i+1}$. The algorithm checks if variable indices i and $j = i + 1$ define an infeasible S2CLP. If not, variable indices i and j define an optimal basis and an optimal solution of the corresponding S2CLP, or the ratio algorithm searches for another pair of variable indices with a better objective function value.

Since the ratio algorithm is an extension from the slope algorithm, the majority of theoretical results can be easily derived from Chapter 3.2. Thus, this section only provides the

primary arguments to show that RA correctly solves a 2CLP. The proof that RA identifies an optimal basis for 2CLPs is shown in Chapter 4.3.

Algorithm 3.2 : The Ratio Algorithm (RA)

```

1: begin
2:   From a 2CLP, create the corresponding S2CLP;
3:   Calculate  $\beta = (\beta_1, \beta_2, \dots, \beta_{n+2})$ ;
4:   Let  $K = (\kappa_1, \kappa_2, \dots, \kappa_{n+2})$  be a list of sorted variable indices such that  $\beta_{\kappa_i} \leq \beta_{\kappa_{i+1}}$ 
 $\forall i \in \{1, 2, \dots, n+1\}$ ;
5:   Determine  $i' \in N'$  such that  $\beta_{\kappa_{i'}} < \frac{b_2}{b_1} \leq \beta_{\kappa_{i'+1}}$ ;
6:    $j' \leftarrow i' + 1$ ;
7:   if  $(\beta_{\kappa_{i'}} = -2M$  and  $\beta_{\kappa_{j'}} \geq M)$  or  $(-2M < \beta_{\kappa_{i'}} < -M$  and  $\beta_{\kappa_{j'}} = 2M)$  or
 $(\beta_{\kappa_{i'}} = -M$  and  $\beta_{\kappa_{j'}} = 2M)$  or
 $(-2M < \beta_{\kappa_{i'}} < -M$  and  $M < \beta_{\kappa_{j'}} < 2M$  and  $\frac{a_{2,\kappa_{i'}}}{a_{1,\kappa_{i'}}} \leq \frac{a_{2,\kappa_{j'}}}{a_{1,\kappa_{j'}}})$  then
8:     return S2CLP is infeasible;
9:   else
10:     $i \leftarrow i'$ ;
11:     $j \leftarrow j'$ ;
12:    Calculate  $\bar{x} \leftarrow (\bar{x}_{\kappa_{i'}}, \bar{x}_{\kappa_{j'}})$  and  $z' \leftarrow c_{\kappa_{i'}} \bar{x}_{\kappa_{i'}} + c_{\kappa_{j'}} \bar{x}_{\kappa_{j'}}$ ;
13:    while  $i > 1$  or  $j < n+2$  do
14:      if  $i > 1$  then  $i \leftarrow i - 1$ ;
15:      Calculate  $\bar{x} \leftarrow (\bar{x}_{\kappa_i}, \bar{x}_{\kappa_{j'}})$  and  $z \leftarrow c_{\kappa_i} \bar{x}_{\kappa_i} + c_{\kappa_{j'}} \bar{x}_{\kappa_{j'}}$ ;
16:      if  $z < z'$ ,  $\bar{x}_{\kappa_i} \geq 0$ , and  $\bar{x}_{\kappa_{j'}} \geq 0$  then
17:         $i' \leftarrow i$ ;
18:         $j' \leftarrow j'$ ;
19:        Calculate  $\bar{x} \leftarrow (\bar{x}_{\kappa_{i'}}, \bar{x}_{\kappa_{j'}})$  and  $z' \leftarrow c_{\kappa_{i'}} \bar{x}_{\kappa_{i'}} + c_{\kappa_{j'}} \bar{x}_{\kappa_{j'}}$ ;
20:      if  $j < n+2$  then  $j \leftarrow j + 1$ ;
21:      Calculate  $\bar{x} \leftarrow (\bar{x}_{\kappa_{i'}}, \bar{x}_{\kappa_j})$  and  $z \leftarrow c_{\kappa_{i'}} \bar{x}_{\kappa_{i'}} + c_{\kappa_j} \bar{x}_{\kappa_j}$ ;
22:      if  $z < z'$ ,  $\bar{x}_{\kappa_{i'}} \geq 0$ , and  $\bar{x}_{\kappa_j} \geq 0$  then
23:         $j' \leftarrow j$ ;
24:         $i' \leftarrow i'$ ;
25:        Calculate  $\bar{x} \leftarrow (\bar{x}_{\kappa_{i'}}, \bar{x}_{\kappa_{j'}})$  and  $z' \leftarrow c_{\kappa_{i'}} \bar{x}_{\kappa_{i'}} + c_{\kappa_{j'}} \bar{x}_{\kappa_{j'}}$ ;
26:       $z^* \leftarrow z'$ ;
27:       $x_{\kappa_{i'}}^* \leftarrow \bar{x}_{\kappa_{i'}}$ ;
28:       $x_{\kappa_{j'}}^* \leftarrow \bar{x}_{\kappa_{j'}}$ ;
29:       $x_q^* \leftarrow 0$  for all  $q \in N' \setminus (\kappa_{i'} \cup \kappa_{j'})$ ;
30:       $i^* \leftarrow \kappa_{i'}$ ;
31:       $j^* \leftarrow \kappa_{j'}$ ;
32:      return  $z^*$ ,  $x^*$ ,  $i^*$ ,  $j^*$ ,  $\beta_{i^*}$ , and  $\beta_{j^*}$ ;
33: end

```

The check for an infeasible S2CLP identifies if either $\bar{x}_i \not\geq 0$, $\bar{x}_j \not\geq 0$, and/or no solution exists to the system of equations defined by variable indices i and j . From Algorithm 3.2, four conditions exist to determine if an S2CLP is infeasible: (1st) $\beta_i = -2M$ and $\beta_j \geq M$; (2nd) $-2M < \beta_i < -M$ and $\beta_j = 2M$; (3rd) $\beta_i = -M$ and $\beta_j = 2M$; (4th) $-2M < \beta_i < -M$, $M < \beta_j < 2M$, and $\frac{a_{2,i}}{a_{1,i}} \leq \frac{a_{2,j}}{a_{1,j}}$. Observe that an S2CLP is infeasible if at least one of the four conditions is satisfied. For instance, the second condition states $\beta_i = -M + \frac{a_{2,i}}{a_{1,i}}$ and $\beta_j = 2M$. Since $a_{2,i} < 0$, $a_{2,j} = 0$, and $b_2 > 0$, clearly $\bar{x}_i \not\geq 0$. Given $\bar{x}_i < 0$, $a_{1,i} > 0$, $a_{1,j} < 0$, and $b_1 > 0$, then $\bar{x}_j \not\geq 0$ and variable indices i and j define an infeasible S2CLP. The reader can easily verify the other three conditions and conclude that all three also define an infeasible S2CLP.

If an S2CLP is feasible, the check for optimality defined by the while loop in Algorithm 3.2 begins with $\bar{x} = (\bar{x}_i, \bar{x}_j)$ and $z' = c_i\bar{x}_i + c_j\bar{x}_j$. The ratio algorithm checks whether there exists a variable index h with $\beta_h < \beta_i < \beta_j \leq 2M$, $\bar{x} = (\bar{x}_h, \bar{x}_j)$, and $z = c_h\bar{x}_h + c_j\bar{x}_j$ such that $z < z'$. If so, then variable index i does not define an optimal solution to the corresponding S2CLP. Similarly, the ratio algorithm identifies a variable index k with $\beta_i < \beta_j < \beta_k \leq 2M$, $\bar{x} = (\bar{x}_i, \bar{x}_k)$, and $z = c_i\bar{x}_i + c_k\bar{x}_k$. If $z < z'$, then variable index j does not define an optimal solution to S2CLP. This argument can be easily extended to a variable index g such that $\beta_g < \beta_h < \beta_i < \beta_j \leq 2M$ or a variable index l such that $\beta_i < \beta_j < \beta_k < \beta_l \leq 2M$. If $z = c_g\bar{x}_g + c_j\bar{x}_j < z'$, then both h and i do not define an optimal solution. Similarly, if $z = c_i\bar{x}_i + c_l\bar{x}_l < z'$, then j and k do not define an optimal solution to S2CLP. Observe that RA also eliminates variables that have $z < z'$, but violate the nonnegativity conditions.

To prove that RA requires $O(n \log n)$ effort, observe that the values of β_i are calculated in $O(n)$ time and sorted in $O(n \log n)$ time. Determining variable index i such that $\beta_i < \frac{b_2}{b_1} \leq \beta_{i+1}$ requires $O(n)$ effort. The algorithm determines if an S2CLP is infeasible in constant time. Each computation within the while loop requires $O(1)$ effort. Observe that every step of the while loop either concludes $z \not< z'$ or calculates a new \bar{x} and updates variable indices i or j because $z < z'$. If $z < z'$, then every variable index between i and j does not define an optimal solution to S2CLP due to the aforementioned arguments, and the loop no longer visits these indices. Thus, using amortized analysis the while loop requires $O(n)$

effort. Reporting an optimal solution and the variable indices that define an optimal basis takes $O(1)$. Consequently, the theoretical computational complexity of RA is bounded by sorting a list of variable indices, which is $O(n \log n)$.

Observe that $\binom{n}{2}$ bases exist in every two constraint linear program, and two primary methods exist to find an optimal basis to such a simple problem. Finding an optimal basis by enumerating all possible bases and checking the reduced cost for each nonbasic variable requires $O(n^3)$ effort. The simplex method can also find an optimal basis to LPs with only two constraints, but its worst case complexity for this class of problems is also $O(n^3)$. Since RA only requires $O(n \log n)$ effort, this technique is the theoretically fastest method to solve these simple LPs with only two constraints.

Example 3.3.1 presents a 2CLP and demonstrates the implementation of RA to solve this LP with only two constraints.

Example 3.3.1. *Consider the following 2CLP.*

$$\begin{aligned}
 &\text{minimize } z = 12x_1 + 8x_2 + 12x_3 + 8x_4 + 30x_5 + 20x_6 + 57x_7 + 6x_8 + 25x_9 + 10x_{10} \\
 &\text{subject to } \quad -3x_1 \quad + 2x_3 + 4x_4 - 3x_5 + 3x_6 + 7x_7 + x_8 + x_9 + 2x_{10} \geq 12 \\
 &\quad \quad \quad 2x_1 + x_2 + x_3 - x_4 - 10x_5 + x_6 + 3x_7 \quad + x_9 - x_{10} \geq 5 \\
 &\quad \quad \quad x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \geq 0
 \end{aligned}$$

The ratio algorithm begins by creating the following S2CLP from the above 2CLP and calculates β_i for each $i \in N'$:

$$\begin{aligned}
 &\text{minimize } z = 12x_1 + 8x_2 + 12x_3 + 8x_4 + 30x_5 + 20x_6 + 57x_7 + 6x_8 + 25x_9 + 10x_{10} \\
 &\text{subject to } \quad -3x_1 \quad + 2x_3 + 4x_4 - 3x_5 + 3x_6 + 7x_7 + x_8 + x_9 + 2x_{10} - x_{11} = 12 \\
 &\quad \quad \quad 2x_1 + x_2 + x_3 - x_4 - 10x_5 + x_6 + 3x_7 \quad + x_9 - x_{10} \quad - x_{12} = 5 \\
 &\quad \quad \quad x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \geq 0.
 \end{aligned}$$

Therefore, $\beta_1 = M + \frac{3}{2}$, $\beta_2 = M$, $\beta_3 = \frac{1}{2}$, $\beta_4 = -M - \frac{1}{4}$, $\beta_5 = 3M$, $\beta_6 = \frac{1}{3}$, $\beta_7 = \frac{3}{7}$, $\beta_8 = -M$, $\beta_9 = 1$, $\beta_{10} = -M - \frac{1}{2}$, $\beta_{11} = 2M$, and $\beta_{12} = -2M$. The ratio algorithm sorts all

variables in a non-descending order according to their β_i values and creates a list of sorted variables $K = (12, 10, 4, 8, 6, 7, 3, 9, 2, 1, 11, 5)$. The algorithm searches for an $i' \in N'$ such that $\beta_{\kappa_{i'}} < \frac{b_2}{b_1} \leq \beta_{\kappa_{i'+1}}$. Since $\beta_6 = \frac{1}{3} < \frac{b_2}{b_1} = \frac{5}{12} \leq \beta_7 = \frac{3}{7}$, the algorithm determines $i' = 5$ and $j' = 6$. Observe that none of the conditions for an infeasible S2CLP are satisfied because $\beta_6 = \frac{1}{3}$ and $\beta_7 = \frac{3}{7}$. Thus, the algorithm follows and assign i to 5 and j to 6.

The algorithm calculates $\bar{x} = (\bar{x}_{\kappa_{i'}}, \bar{x}_{\kappa_{j'}}) = (\bar{x}_6, \bar{x}_7) = (\frac{1}{2}, \frac{3}{2})$ along with $z' = c_6\bar{x}_6 + c_7\bar{x}_7 = 95\frac{1}{2}$, decreases i to 4, and calculates $\bar{x} = (\bar{x}_{\kappa_i}, \bar{x}_{\kappa_{j'}}) = (\bar{x}_8, \bar{x}_7) = (\frac{1}{3}, \frac{5}{3})$ along with $z = c_8\bar{x}_8 + c_7\bar{x}_7 = 97$. Since $z = 97 \not< z' = 95\frac{1}{2}$, no changes are made to i' and j . The ratio algorithm follows by increasing j to 7 and calculating $\bar{x} = (\bar{x}_{\kappa_{i'}}, \bar{x}_{\kappa_j}) = (\bar{x}_6, \bar{x}_3) = (2, 3)$ along with $z = c_6\bar{x}_6 + c_3\bar{x}_3 = 76$. Since $z = 76 < z' = 95\frac{1}{2}$, j' becomes 7, i returns to 5, and z' becomes 76.

The algorithm decreases i to 4 and calculates $\bar{x} = (\bar{x}_{\kappa_i}, \bar{x}_{\kappa_{j'}}) = (\bar{x}_8, \bar{x}_3) = (2, 5)$ along with $z = c_8\bar{x}_8 + c_3\bar{x}_3 = 72$. Since $z = 72 < z' = 76$, i' becomes 4 and j continues at 7. Following RA, j increases to 8, $\bar{x} = (\bar{x}_{\kappa_{i'}}, \bar{x}_{\kappa_j}) = (\bar{x}_8, \bar{x}_9) = (7, 5)$, $z = c_8\bar{x}_8 + c_9\bar{x}_9 = 167$, and no changes occur to j' and i because $z = 167 \not< z' = 72$. When i decreases to 3, $\bar{x} = (\bar{x}_{\kappa_i}, \bar{x}_{\kappa_{j'}}) = (\bar{x}_4, \bar{x}_3) = (\frac{1}{3}, \frac{16}{3})$ and $z = c_4\bar{x}_4 + c_3\bar{x}_3 = 66\frac{2}{3}$. Since $z = 66\frac{2}{3} < z' = 72$, i' becomes 3, j returns to 7, and z' becomes $66\frac{2}{3}$.

The reader can easily continue with RA and verify that the following \bar{x} 's and z 's become: $\bar{x} = (\bar{x}_4, \bar{x}_9) = (\frac{7}{5}, \frac{32}{5})$ and $z = 171\frac{1}{5}$; $\bar{x} = (\bar{x}_{10}, \bar{x}_3) = (\frac{1}{2}, \frac{11}{2})$ and $z = 71$; $\bar{x} = (\bar{x}_4, \bar{x}_2) = (3, 8)$ and $z = 88$; $\bar{x} = (\bar{x}_{12}, \bar{x}_3) = (1, 6)$ and $z = 72$; $\bar{x} = (\bar{x}_4, \bar{x}_1) = (\frac{39}{5}, \frac{32}{5})$ and $z = 139\frac{1}{5}$; $\bar{x} = (\bar{x}_4, \bar{x}_{11}) = (-5, -32)$ and $z = -40$; $\bar{x} = (\bar{x}_4, \bar{x}_5) = (\frac{105}{43}, -\frac{32}{43})$ and $z = -2\frac{34}{43}$. Observe that $z \not< z' = 66\frac{2}{3}$ for all z 's, except the last two cases. However, these two cases have $\bar{x}_4 \not\geq 0$, $\bar{x}_{11} \not\geq 0$, and $\bar{x}_5 \not\geq 0$, which violates the nonnegative conditions. Consequently, RA reports $z^* = 66\frac{2}{3}$, $x^* = (0, 0, \frac{16}{3}, \frac{1}{3}, 0, 0, 0, 0, 0, 0, 0, 0)$, $i^* = 4$, $j^* = 3$, $\beta_{i^*} = -M - \frac{1}{4}$, and $\beta_{j^*} = \frac{1}{2}$.

Similarly to SA, i^* , j^* , β_{i^*} , and β_{j^*} are not part of the optimal solution of any S2CLP. However, this information is critical to implement RA within a dual simplex framework. The next chapter presents the results when SA and RA are both implemented to perform two dimensional searches in a simplex framework.

Chapter 4

Two Dimensional Searches in a Simplex Framework

This chapter presents theoretical and computational results when two dimensional searches are implemented within a simplex framework. Chapter 4.1 describes the basic knowledge of a simplex framework. Chapter 4.2 presents the double pivot simplex method, a primal simplex framework in which the slope algorithm determines up to two basic leaving variables. Chapter 4.3 shows the double pivot dual simplex method, a dual simplex framework where the ratio algorithm finds up to two nonbasic entering variables.

The primary content of this chapter is based on the journal paper, “The Double Pivot Simplex Method”, published in *Mathematical Methods of Operations Research* (Vitor and Easton, 2018b). Additionally, some theoretical results are retrieved from the conference paper, “The Ratio Algorithm to Solve the Optimal Basis of Two Constraint Linear Programs”, published in the *Proceedings of the 2018 IISE Annual Conference* (Vitor, 2018).

4.1 The Primal and Dual Simplex Frameworks

This section presents the mathematical concept of a simplex framework. Both the primal and dual simplex frameworks are discussed based on the well-known simplex method and

the dual simplex method. These techniques are well documented in the literature and are taught in numerous undergraduate and graduate courses in various disciplines worldwide. Additional information can also be found in [Winston \(2004\)](#), [Bazaraa et al. \(2010\)](#), and [Hillier and Lieberman \(2015\)](#).

Let a standard linear program (SLP) be an LP where all inequalities are converted into linear equations. This typically requires adding a slack, surplus, or artificial variable to each constraint. Mathematically, an SLP is defined as:

$$\begin{aligned} & \text{maximize } z = c^T x \\ & \text{subject to } \quad Ax = b \\ & \quad \quad \quad x \geq 0. \end{aligned}$$

For this chapter, assume $c \in \mathbb{R}^{n+r}$, $x \in \mathbb{R}^{n+r}$, $A \in \mathbb{R}^{r \times (n+r)}$, and $b \in \mathbb{R}^r$. Furthermore, let $N = \{1, 2, \dots, n+r\}$ be the set of variable indices, $R = \{1, 2, \dots, r\}$ be the set of constraint indices, and $S = \{x \in \mathbb{R}_+^{n+r} : Ax = b\}$ be the feasible region of an SLP. Observe that in this case, c is augmented with r zeros while A is augmented with an $r \times r$ identity matrix.

Formally, $BV \subseteq N$ is said to be a basis if $|BV| = |R|$ and $A_{,BV}$ is nonsingular. The set of nonbasic indices is $NBV = N \setminus BV$. The corresponding basic and nonbasic variables are x_{BV} and x_{NBV} , respectively, where $x_{BV} = A_{,BV}^{-1}b$ and $x_{NBV} = 0$. If $A_{,BV}^{-1}b \geq 0$, then BV is a feasible basis with x_{BV} and x_{NBV} being the corresponding basic feasible solution. Moreover, if BV is feasible and $c_i^\pi = c_{BV}^T A_{,BV}^{-1} A_{,i} - c_i \geq 0$ for each $i \in NBV$ where c^π is the calculated reduced cost, then BV is an optimal basis.

This dissertation follows the common notation that a period “.” represents all the columns of a given matrix and a colon “:” denotes all the rows of a given matrix or vector. Furthermore, a set as a subscript restricts the matrix or vector to only those indices of the set. For instance, $A_{,BV}$ represents the columns of A restricted to the indices in BV , $A_{,BV}$ corresponds to the rows of A restricted to the indices in BV , and x_{BV} is the x values of the indices in BV . Since a simplex framework exchanges elements in BV with elements in NBV , order is important and every basis BV in the remainder of this dissertation is viewed

as an r tuple. Moreover, the process of exchanging basic variables with nonbasic variables is referred to as a pivot.

Prior to describing the primal and dual simplex frameworks, one should understand the primary steps of the simplex method and the dual simplex method. The simplex method (SM) starts with a basic feasible solution and moves toward optimality. That is, SM begins with an SLP and a feasible basis BV such that $c_{BV}^T A_{BV}^{-1} A_i - c_i < 0$ for some $i \in NBV$ and $(A_{BV}^{-1} b)_j \geq 0$ for all $j \in BV$. The first step of SM is to evaluate each nonbasic variable's reduced cost, $c_i^\pi = c_{BV}^T A_{BV}^{-1} A_i - c_i$ for all $i \in NBV$. If all nonbasic reduced costs are nonnegative, then BV is an optimal basis, and the corresponding basic feasible solution with $z = c_{BV}^T x_{BV}$ represents an optimal solution to SLP.

If BV is not optimal, then there exists an entering nonbasic variable with index $p \in NBV$ such that $c_p^\pi < 0$. Define $R^+ = \{j \in R : (A_{BV}^{-1} A_p)_j > 0\}$. If $R^+ = \emptyset$, then the problem is unbounded. If not, SM performs the minimum ratio test and identifies a $j^* \in R^+$ such that:

$$\frac{(A_{BV}^{-1} b)_{j^*}}{(A_{BV}^{-1} A_p)_{j^*}} \leq \frac{(A_{BV}^{-1} b)_j}{(A_{BV}^{-1} A_p)_j}$$

for all $j \in R^+$. The simplex method replaces the j^{*th} element in BV with p . This process is referred to as a classic pivot in this dissertation. The algorithm continues until an optimal basis to SLP is identified or SLP is shown to be unbounded.

In contrast, the dual simplex method (DSM) starts with an SLP and a super optimal basic solution. Let BV be a super optimal basis to SLP if $c_{BV}^T A_{BV}^{-1} A_i - c_i \geq 0$ for all $i \in NBV$. Frequently, a super optimal BV also represents an infeasible solution to SLP so that $(A_{BV}^{-1} b)_j < 0$ for some $j \in BV$. In this case, DSM moves toward feasibility. From SLP and BV , the dual simplex method first evaluates the solution of each basic variable, $(A_{BV}^{-1} b)_j$ for all $j \in BV$. If the value of all basic variables is nonnegative, then BV is feasible, and consequently, an optimal basis. Thus, the corresponding basic feasible solution with $z = c_{BV}^T x_{BV}$ represents an optimal solution to SLP.

If BV is not feasible, then there exists a leaving basic variable with index $p' \in BV$ such that $(A_{BV}^{-1} b)_{p'} < 0$. Let $N^- = \{i \in N : (A_{BV}^{-1} A_i)_{p'} < 0\}$. If $N^- = \emptyset$, then the

problem is infeasible. Otherwise, DSM identifies an $i^* \in N^-$ using an equivalent version of the minimum ratio test such that:

$$\left| \frac{c_{BV}^T A_{.BV}^{-1} A_{.i^*} - c_{i^*}}{(A_{.BV}^{-1} A_{.i^*})_{p'}} \right| \leq \left| \frac{c_{BV}^T A_{.BV}^{-1} A_{.i} - c_i}{(A_{.BV}^{-1} A_{.i})_{p'}} \right|$$

for all $i \in N^-$. The dual simplex method performs a classic pivot and replaces the p^{th} element in BV with i^* . The algorithm continues until a feasible basis to SLP is identified (an optimal basis) or SLP is shown to be infeasible.

Define a simplex framework as the basic steps of SM or DSM independent of the pivoting method. As an example, let consider the steps of SM but the same logic can be applied if DSM is chosen. A simplex framework starts with an SLP and a basis BV . The simplex framework determines a set of nonbasic variables $Q \subseteq NBV$. Thus, a new basis $BV^* \subset BV \cup Q$ is identified, and this basis replaces BV . In this case, the act of selecting Q and replacing BV with BV^* is referred to as a pivot. The simplex framework repeats this process until an optimal basis to SLP is obtained or SLP is shown to be unbounded.

To generalize the idea of a simplex framework for a primal and dual approach, observe that a primal simplex framework starts with an SLP and a feasible basis BV . The method identifies $Q \subseteq NBV$ such that $Q \neq \emptyset$ and $c_{BV}^T A_{.BV}^{-1} A_{.q} - c_q < 0$ for some $q \in Q$. The following multidimensional subspace linear program,

$$\begin{aligned} & \text{maximize } z = (c_Q^T - c_{BV}^T A_{.BV}^{-1} A_{.Q}) x_Q \\ & \text{subject to } (A_{.BV}^{-1} A_{.BV}) x_{BV} + (A_{.BV}^{-1} A_{.Q}) x_Q = A_{.BV}^{-1} b \\ & \quad x_{BV}, x_Q \geq 0, \end{aligned}$$

is solved. Let BV^* be an optimal basis to the above subproblem, and BV^* replaces BV . This process continues until an optimal basis to SLP is obtained.

On the other hand, a dual simplex framework starts with an SLP and a super optimal basis BV . The method identifies $Q' \subseteq BV$ such that $Q' \neq \emptyset$ and $(A_{.BV^{-1}b})_{q'} < 0$ for some $q' \in Q'$. Therefore, the following multidimensional subspace linear program,

$$\begin{aligned} \text{minimize } z &= (c_{NBV}^T - c_{BV}^T A_{.BV^{-1}} A_{.NBV}) x_{NBV} \\ \text{subject to } & (A_{.BV^{-1}} A_{.BV})_{:Q'} (x_{BV})_{:Q'} + (A_{.BV^{-1}} A_{.NBV})_{:Q'} (x_{NBV})_{:Q'} = (A_{.BV^{-1}b})_{:Q'} \\ & x_{BV}, x_{NBV} \geq 0, \end{aligned}$$

is solved such that BV^* is an optimal basis to this multidimensional subspace linear program. Thus, BV^* replaces BV and the process continues until a feasible basis, and consequently an optimal basis, to SLP is obtained. Observe that the optimization direction of these multidimensional subspace linear programs can be easily changed depending on whether SLP is a maximization or minimization problem.

Both SM and DSM are simplex frameworks that perform classic pivots. The algorithms exchange exactly one basic variable with a nonbasic variable at each iteration. That is, $|Q| = 1$ and $|Q'| = 1$. The simplex method identifies an entering nonbasic variable and uses the minimum ratio test to find a leaving basic variable. In this case, a one dimensional subspace linear program consists of $r + 1$ variables, r constraints, and $r + 1$ nonnegativity constraints. The $r + 1$ variables are the r basic variables and the one entering nonbasic variable. On the other hand, DSM identifies a leaving basic variable and uses an equivalent version of the minimum ratio test to determine an entering nonbasic variable. That is, a one dimensional subspace linear program has $n + 1$ variables, 1 constraint, and $n + 1$ nonnegative constraints. The $n + 1$ variables are the n nonbasic variables and the one leaving basic variable. In both cases, the minimum ratio test performs a one dimensional search to find a new solution at each iteration.

The following sections present novel simplex frameworks where up to two basic variables are exchanged with two nonbasic variables at every iteration. In this case, either SA or RA perform a two dimensional search at each step to identify a new solution.

4.2 The Double Pivot Simplex Method

The double pivot simplex method (DPSM) is a primal simplex framework to solve LPs. This novel technique has $|Q| = 2$, and a two dimensional subspace linear program is solved at each iteration. Observe that if a non-optimal basis is selected to this subproblem, then $BV \cup Q$ could remain unchanged, and the algorithm may never terminate. Consequently, the algorithms from [Shamos and Hoey \(1976\)](#), [Megiddo \(1983\)](#), or [Dyer \(1984\)](#) cannot be directly applied to solve these two dimensional subspace linear programs since none of these methods determine an optimal basis. On the other hand, SA can be implemented to solve DPSM's subproblems since SA determines both an optimal basis and an optimal solution to simple 2VLPs.

Prior to describing DPSM, one should question whether SA can identify an optimal basis to a 2VLP. [Theorem 3.2.3](#) proves that SA finds an optimal solution to a 2VLP, but not necessarily an optimal basis. To prove SA can also identify an optimal basis to any multidimensional subspace linear program with $|Q| = 2$, convert 2VLP into standard form (S2VLP) by adding r slack variables. Each constraint of an S2VLP has the form of:

$$a_{j,1}x_1 + a_{j,2}x_2 + x_{j+2} = b_j \quad \forall j \in R.$$

If implemented within a primal simplex framework, these slack variables represent the initial basic variables. Furthermore, the only nonzero reduced costs are c_1 and c_2 , which are both positive. The right-hand side is precisely the values of the basic variables, which are greater than or equal to zero. Removing the basic variables from this instance results in a 2VLP. [Theorem 4.2.1](#) proves that SA returns sufficient information to determine an optimal basis, assuming that S2VLP is bounded. Additionally, this result also provides an alternate proof to [Theorem 3.2.3](#).

Theorem 4.2.1. *Given a bounded 2VLP and a sufficiently large positive number M , the slope algorithm reports j^* , k^* , α_{j^*} , and α_{k^*} such that the following sets are an optimal basis for the corresponding S2VLP:*

i) *If $\alpha_{j^*} = -2M$, then $BV = \{3, 4, \dots, k^* + 1, 1, k^* + 3, \dots, r + 2\}$;*

ii) *If $\alpha_{k^*} = 2M$, then $BV = \{3, 4, \dots, j^* + 1, 2, j^* + 3, \dots, r + 2\}$;*

iii) *If $\alpha_{j^*} \neq -2M$ and $\alpha_{k^*} \neq 2M$, then $BV = \{3, 4, \dots, j^* + 1, 1, j^* + 3, \dots, k^* + 1, 2, k^* + 3, \dots, r + 2\}$.*

Proof. Given a bounded 2VLP and a sufficiently large positive number M , the slope algorithm reports j^* , k^* , α_{j^*} , and α_{k^*} . To prove BV is an optimal basis to S2VLP, all possible cases of α_{j^*} and α_{k^*} are examined.

Assume that $\alpha_{j^*} = -2M$, and let $BV = \{3, 4, \dots, k^* + 1, 1, k^* + 3, \dots, r + 2\}$ with $NBV = \{2, k^* + 2\}$. Since 2VLP is bounded, $-M < \alpha_{k^*} < M$, $a_{k^*,1} > 0$, and $a_{k^*,2} > 0$; therefore, $c_{k^*+2}^\pi = \frac{c_1}{a_{k^*,1}} > 0$ and $c_2^\pi = \frac{a_{k^*,2}c_1}{a_{k^*,1}} - c_2$, where c^π is the calculated reduced cost. From SA, $\frac{c_2}{c_1} \leq \alpha_{k^*}$ and the conditions of Corollary 3.2.2 are satisfied. Therefore, $c_2 a_{k^*,1} \leq c_1 a_{k^*,2}$, which implies $c_2^\pi \geq 0$. Since $a_{k^*,1} > 0$, then the columns of $BV = \{3, 4, \dots, k^* + 1, 1, k^* + 3, \dots, r + 2\}$ in S2VLP are linearly independent and therefore, BV is an optimal basis to S2VLP.

Assume that $\alpha_{k^*} = 2M$, $BV = \{3, 4, \dots, j^* + 1, 2, j^* + 3, \dots, r + 2\}$, and $NBV = \{1, j^* + 2\}$. Since S2VLP is bounded, $-M < \alpha_{j^*} < M$, $a_{j^*,1} > 0$, and $a_{j^*,2} > 0$; therefore, $c_{j^*+2}^\pi = \frac{c_2}{a_{j^*,2}} > 0$ and $c_1^\pi = \frac{a_{j^*,1}c_2}{a_{j^*,2}} - c_1$. From SA, $\alpha_{j^*} < \frac{c_2}{c_1}$ and the conditions of Corollary 3.2.1 are satisfied. Therefore, $c_1 a_{j^*,2} < c_2 a_{j^*,1}$, which implies $c_1^\pi > 0$. Since $a_{j^*,2} > 0$, then the columns of $BV = \{3, 4, \dots, j^* + 1, 2, j^* + 3, \dots, r + 2\}$ in S2VLP are linearly independent, and BV is an optimal basis to S2VLP.

Assume that $\alpha_{j^*} \neq -2M$, $\alpha_{k^*} \neq 2M$, $BV = \{3, 4, \dots, j^* + 1, 1, j^* + 3, \dots, k^* + 1, 2, k^* + 3, \dots, r + 2\}$, and $NBV = \{j^* + 2, k^* + 2\}$. The first step is to prove that the columns of BV in S2VLP are linearly independent. Since $\alpha_{j^*} < \frac{c_2}{c_1} \leq \alpha_{k^*} \leq 2M$, the conditions of Lemma 3.2.2 are satisfied. Thus, $a_{j^*,2} a_{k^*,1} < a_{j^*,1} a_{k^*,2}$ and so $a_{j^*,2} a_{k^*,1} - a_{j^*,1} a_{k^*,2} < 0$ (\dagger). Consequently, the columns of BV in S2VLP are linearly independent. One can verify that

$c_{k^*+2}^\pi = \frac{-c_2 a_{j^*,1} + c_1 a_{j^*,2}}{a_{j^*,2} a_{k^*,1} - a_{j^*,1} a_{k^*,2}}$. The conditions of Corollary 3.2.1 are satisfied, so $a_{j^*,2} c_1 < a_{j^*,1} c_2$. Combining this fact with (†) results in $c_{k^*+2}^\pi > 0$. Similarly, $c_{j^*+2}^\pi = \frac{c_2 a_{k^*,1} - c_1 a_{k^*,2}}{a_{j^*,2} a_{k^*,1} - a_{j^*,1} a_{k^*,2}}$. The conditions of Corollary 3.2.2 are satisfied, so $c_2 a_{k^*,1} \leq c_1 a_{k^*,2}$. Coupling this fact with (†) results in $c_{j^*+2}^\pi \geq 0$. Thus, BV is an optimal basis for S2VLP. \square

Given the primary results, DPSM (Algorithm 4.1) is presented within the context of a revised simplex framework. The reader can easily modify DPSM to create a dictionary or tableau version. The double pivot simplex method follows the spirit of Dantzig's rule and selects the two indices with the most negative reduced cost for the entering nonbasic variables. However, many of the pivoting rules discussed in Chapter 2.1.3 could have been applied in this case. The input to DPSM is an SLP, a feasible basis BV (typically the slack variables), and a sufficiently large positive number M . Observe that DPSM performs a classic pivot if there is only one negative reduced cost (lines 14-22).

In the absence of degeneracy (recall discussion in Chapter 2.1.2 and see additional details in Chapter 4.2.1), DPSM correctly solves an LP within a finite number of steps. Theorem 4.2.2 formalizes this claim.

Theorem 4.2.2. *Given a nondegenerate SLP, an initial feasible basis BV , and a sufficiently large positive number M , DPSM correctly terminates within a finite number of steps.*

Proof. Given a nondegenerate SLP, an initial feasible basis BV , and a sufficiently large positive number M , let z be the objective function value of the current basis and \hat{z} denote the objective function value after DPSM performs a pivot. The double pivot simplex method can perform one out of four types of pivots at each iteration. That is, one of the three double pivots or a classic pivot. If a double pivot is performed, then $c_p^\pi = c_p - c_{BV}^T A_{BV}^{-1} A_p > 0$, $c_q^\pi = c_q - c_{BV}^T A_{BV}^{-1} A_q > 0$, and $\hat{z} = z + c_p^\pi x_p^* + c_q^\pi x_q^*$ where (x_p^*, x_q^*) is an optimal solution to 2VLP. If a classic pivot is performed, then $c_p^\pi > 0$ and $\hat{z} = z + c_p^\pi \theta^*$ where θ^* is the final value of θ in Algorithm 4.1 (lines 14-22).

If a double pivot where two nonbasic variables enter the basis is performed, then $\alpha_{j^*} \neq -2M$ and $\alpha_{k^*} \neq 2M$, which results in $x_p^* > 0$ and $x_q^* > 0$. If a double pivot where the nonbasic variable with the most negative reduced cost enters the basis is performed, then

$\alpha_{j^*} = -2M$, $x_p^* > 0$, and $x_q^* = 0$. If a double pivot where the nonbasic variable with the second most negative reduced cost enters the basis is performed, then $\alpha_{k^*} = 2M$, which implies $x_p^* = 0$ and $x_q^* > 0$. If a classic pivot is performed, then $0 < \theta < M$. Consequently, either pivot results in $\hat{z} > z$ since SLP is nondegenerate.

Because each iteration of DPSM either pivots to an improved basic feasible solution or determines a ray of unboundedness, DPSM does not visit the same basis more than once. Since there are at most $\binom{n+r}{r}$ bases, DPSM terminates within a finite number of steps. \square

Algorithm 4.1 : The Double Pivot Simplex Method (DPSM)

```

1: begin
2:   while  $c_{BV}^T A_{.BV}^{-1} A_{.i} - c_i \not\geq 0$  for all  $i \in (N \setminus BV)$  do
3:      $p \leftarrow \operatorname{argmin}_{p \in (N \setminus BV)} c_{BV}^T A_{.BV}^{-1} A_{.p} - c_p$ ;
4:      $q \leftarrow \operatorname{argmin}_{q \in (N \setminus (BV \cup \{p\}))} c_{BV}^T A_{.BV}^{-1} A_{.q} - c_q$ ;
5:     if  $c_{BV}^T A_{.BV}^{-1} A_{.q} - c_q < 0$  then
6:       Let 2VLP be:
           maximize  $z = (c_p - c_{BV}^T A_{.BV}^{-1} A_{.p})x_p + (c_q - c_{BV}^T A_{.BV}^{-1} A_{.q})x_q$ 
           subject to  $(A_{.BV}^{-1} A_{.p})x_p + (A_{.BV}^{-1} A_{.q})x_q \leq A_{.BV}^{-1} b$ 
                     $x_p, x_q \geq 0$ ;
7:       Solve 2VLP with SA;
8:       if 2VLP is unbounded then return SLP is unbounded;
9:       if  $\alpha_{j^*} = -2M$  then  $BV_{k^*} \leftarrow p$ ;
10:      if  $\alpha_{k^*} = 2M$  then  $BV_{j^*} \leftarrow q$ ;
11:      if  $\alpha_{j^*} \neq -2M$  and  $\alpha_{k^*} \neq 2M$  then
12:         $BV_{j^*} \leftarrow p$ ;
13:         $BV_{k^*} \leftarrow q$ ;
14:      else
15:         $\theta \leftarrow M$ ;
16:        for each  $i \in R$  do
17:          if  $(A_{.BV}^{-1} A_{.p})_i > 0$  and  $\frac{(A_{.BV}^{-1} b)_i}{(A_{.BV}^{-1} A_{.p})_i} < \theta$  then
18:             $\theta \leftarrow \frac{(A_{.BV}^{-1} b)_i}{(A_{.BV}^{-1} A_{.p})_i}$ ;
19:             $l \leftarrow i$ ;
20:        if  $\theta = M$  then return SLP is unbounded;
21:        else
22:           $BV_l \leftarrow p$ ;
23:        return  $x_{BV}^* \leftarrow A_{.BV}^{-1} b$ ,  $x_{(N \setminus BV)}^* = 0$ , and  $z^* = c^T x^*$ ;
24: end

```

To assess the benefit of DPSM over SM, one should compare the improvement in objective function value and the theoretical effort per iteration of both techniques. Assume that both DPSM and SM have the same basic feasible solution. If one of DPSM's entering nonbasic variables is identical to the entering basic variable, x_p , from SM, then the objective function value from DPSM's pivot is at least as good as the objective function value from SM's pivot (see Chapter 5.2.1 for discussion). Furthermore, DPSM and SM only pivot to the same basis when SA returns $\alpha_{j^*} = -2M$.

The theoretical effort required by an iteration of SM with a classic pivot involves calculating $A_{.BV}^{-1}$ and identifying an improving nonbasic variable x_p , which is achieved by evaluating $c_{BV}^T A_{.BV}^{-1} A_{.i} - c_i$ for each $i \in NBV$. The minimum ratio test calculates $A_{.BV}^{-1} A_{.p}$, $A_{.BV}^{-1} b$, and performs a division. Given a revised simplex framework, changing the entering basic variable with the leaving basic variable requires $O(1)$ effort. Thus, the theoretical effort per iteration of SM is bounded by calculating the inverse and identifying the entering nonbasic and leaving basic variables. Consequently, each iteration of SM requires $O(rn + I(r))$ effort where $I(r)$ is the time required to find the inverse of an $r \times r$ matrix.

The theoretical effort per iteration of DPSM requires calculating $A_{.BV}^{-1}$ and identifying two improving nonbasic variables, x_p and x_q . These steps are nearly identical to SM and require identical theoretical effort. Thus, SA determines the leaving basic variables in $O(S(r))$ effort, where $S(r)$ is the time required to sort a set of r elements. Exchanging the basic variables is performed again in $O(1)$. Observe that $n \geq r$ due to the addition of slack or artificial variables, and SA's running time is dominated by $O(I(r))$. Therefore, each double pivot is restricted by calculating the inverse and identifying the entering nonbasic variables, which requires $O(rn + I(r))$ effort. Consequently, a double pivot and a classic pivot require the same theoretical effort per iteration, assuming a revised simplex framework.

As discussed in Chapter 2.1.1, state-of-the-art commercial and open source LP solvers do not calculate the inverse of a matrix at every iteration. Instead, solvers update the basis inverse matrix. Consequently, the theoretical running time of SM and DPSM in practice is $O(rn + U(r))$ where $U(r)$ is the effort required to update the basis factorization of a problem

with r constraints. Observe that developing an efficient technique to update the basis matrix inverse with two variables at a time is a critical future research topic.

Example 4.2.1 demonstrates the implementation of DPSM in a tableau format. Additionally, Table 4.1 presents the three tableaus that demonstrate DPSM's pivots.

Example 4.2.1. *Consider the following LP.*

$$\begin{aligned}
 &\text{maximize } z = 20x_1 + 12x_2 + 15x_3 + 6x_4 \\
 &\text{subject to} \quad x_1 - 2x_2 + 3x_3 + x_4 \leq 99 \quad (1) \\
 &\quad \quad \quad x_1 \quad \quad + x_3 \quad \quad \leq 40 \quad (2) \\
 &\quad \quad \quad 4x_1 + 9x_2 + x_3 + 4x_4 \leq 106 \quad (3) \\
 &\quad \quad \quad 2x_1 + 2x_2 + x_3 + x_4 \leq 60 \quad (4) \\
 &\quad \quad \quad 2x_1 - x_2 + 5x_3 \quad \leq 170 \quad (5) \\
 &\quad \quad \quad x_1, x_2, x_3, x_4 \geq 0
 \end{aligned}$$

The first tableau in Table 4.1 represents the above LP in standard form (SLP) and DPSM begins with a feasible basis $BV = \{5, 6, 7, 8, 9\}$. The nonbasic variables with the two most negative reduced costs correspond to x_1 and x_3 with $p = 1$ and $q = 3$. The slope algorithm solves the following 2VLP:

$$\begin{aligned}
 &\text{maximize } z = 20x_1 + 15x_3 \\
 &\text{subject to} \quad x_1 + 3x_3 \leq 99 \\
 &\quad \quad \quad x_1 + x_3 \leq 40 \\
 &\quad \quad \quad 4x_1 + x_3 \leq 106 \\
 &\quad \quad \quad 2x_1 + x_3 \leq 60 \\
 &\quad \quad \quad 2x_1 + 5x_3 \leq 170 \\
 &\quad \quad \quad x_1, x_3 \geq 0.
 \end{aligned}$$

The slope algorithm computes $\alpha = (3, 1, \frac{1}{4}, \frac{1}{2}, \frac{5}{2}, 2M, -2M)$ and identifies $\alpha_4 = \frac{1}{2} < \frac{c_3}{c_1} = \frac{15}{20} \leq \alpha_2 = 1$ with $\bar{x} = (x_1, x_3) = (20, 20)$. This point satisfies all constraints, and SA returns $j^* = 4, k^* = 2, \alpha_4 = \frac{1}{2}$, and $\alpha_2 = 1$. Due to the returned values, the nonbasic indices 1 and 3

replace the fourth and second elements in BV , resulting in $BV = \{5, 3, 7, 1, 9\}$. The second tableau in Table 4.1 demonstrates this double pivot's outcome.

The next iteration begins by identifying the nonbasic variables with the two most negative reduced costs, x_2 and x_4 , resulting in $p = 2$ and $q = 4$. The slope algorithm solves the following 2VLP:

$$\begin{aligned}
 & \text{maximize } z = 2x_2 + x_4 \\
 & \text{subject to } \quad 2x_2 + 3x_4 \leq 19 \\
 & \quad \quad \quad -2x_2 - x_4 \leq 20 \\
 & \quad \quad \quad 3x_2 + x_4 \leq 6 \\
 & \quad \quad \quad 2x_2 + x_4 \leq 20 \\
 & \quad \quad \quad 5x_2 + 3x_4 \leq 30 \\
 & \quad \quad \quad x_2, x_4 \geq 0.
 \end{aligned}$$

The slope algorithm calculates $\alpha = (\frac{3}{2}, 3M, \frac{1}{3}, \frac{1}{2}, \frac{3}{5}, 2M, -2M)$, identifies $\alpha_3 = \frac{1}{3} < \frac{c_4}{e_2} = \frac{1}{2} \leq \alpha_4 = \frac{1}{2}$, and assigns \bar{x} to $(-14, 48)$. The slope algorithm eventually determines that an optimal solution occurs at $x^* = (0, 6)$, obtained from constraints (3) and (6) that intersect at an optimal basis, and returns $j^* = 3$, $k^* = 6$, $\alpha_3 = \frac{1}{3}$, and $\alpha_6 = 2M$. Since $\alpha_{k^*} = 2M$, the nonbasic index 4 replaces the third element in BV , resulting in $BV = \{5, 3, 4, 1, 9\}$. The third tableau in Table 4.1 presents the result of this double pivot. Observe that there are no other nonbasic variables with negative reduced cost in the third tableau. Thus, $BV = \{5, 3, 4, 1, 9\}$ is an optimal basis to the corresponding SLP, and DPSM reports an optimal solution $z^* = 706$ and $x^* = (14, 0, 26, 6, 1, 0, 0, 0, 12)$.

In contrast, SM solves the LP from Example 4.2.1 in four iterations. From an initial feasible basis $BV = \{5, 6, 7, 8, 9\}$, SM successively moves to the following bases: $\{5, 6, 1, 8, 9\}$, $\{5, 6, 1, 3, 9\}$, $\{5, 2, 1, 3, 9\}$, and $\{5, 4, 1, 3, 9\}$. Consequently, DPSM performs 50% fewer iterations than SM in this example.

Table 4.1: *Double pivot simplex tableau from Example 4.2.1*

BV	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	RHS
		1	-20	-12	-15	-6	0	0	0	0	0
x_5	0	1	-2	3	1	1	0	0	0	0	99
x_6	0	1	0	1	0	0	1	0	0	0	40
x_7	0	4	9	1	4	0	0	1	0	0	106
x_8	0	2	2	1	1	0	0	0	1	0	60
x_9	0	2	-1	5	0	0	0	0	0	1	170
	1	0	-2	0	-1	0	10	0	5	0	700
x_5	0	0	2	0	3	1	-5	0	2	0	19
x_3	0	0	-2	1	-1	0	2	0	-1	0	20
x_7	0	0	3	0	1	0	2	1	-3	0	6
x_1	0	1	2	0	1	0	-1	0	1	0	20
x_9	0	0	5	0	3	0	-8	0	3	1	30
	1	0	1	0	0	0	12	1	2	0	706
x_5	0	0	-7	0	0	1	-11	-3	11	0	1
x_3	0	0	1	1	0	0	4	1	-4	0	26
x_4	0	0	3	0	1	0	2	1	-3	0	6
x_1	0	1	-1	0	0	0	-3	-1	4	0	14
x_9	0	0	-4	0	0	0	-14	-3	12	1	12

Observe that DPSM has three types of double pivots and Example 4.2.1 presents two of them. The first double pivot replaces two basic variables with two nonbasic variables. The second double pivot exchanges only one basic variable with one nonbasic variable, which implies that one of the entering nonbasic variables, x_2 , is also a “leaving” variable. In this pivot, the index that enters the basis corresponds to the nonbasic variable with the second most negative reduced cost, x_4 . The other type of double pivot ($\alpha_{j^*} = -2M$) corresponds exactly to a classic pivot and the variable with the most negative reduced cost is the only entering nonbasic variable.

If one of the nonbasic entering variables in DPSM is identical to SM, double pivots are guaranteed to improve the objective function value by at least as much as classic pivots. The classic pivot from the initial feasible basis $BV = \{5, 6, 7, 8, 9\}$ results in an objective function value of $z = 530$. Thus, the relative improvement of a double pivot is:

$$\left(\frac{700 - 0}{530 - 0} - 1 \right) \times 100\% = 32.1\%.$$

Performing a classic pivot from $BV = \{5, 3, 7, 1, 9\}$ results in an objective function value of $z = 704$. The second double pivot's relative improvement is:

$$\left(\frac{706 - 700}{704 - 700} - 1 \right) \times 100\% = 50\%.$$

Therefore, even if a double pivot has a single nonbasic entering and basic leaving variable, the benefit may still be substantial.

4.2.1 The Double Pivot Simplex Method and Degeneracy

Recall from Chapter 2.1.2 that degeneracy occurs when a basic feasible solution has at least one basic variable that equals zero. Solving degenerate LPs by using SM without implementing any finite pivoting rules may result in SM visiting different basic feasible solutions with an identical objective function value. This behavior may potentially lead to cycling. This section demonstrates that implementing SA instead of the minimum ratio test in a primal simplex framework diminishes some of the issues caused by degeneracy.

Theorem 3.2.3 guarantees that an optimal solution to an SA2VLP occurs at the intersection of the j^{th} and k^{th} constraints as long as $\alpha_j < \frac{c_2}{c_1}$, $\alpha_k \geq \frac{c_2}{c_1}$, SA2VLP $_{j,k}$ is bounded, and the point intersecting at j and k is feasible. The slope algorithm returns constraints j^* and k^* , which not only fulfill Theorem 3.2.3's conditions, but also satisfy $\alpha_{k^*} - \alpha_{j^*} \leq \alpha_k - \alpha_j$ for all j and k pairs of constraints that meet Theorem 3.2.3's conditions (Algorithm 3.1, line 5). The selection of these particular constraints results in an optimal basis according to Theorem 4.2.1.

Example 4.2.2 provides a degenerate 2VLP that helps explain this concept. The graphical representation of this 2VLP is presented in Figure 4.1.

Example 4.2.2. Consider the following 2VLP.

$$\begin{aligned}
& \text{maximize } z = 5x_1 + 4x_2 \\
& \text{subject to} \quad x_1 - x_2 \leq 3 \quad (1) \\
& \quad \quad \quad x_1 \leq 3 \quad (2) \\
& \quad \quad \quad 2x_1 + x_2 \leq 6 \quad (3) \\
& \quad \quad \quad 3x_1 + 2x_2 \leq 10 \quad (4) \\
& \quad \quad \quad x_1 + x_2 \leq 4 \quad (5) \\
& \quad \quad \quad 2x_1 + 3x_2 \leq 10 \quad (6) \\
& \quad \quad \quad x_1 + 2x_2 \leq 6 \quad (7) \\
& \quad \quad \quad \quad \quad x_2 \leq 3 \quad (8) \\
& \quad \quad \quad -x_1 + x_2 \leq 3 \quad (9) \\
& \quad \quad \quad -x_1 - x_2 \leq 0 \quad (10) \\
& \quad \quad \quad x_1, x_2 \geq 0
\end{aligned}$$

Solving the above 2VLP with SA results in $z^* = 18$, $x^* = (2, 2)$, $j^* = 4$, $k^* = 5$. Observe that $\alpha_4 = \frac{2}{3} < \frac{c_2}{c_1} = \frac{4}{5} < \alpha_5 = 1$. There are six pairs of constraints that satisfy Theorem 3.2.3 and identify an optimal solution: (3) and (5); (3) and (6); (3) and (7); (4) and (5); (4) and (6); (4) and (7). The reader can easily convert the above 2VLP into an S2VLP and verify that any one of these six bases is also an optimal basis to the 2VLP. The slope algorithm chooses constraints (4) and (5) because this pair minimizes $\alpha_k - \alpha_j$ ($\alpha_5 - \alpha_4 = \frac{1}{3}$). Therefore, SA identifies an optimal basis even for degenerate 2VLPs.

One can see that the following four pairs of constraints also identify an optimal solution to the corresponding 2VLP: (3) and (4); (5) and (6); (5) and (7); (6) and (7). However, none of the bases corresponding to these pairs of constraints identify an optimal basis. Observe that the algorithms from Shamos and Hoey (1976), Megiddo (1983), or Dyer (1984) all solve a two variable linear program, but these techniques may end with any one of these four pairs of constraints intersecting at an optimal solution. Consequently, these fast methods do not always identify an optimal basis and cannot be used as a double pivoting strategy

within a primal simplex framework as previously mentioned. Additionally, recall that all these methods may not necessarily find an optimal basis to the corresponding dual of the above 2VLP, which led to the development of RA.

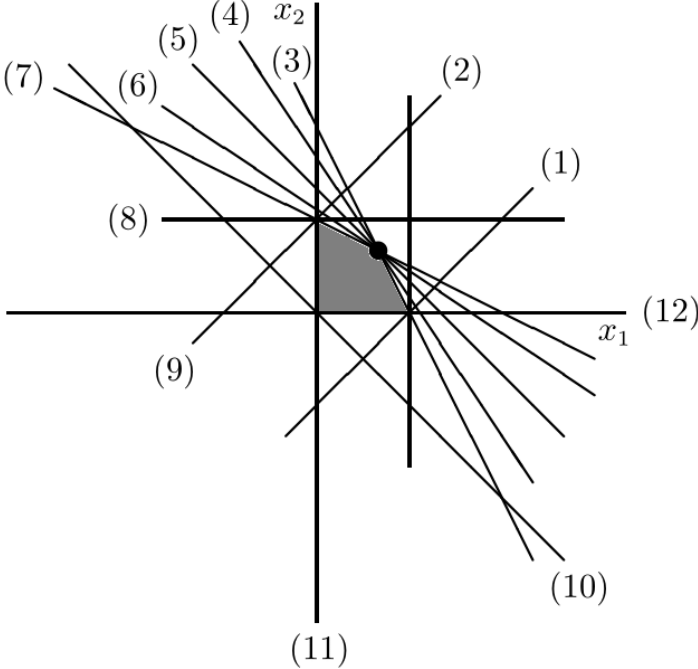


Figure 4.1: Graphical representation of Example 4.2.2

In contrast, implementing classic pivots on this degenerate 2VLP demonstrates a weakness of SM. Performing four classic pivots by partially applying Bland’s rule to resolve ties in the minimum ratio test results in a basis with constraints (5) and (6). Even though this basis identifies an optimal solution, it does not identify an optimal basis as aforementioned, and one more pivot is required to obtain an optimal basis. Therefore, degeneracy caused additional work for SM, but not for DPSM.

To determine whether or not DPSM performs better than SM on degenerate LPs that cycle, consider the 11 instances summarized by Gass and Vinjamuri (2004). These 11 problems cycle when solved with SM without any anti-cycling technique. However, DPSM identifies an optimal solution (or an unbounded LP) in every one of these 11 LPs. Therefore, DPSM avoids cycling on these frequently demonstrated degenerate LPs.

In conclusion, DPSM handles the issues caused by degenerate linear programming problems more effectively than SM. However, one should not infer from this claim that DPSM completely eliminates all issues with respect to degeneracy. Determining whether or not DPSM cycles is an unresolved research question.

4.2.2 Computational Study

The previous sections presented DPSM and provided evidence that it is theoretically superior to SM. This section discusses some experiments to determine whether or not DPSM is computationally faster than SM. The study was performed on an Intel[®] Core[™] i7-6700 3.4GHz processor with 32 GB of RAM, and implemented in C++. A portion of this study implemented DPSM and SM with CPLEX Version 12.7, a high performance mathematical programming solver (IBM, 2016). In this case, CPLEX's preprocessing operations were turned off in order to measure the real effectiveness of DPSM over SM and CPLEX's primal simplex algorithm.

4.2.2.1 Preliminary Implementation and Results

The first experiment of this computation study implemented DPSM and SM explicitly. The code not only computed $A_{.BV}^{-1}$ at every iteration using the LU decomposition package from Press et al. (2007), but also determined explicitly the reduced cost $c_i^\pi = c_{BV}^T A_{.BV}^{-1} A_{.i} - c_i$ for all nonbasic variables $i \in NBV$, the right-hand side $A_{.BV}^{-1} b$, and the constraint values $A_{.BV}^{-1} A_{.p}$ and $A_{.BV}^{-1} A_{.q}$ of both improving nonbasic variables.

To test DPSM and SM's explicit implementations, dense and sparse random instances were generated. These random LPs take the form of:

$$\begin{aligned}
 & \text{maximize } z = \sum_{i=1}^n c_i x_i \\
 & \text{subject to } \sum_{i=1}^n a_{j,i} x_i \leq b_j \quad \forall j \in \{1, 2, \dots, r\} \\
 & \quad \quad \quad x_i \geq 0 \quad \quad \quad \forall i \in \{1, 2, \dots, n\}.
 \end{aligned}$$

Dense random LPs have each $a_{j,i} \in \mathbb{Z}_+$ randomly generated and uniformly distributed between 0 and 1,000, $b_j = \lfloor \frac{1}{2} \sum_{i=1}^n a_{j,i} \rfloor$, and $c_i = \sum_{j=1}^r a_{j,i} + \lfloor 200\gamma_i \rfloor$ where γ_i is a uniform random number between 0 and 1. Sparse random LPs have every $a_{j,i} = 1$ if a uniform random number between 0 and 1 is greater than or equal to ξ , where ξ is a sparseness coefficient; or $a_{j,i} = 0$ otherwise. Right-hand side and cost coefficient values for sparse LPs are generated identically as dense LPs. Observe that these random instances are similar to the problems suggested by [Chu and Beasley \(1998\)](#).

These random instances were solved with 200, 400, 600, 800, and 1,000 variables and 100, 200,..., 1,000 constraints. To avoid random anomalies with respect to the randomly generated problems, 20 different instances were solved for each combination of variables and constraints. Tables [4.2-4.4](#) present the average percentage improvement in the number of pivots and solution time obtained with DPSM over SM for all 20 problems of each size. Improvement in the number of pivots and solution time is defined as:

$$\delta_{\frac{\text{DPSM}}{\text{SM}}} = \left(\frac{y_{\text{SM}} - y_{\text{DPSM}}}{y_{\text{SM}}} \right) \times 100\%,$$

where y_{DPSM} is the number of pivots or solution time performed by DPSM and y_{SM} indicates the number of pivots or solution time performed by SM. Observe that Tables [4.2-4.4](#) total 3,000 instances solved with each DPSM and SM.

Table [4.2](#) shows the results for dense random LPs. Tables [4.3](#) and [4.4](#) provide the computational results for sparse random LPs, and these problems have a sparseness coefficient of $\xi = 0.75$ and $\xi = 0.90$, respectively. When dense random LPs are analyzed, DPSM averages 17.3% fewer pivots and 17.2% less computational time. Evaluating sparse random LPs with $\xi = 0.75$ results in an average improvement in the number of pivots of approximately 29.9% and solution time of nearly 29.7%. If sparse random LPs with $\xi = 0.90$ are considered, the number of pivots is decreased by 29.5% and solution time by 29.4%, on average. Observe that there is a high correlation between the number of pivots and solution time. This follows the theoretical run time analysis, which implies that implementing SA instead of the minimum ratio test had a negligible impact on the total running time of these experiments.

Table 4.2: *Improvement in the number of pivots and solution time of DPSM over SM (Dense Random LPs)*

Cont/Var	Average % Improvement Number of Pivots					Average % Improvement Solution Time				
	200	400	600	800	1,000	200	400	600	800	1,000
100	19.3	19.1	16.9	15.7	18.6	18.8	18.6	16.6	15.4	18.6
200	18.9	20.5	16.7	17.4	17.4	18.7	20.4	16.7	17.4	17.4
300	20.0	17.7	17.9	16.6	16.1	19.9	17.7	17.8	16.6	15.9
400	21.0	16.3	17.0	16.5	16.8	20.8	16.2	16.9	16.4	16.8
500	18.6	15.9	15.9	16.8	14.9	18.5	15.8	16.2	16.7	14.9
600	19.7	16.8	16.8	17.0	16.0	19.6	16.8	16.7	16.9	15.9
700	17.4	17.2	16.5	17.1	15.7	17.1	17.0	16.3	16.8	15.4
800	19.0	19.5	16.7	15.0	16.2	18.8	19.3	16.5	14.6	16.0
900	19.2	16.7	16.9	15.0	15.9	19.1	16.7	17.0	15.1	16.0
1,000	19.5	19.8	15.7	16.0	14.6	19.4	19.6	15.6	16.0	14.5
Average	19.3	18.0	16.7	16.3	16.2	19.1	17.8	16.6	16.2	16.1
	17.3%					17.2%				

Table 4.3: *Improvement in the number of pivots and solution time of DPSM over SM (Sparse Random LPs, $\xi = 0.75$)*

Cont/Var	Average % Improvement Number of Pivots					Average % Improvement Solution Time				
	200	400	600	800	1,000	200	400	600	800	1,000
100	28.6	27.4	24.4	20.7	26.2	28.0	26.6	24.4	20.5	26.1
200	33.8	28.2	30.5	27.2	27.2	33.5	28.0	30.3	27.1	27.1
300	35.4	29.0	27.4	30.6	27.7	35.3	29.0	27.3	30.5	27.6
400	31.1	30.8	34.4	28.0	27.1	30.9	30.7	34.3	27.9	27.0
500	31.3	30.6	31.6	26.3	28.5	31.2	30.3	31.6	26.2	28.5
600	34.2	33.4	28.9	31.1	28.0	34.3	33.4	28.9	31.3	27.8
700	31.6	32.7	33.7	30.9	30.1	31.5	32.7	33.7	30.8	30.0
800	28.2	34.8	32.0	31.0	28.6	28.0	35.0	31.9	30.8	28.4
900	24.6	34.4	32.1	31.5	29.0	24.4	34.4	32.0	31.2	28.6
1,000	22.5	35.7	32.2	29.9	27.8	22.5	35.6	32.4	29.4	27.6
Average	30.1	31.7	30.7	28.7	28.1	30.0	31.6	30.7	28.6	27.9
	29.9%					29.7%				

Table 4.4: *Improvement in the number of pivots and solution time of DPSM over SM (Sparse Random LPs, $\xi = 0.90$)*

Cont/Var	Average % Improvement Number of Pivots					Average % Improvement Solution Time				
	200	400	600	800	1,000	200	400	600	800	1,000
100	33.0	26.2	29.3	21.0	22.9	33.4	26.1	29.0	20.9	22.8
200	30.2	28.3	24.2	28.1	26.1	30.1	28.2	24.1	28.1	26.1
300	31.2	28.0	27.5	29.3	29.8	31.2	27.9	27.4	29.2	29.7
400	33.9	32.7	30.8	27.4	27.0	33.8	32.7	30.7	27.3	26.9
500	34.2	32.8	30.2	29.1	25.4	33.9	33.0	30.1	29.1	25.2
600	34.4	32.6	32.7	28.3	28.0	34.2	32.5	32.6	28.1	27.8
700	30.1	34.4	29.7	27.7	26.5	29.9	34.2	29.5	27.5	26.3
800	27.7	32.9	31.4	28.7	26.5	27.7	32.7	31.2	28.6	26.1
900	26.9	34.3	31.3	30.2	30.4	26.7	34.1	31.8	29.8	30.3
1,000	25.4	32.5	30.8	31.7	32.0	25.5	32.5	30.7	31.8	31.8
Average	30.7	31.5	29.8	28.1	27.5	30.6	31.4	29.7	28.0	27.3
	29.5%					29.4%				

These 3,000 instances were also run with CPLEX’s primal simplex algorithm. In this case, CPLEX surpassed both DPSM and SM’s explicit implementations and solved the instances in a few seconds, while DPSM and SM required hours. This result is not surprising as explicitly implementing DPSM or SM eliminates decades of computational advancements. Unlike researchers at commercial optimization companies, the author does not have sufficient access to change the pivoting strategy in commercial software from a classic pivot to a double pivot. However, CPLEX has several API routines that allow the manipulation of the basis variables. These routines enabled this computational study to mimic double pivots over CPLEX’s infrastructure and take advantage of numerous computational advancements offered by the solver. The following sections provide details on this implementation and the computational results obtained.

4.2.2.2 CPLEX’s Implementation Details

Among several of the API routines offered by CPLEX, this computational study used four of them. Entering nonbasic variable for SM was obtained from the *CPXgetdj* routine. Routines *CPXgetx* and *CPXbinvacol* were used to perform the minimum ratio test and determine the leaving basic variable. The *CPXpivot* routine updated the basis by swapping the leaving basic variable with the entering nonbasic variable.

A similar implementation was followed for DPSM. Both entering nonbasic variables were obtained from the *CPXgetdj* routine. Two calls to the *CPXbinvacol* routine and one call to the *CPXgetx* routine created a 2VLP. The slope algorithm solved the 2VLP and identified the leaving and entering variable(s). Unfortunately, the *CPXpivot* routine only allows for a single exchange of variables at a time. Thus, if two nonbasic variables entered the basis, the *CPXpivot* routine was called twice. If either of the other type of pivots occurred, the *CPXpivot* routine was called once.

Both DPSM and SM require an initial feasible basis, which can be obtained either from a Big- M or Phase 1 implementation using DPSM or SM (Bazaraa et al., 2010). In this computational study, CPLEX’s Phase 1 reported an initial feasible basis. The same basis was used for both DPSM and SM in every instance solved.

Solution times obtained with these implementations became comparable with CPLEX’s primal simplex algorithm. Obviously, these implementations were still slower than CPLEX, but it only slowed the solution time by less than 50%. With a reasonable implementation of DPSM and SM, the study solved instances from Netlib (Gay, 1985) and MIPLIB (Koch et al., 2011), benchmark libraries with linear and mixed integer linear programs, respectively. Instances from MIPLIB were changed to LPs by eliminating the integrality constraints. To avoid the upper and lower bound simplex implementations, additional constraints were added to the problems in order to represent these bounds. Number of constraints, variables, and nonzeros shown in Tables 4.5-4.7 account for these transformations.

During the computational experiments, the author encountered serious issues with numerical instability from the *CPXpivot* routine. This issue prohibited DPSM and SM from

terminating and/or led to numerically singular bases for many of the benchmark instances. When a basis is found to be singular, CPLEX removes one or more variables from the current basis and re-includes these variables on further iterations when an optimal basis is obtained. If after re-including these variables the basis is no longer optimal, then CPLEX proceeds until an optimal basis is found; otherwise, an optimal solution and optimal basis to the problem has been found.

Unfortunately, the author could not duplicate CPLEX's internal repair process in the implementation of DPSM or SM. Failure to correctly repair the basis may result in incorrect and/or illegal pivots. The author attempted to fix this problem through more frequent refactoring of the basis (*CPX_PARAM_REINV*), restricting the number of times CPLEX repairs the basis (*CPX_PARAM_SINGLIM*), and tracking the kappa value to identify when a pivot makes the basis become unstable. However, none of these attempts resolved the aforementioned issues.

The reader should know that CPLEX's numerical instability occurred in both DPSM and SM. The computational experiments found problems where DPSM solved, but SM did not, and vice versa. Therefore, this computational study only reports the results of the benchmark problems where both DPSM and SM solved the instance with CPLEX's API routines.

4.2.2.3 CPLEX's Results and Analysis

Tables 4.5 and 4.6 show all instances solved from Netlib. Table 4.7 presents all instances solved from MIPLIB. Overall, 46 instances were solved from Netlib and 10 from MIPLIB. Because of the number of instances from Netlib, results to this benchmark library are divided in two tables, and the average of all problems is presented in Table 4.6. These tables present the number of pivots (Phase 2 only) performed by each method, including CPLEX's primal simplex implementation. Tables also include the number of constraints, variables, and nonzeros of each instance. Furthermore, Tables 4.5-4.7 describe the percentage improvement

obtained with DPSM over SM and DPSM over CPLEX. Similar to $\delta_{\frac{\text{DPSM}}{\text{SM}}}$, improvement in the number of pivots of DPSM over CPLEX's primal simplex algorithm is defined as:

$$\delta_{\frac{\text{DPSM}}{\text{CPLEX}}} = \left(\frac{y_{\text{CPLEX}} - y_{\text{DPSM}}}{y_{\text{CPLEX}}} \right) \times 100\%,$$

where y_{DPSM} is the number of pivots performed by DPSM and y_{CPLEX} equals the number of pivots performed by CPLEX's primal simplex algorithm.

Table 4.5: *Improvement in the number of pivots of DPSM over SM and CPLEX's primal simplex algorithm (Netlib)*

Name	Cont	Var	Nz	Number of Pivots			$\delta_{\frac{\text{DPSM}}{\text{SM}}}$	$\delta_{\frac{\text{DPSM}}{\text{CPLEX}}}$
				CPLEX	SM	DPSM		
ADLITTLE	56	138	424	58	100	56	44.0%	3.4%
AGG	488	615	2,862	64	85	45	47.1%	29.7%
AGG2	516	758	4,740	85	87	55	36.8%	35.3%
AGG3	516	758	4,758	56	64	34	46.9%	39.3%
BANDM	305	472	2,494	227	305	152	50.2%	33.0%
BEACONFD	173	295	3,408	88	95	52	45.3%	40.9%
BRANDY	220	303	2,202	194	271	208	23.2%	-7.2%
CZPROB	1,158	3,791	11,166	1,122	3,741	1,681	55.1%	-49.8%
DEGEN2	444	757	4,201	1,041	6,771	1,145	83.1%	-10.0%
E226	223	472	2,768	247	435	231	46.9%	6.5%
FFFFF800	524	1,028	6,401	216	278	135	51.4%	37.5%
FINNIS	619	1,141	2,959	277	670	344	48.7%	-24.2%
FIT1D	1,050	2,075	15,479	911	1,064	847	20.4%	7.0%
FIT1P	1,026	2,076	10,666	987	1,357	797	41.3%	19.3%
FIT2D	10,525	21,024	150,042	20,166	13,298	11,663	12.3%	42.2%
GANGES	1,713	2,110	7,745	294	678	268	60.5%	8.8%
GROW7	420	581	3,172	208	246	139	43.5%	33.2%
GROW15	900	1,245	6,820	977	871	759	12.9%	22.3%
KB2	52	77	331	35	40	37	7.5%	-5.7%
LOTFI	153	366	1,136	177	377	261	30.8%	-47.5%
RECIPELP	207	320	919	33	34	16	52.9%	51.5%
SC50A	50	78	160	20	18	11	38.9%	45.0%
SC50B	50	78	148	19	16	11	31.3%	42.1%
SCAGR7	129	185	465	81	84	52	38.1%	35.8%

Table 4.6: *Improvement in the number of pivots of DPSM over SM and CPLEX's primal simplex algorithm (Netlib) - continued*

Name	Cont	Var	Nz	Number of Pivots			$\delta_{\text{DPSM}}^{\text{SM}}$	$\delta_{\text{DPSM}}^{\text{CPLEX}}$
				CPLEX	SM	DPSM		
SCAGR25	471	671	1,725	357	433	259	40.2%	27.5%
SCFXM1	330	600	2,732	171	227	118	48.0%	31.0%
SCFXM2	660	1,200	5,469	327	503	276	45.1%	15.6%
SCFXM3	990	1,800	8,206	502	794	445	44.0%	11.4%
SCORPION	388	466	1,534	270	271	242	10.7%	10.4%
SEBA	1,023	1,537	5,369	65	70	36	48.6%	44.6%
SHARE1B	117	253	1,179	150	404	198	51.0%	-32.0%
SHARE2B	96	162	777	40	80	39	51.3%	2.5%
SHELL	912	1,903	4,060	325	404	244	39.6%	24.9%
SHIP04L	402	2,166	6,380	346	366	195	46.7%	43.6%
SHIP04S	402	1,506	4,400	155	145	76	47.6%	51.0%
SHIP08L	778	4,363	12,882	668	717	382	46.7%	42.8%
SHIP08S	778	2,467	7,194	454	567	291	48.7%	35.9%
SHIP12L	1,151	5,533	16,276	276	262	153	41.6%	44.6%
SHIP12S	1,151	2,869	8,284	187	187	101	46.0%	46.0%
STANDATA	479	1,383	3,459	138	113	62	45.1%	55.1%
STANDGUB	481	1,492	3,567	113	121	97	19.8%	14.2%
STANDMPS	587	1,383	4,107	418	409	245	40.1%	41.4%
STOCFOR1	117	165	501	91	59	32	45.8%	64.8%
STOCFOR2	2,157	3,045	9,357	1,057	1,315	946	28.1%	10.5%
STOCFOR3	16,675	23,541	72,721	18,931	21,221	18,254	14.0%	3.6%
TUFF	366	660	4,626	85	253	124	51.0%	-45.9%
Average							40.6%	20.3%

Tables 4.5-4.7 combined show that DPSM averages 41% fewer pivots than SM and 22% fewer pivots than CPLEX's primal simplex algorithm. Each double pivot identified an average of only 2 constraints that updated \bar{x} , implying that Theorem 3.2.2 and Corollaries 3.2.3, 3.2.4, and 3.2.5 are infrequently implemented. The study shows that on average 83% of pivots entered two nonbasic variables in the basis, 2% entered the nonbasic variable with the most negative reduced cost, and 15% entered the nonbasic variable with the second most negative reduced cost. Overall, only 0.05% of the iterations had a single negative reduced

cost, which implies that the vast majority of the iterations pivoted with SA and not with the minimum ratio test.

Table 4.7: *Improvement in the number of pivots of DPSM over SM and CPLEX’s primal simplex algorithm (MIPLIB)*

Name	Cont	Var	Nz	Number of Pivots			$\delta_{\frac{\text{DPSM}}{\text{SM}}}$	$\delta_{\frac{\text{DPSM}}{\text{CPLEX}}}$
				CPLEX	SM	DPSM		
50v-10	1880	3843	6222	222	212	116	45.3%	47.7%
dfn-gwin-UUM	158	985	2679	137	126	76	39.7%	44.5%
ger50_17_trans	499	22641	172262	751	623	345	44.6%	54.1%
germanrr	21703	32269	207927	78	77	55	28.6%	29.5%
ic97_potential	1774	2502	5640	47	48	24	50.0%	48.9%
janos-us-DDM	760	2268	6468	342	318	161	49.4%	52.9%
mcsched	3854	5399	13487	3118	4158	3666	11.8%	-17.6%
noswot	282	408	1115	10	33	10	69.7%	0.0%
ns1766074	182	262	828	83	224	137	38.8%	-65.1%
timtab1	568	794	1623	54	53	30	43.4%	44.4%
Average							42.1%	23.9%

The average relative improvement in objective function value per iteration of all benchmark instances solved is $\bar{\Delta} = 170\%$. Define improvement in the objective function value per iteration as:

$$\Delta = \left(\frac{z_{\text{double pivot}} - z_{\text{current}}}{z_{\text{classic}} - z_{\text{current}}} - 1 \right) \times 100\%,$$

where z_{current} is the z value of the current basis, $z_{\text{double pivot}}$ is the z value after a double pivot, and z_{classic} is the z value after a classic pivot. If a classic pivot does not improve the objective function value, then:

$$\Delta = \left(\frac{z_{\text{double pivot}} - z_{\text{current}}}{z_{\text{current}}} \right) \times 100\%.$$

In other words, Δ represents the percentage improvement in objective function value that occurs by using a double pivot compared to the improvement attributed to the use of a classic pivot. Hence, all values of Δ are averaged to create $\bar{\Delta}$.

When analyzing based on pivot type, $\bar{\Delta} = 473\%$ for double pivots that enter two nonbasic variables into the basis, $\bar{\Delta} = 135\%$ for double pivots that enter the nonbasic variable with the second most negative reduced cost, and obviously $\bar{\Delta} = 0\%$ for double pivots that enter the nonbasic variable with the most negative reduced cost.

Even though DPSM outperforms SM and CPLEX's primal simplex algorithm in number of pivots, the question of whether or not DPSM is computationally faster still remains. Tables 4.5, 4.6, and 4.7 do not include solution times because the vast majority of these instances were solved by DPSM, SM and CPLEX's primal simplex algorithm in less than a tenth of a second. Obtaining reliable data on such small time increments is both inconclusive and unconvincing. However, it suffices to say that DPSM, SM, and CPLEX's primal simplex algorithm were very close in computational time for these instances.

To provide a partial answer, some dense and sparse random LPs that did not present numerical instability issues with respect to the *CPXpivot* routine were solved with DPSM and SM. These LPs take the same form of the LPs described in Chapter 4.2.2.1. In total, 50 problems that ranged from 2,000–10,000 variables and 1,000–5,000 constraints were solved. These instances were large enough to produce reasonable solution times for comparison. On average, these problems solved with DPSM in 62 seconds, SM in 54 seconds, and CPLEX's primal simplex algorithm in 39 seconds.

When broken down by steps, SM spent on average 1% of the solution time to find the entering nonbasic variable, 6% to obtain the right-hand side values and constraint matrix values of the entering nonbasic variable, 1% to perform the minimum ratio test, 91% to swap the leaving and entering variables using the *CPXpivot* routine, and 1% for all other operations. Similarly, DPSM spent on average 1% to find both entering nonbasic variables, 11% to create 2VLPs, 1% to find the leaving basic variable(s) with SA, 86% to exchange the leaving with entering variable(s) using the *CPXpivot* routine (called once or twice depending on the type of pivot), and 1% for all other operations. Therefore, the vast majority of time is spent pivoting and creating data for the minimum ratio tests or 2VLPs. Thus, SA is similar in computational speed to the minimum ratio test, which follows the theoretical analysis that both algorithms have the same theoretical running time per iteration.

This small study demonstrates that SA does not significantly impact DPSM’s solution time. The most expensive step corresponds to the *CPXpivot* routine. This routine not only updates the basis’ inverse factorization, but also calculates the solution of all basic variables, reduced cost values, dual price values, etc. Unfortunately, DPSM frequently calls the *CPXpivot* routine twice during an iteration, which forces unnecessary work. Thus, this implementation of DPSM is at a competitive disadvantage compared to both SM and CPLEX’s primal simplex algorithm. Consequently, only a full implementation of DPSM in a quality commercial or open source linear programming solver can truly answer whether or not DPSM is faster than SM. Fully implementing DPSM requires the development of an efficient method to update the basis factorization for double pivots.

4.3 The Double Pivot Dual Simplex Method

As discussed in Chapter 2.1, the dual simplex method is sometimes preferred to the simplex method in state-of-the-art commercial and open source solvers. This section presents the double pivot dual simplex method (DPDSM), a novel dual simplex framework to solve LPs where $|Q'| = 2$. Similar to the dual simplex method, DPDSM requires an optimal basis to its two dimensional subspace linear program solved at each iteration; that is, a 2CLP. Consequently, RA is the method used to solve DPDSM’s subproblems.

To implement RA within a dual simplex framework, one should question whether RA can identify an optimal basis to a 2CLP. Thus, the major theoretical result in this section is to prove that variable indices i^* and j^* define an optimal basis to a 2CLP. To prove this result, consider the standard form of a 2CLP presented in Chapter 3.3 (S2CLP).

The following three corollaries describe conditions that help proving this critical result. Corollary 4.3.1 presents the relationship between the coefficients of two variable indices. Corollaries 4.3.2 and 4.3.3 describe the relationship between the coefficients of two variable indices and the right-hand side values b_1 and b_2 . For brevity, the proof of each of these three corollaries is not presented. However, the proof from Corollary 4.3.1 can be easily derived

from Lemma 3.2.2, and the proof from Corollaries 4.3.2 and 4.3.3 can be obtained from Corollaries 3.2.1 and 3.2.2.

Corollary 4.3.1. *If S2CLP has i and $j \in N$ such that none of the conditions for an infeasible S2CLP are satisfied, $\beta_i < M$, $-M < \beta_j \leq 2M$, and $\beta_i < \beta_j$, then $a_{2,i}a_{1,j} < a_{1,i}a_{2,j}$. \square*

Corollary 4.3.2. *If S2CLP has an $i \in N$ such that $\beta_i < \frac{b_2}{b_1}$, then $a_{2,i}b_1 < a_{1,i}b_2$. \square*

Corollary 4.3.3. *If S2CLP has a $j \in N$ such that $\frac{b_2}{b_1} \leq \beta_j \leq 2M$, then $a_{1,j}b_2 \leq a_{2,j}b_1$. \square*

Theorem 4.3.1 proves that RA defines an optimal basis to any feasible S2CLP. The proof follows by examining all appropriate cases for β_{i^*} and β_{j^*} .

Theorem 4.3.1. *Given a feasible S2CLP and a sufficiently large positive number M , the ratio algorithm reports i^* , j^* , β_{i^*} , and β_{j^*} such that the following sets are an optimal basis for the corresponding S2CLP:*

- i) If $\beta_{i^*} = -2M$, then $BV = \{j^*, n + 2\}$;*
- ii) If $\beta_{j^*} = 2M$, then $BV = \{n + 1, i^*\}$;*
- iii) If $\beta_{i^*} \neq -2M$ and $\beta_{j^*} \neq 2M$, then $BV = \{i^*, j^*\}$.*

Proof. Given a feasible S2CLP and a sufficiently large positive number M , the ratio algorithm reports i^* , j^* , β_{i^*} , and β_{j^*} . To determine that BV is an optimal basis to the corresponding S2CLP, one must show $c_i^\pi < 0$ for each $i \in NBV$, and b_1^π and b_2^π are nonnegative where c^π and b^π are the calculated reduced cost and right-hand side values, respectively.

From Algorithm 3.2, $c_i^\pi < 0$ for every $i \in N \setminus (i^* \cup j^*) = NBV$ because the z calculated with these variable indices is greater than z' . If $z < z'$, then Algorithm 3.2 has $c_i^\pi > 0$ for some $i \in N \setminus (i^* \cup j^*) = NBV$, but either $b_1^\pi < 0$ and/or $b_2^\pi < 0$. Consequently, the reduced cost of each variable index, except i^* and j^* , is either nonimproving or defines an infeasible basis. Since i^* and j^* are both improving, the remainder of the proof shows that b_1^π and b_2^π are nonnegative by evaluating all appropriate cases for β_{i^*} and β_{j^*} .

If $\beta_{i^*} = -2M$, assume $BV = \{j^*, n+2\}$. Since the S2CLP is feasible, then $-M < \beta_{j^*} < M$, which implies $a_{1,j^*} > 0$ and $a_{2,j^*} > 0$. Therefore, $b_1^\pi = \frac{b_1}{a_{1,j^*}} > 0$ and $b_2^\pi = \frac{a_{2,j^*}b_1 - a_{1,j^*}b_2}{a_{1,j^*}}$. From Corollary 4.3.3, $\frac{b_2}{b_1} \leq \beta_{j^*} \leq 2M$ and $a_{1,j^*}b_2 \leq a_{2,j^*}b_1$. Consequently, $b_2^\pi \geq 0$.

If $\beta_{j^*} = 2M$, assume $BV = \{n+1, i^*\}$. Since the S2CLP is feasible, then $-M < \beta_{i^*} < M$, which implies $a_{1,i^*} > 0$ and $a_{2,i^*} > 0$. Therefore, $b_2^\pi = \frac{b_2}{a_{2,i^*}} > 0$ and $b_1^\pi = \frac{a_{1,i^*}b_2 - a_{2,i^*}b_1}{a_{2,i^*}}$. From Corollary 4.3.2, $\beta_{i^*} < \frac{b_2}{b_1}$ and $a_{2,i^*}b_1 < a_{1,i^*}b_2$. Consequently, $b_1^\pi > 0$.

If $\beta_{i^*} \neq -2M$ and $\beta_{j^*} \neq 2M$, assume $BV = \{i^*, j^*\}$. Observe that $b_1^\pi = \frac{a_{1,j^*}b_2 - a_{2,j^*}b_1}{a_{2,i^*}a_{1,j^*} - a_{1,i^*}a_{2,j^*}}$ and $b_2^\pi = \frac{a_{2,i^*}b_1 - a_{1,i^*}b_2}{a_{2,i^*}a_{1,j^*} - a_{1,i^*}a_{2,j^*}}$. Since $\beta_{i^*} < \frac{b_2}{b_1} \leq \beta_{j^*} \leq 2M$, conditions of Corollaries 4.3.1, 4.3.2, and 4.3.3 are satisfied. From Corollaries 4.3.1 and 4.3.3, $a_{2,i^*}a_{1,j^*} < a_{1,i^*}a_{2,j^*}$ and $a_{1,j^*}b_2 \leq a_{2,j^*}b_1$, respectively. Thus, $b_1^\pi \geq 0$. From Corollary 4.3.2, $a_{2,i^*}b_1 < a_{1,i^*}b_2$. Combining this fact with Corollary 4.3.1 results in $b_2^\pi > 0$.

Since $a_{1,j^*} > 0$ when $\beta_{i^*} = -2M$, $a_{2,i^*} > 0$ when $\beta_{j^*} = 2M$, and $a_{2,i^*}a_{1,j^*} - a_{1,i^*}a_{2,j^*} < 0$ when $\beta_{i^*} \neq -2M$ and $\beta_{j^*} \neq 2M$, then the columns of BV are linearly independent in all three cases. Consequently, BV is an optimal basis to the S2CLP and the proof is shown. \square

Algorithm 4.2 presents DPDSM. Similar to DPSM, the double pivot dual simplex method is shown within the context of a revised simplex framework, and the algorithm chooses the two indices with the most negative right-hand side values for the leaving basic variables. The input to DPDSM is an SLP, a super optimal basis BV , and a sufficiently large positive number M .

The double pivot dual simplex method correctly solves a nondegenerate SLP following the same arguments and steps used to prove Theorem 4.2.2. The following corollary formalizes this claim.

Corollary 4.3.4. *Given a nondegenerate SLP, an initial super optimal basis BV , and a sufficiently large positive number M , DPDSM correctly terminates within a finite number of steps.* \square

Since DPDSM begins with a super optimal basis, the algorithm either finds that the corresponding SLP is infeasible or an optimal basis and its corresponding optimal solution. The

double pivot dual simplex method decreases the “infeasibility” at each iteration by at least as much as DSM. Furthermore, one can easily verify that DPDSM’s theoretical running time is $O(rn + I(r))$ when considering the classical revised simplex framework implementation, or $O(rn + U(r))$ when considering a commercial or open source solver implementation.

Algorithm 4.2 : The Double Pivot Dual Simplex Method (DPDSM)

```

1: begin
2:   while  $(A_{.BV}^{-1}b)_j \not\geq 0$  for all  $j \in BV$  do
3:      $p' \leftarrow \operatorname{argmin}_{p' \in BV} (A_{.BV}^{-1}b)_{p'}$ ;
4:      $q' \leftarrow \operatorname{argmin}_{q' \in (BV \cup \{p\})} (A_{.BV}^{-1}b)_{q'}$ ;
5:     if  $(A_{.BV}^{-1}b)_{q'} < 0$  then
6:       Let 2CLP be:
           minimize  $z = (c_{(N \setminus BV)}^T - c_{BV}^T A_{.BV}^{-1} A_{.(N \setminus BV)}) x_{(N \setminus BV)}$ 
           subject to  $-(A_{.BV}^{-1} A_{.(N \setminus BV)})_{p'} x_{(N \setminus BV)} \geq -(A_{.BV}^{-1} b)_{p'}$ 
                      $-(A_{.BV}^{-1} A_{.(N \setminus BV)})_{q'} x_{(N \setminus BV)} \geq -(A_{.BV}^{-1} b)_{q'}$ 
                      $x_{(N \setminus BV)} \geq 0$ ;
7:       Solve 2CLP with RA;
8:       if 2CLP is infeasible then return SLP is infeasible;
9:       if  $\beta_{i^*} = -2M$  then  $BV_{p'} \leftarrow j^*$ ;
10:      if  $\beta_{j^*} = 2M$  then  $BV_{q'} \leftarrow i^*$ ;
11:      if  $\beta_{i^*} \neq -2M$  and  $\beta_{j^*} \neq 2M$  then
12:         $BV_{p'} \leftarrow i^*$ ;
13:         $BV_{q'} \leftarrow j^*$ ;
14:      else
15:         $\theta \leftarrow M$ ;
16:        for each  $k \in N$  do
17:          if  $(A_{.BV}^{-1} A_{.k})_{p'} < 0$  and  $\left| \frac{c_{BV}^T A_{.BV}^{-1} A_{.k} - c_k}{(A_{.BV}^{-1} A_{.k})_{p'}} \right| < \theta$  then
18:             $\theta \leftarrow \left| \frac{c_{BV}^T A_{.BV}^{-1} A_{.k} - c_k}{(A_{.BV}^{-1} A_{.k})_{p'}} \right|$ ;
19:             $l \leftarrow k$ ;
20:          if  $\theta = M$  then return SLP is infeasible;
21:          else
22:             $BV_{p'} \leftarrow l$ ;
23:        return  $x_{BV}^* \leftarrow A_{.BV}^{-1} b$ ,  $x_{(N \setminus BV)}^* = 0$ , and  $z^* = c^T x^*$ ;
24: end

```

To demonstrate the implementation of DPDSM, Example 4.3.1 is presented. Table 4.8 depicts the three tableaus that show DPDSM’s pivots.

Example 4.3.1. Consider the following LP.

$$\text{minimize } z = 99x_1 + 40x_2 + 106x_3 + 60x_4 + 170x_5$$

$$\text{subject to } \quad x_1 + x_2 + 4x_3 + 2x_4 + 2x_5 \geq 20 \quad (1)$$

$$\quad -2x_1 \quad + 9x_3 + 2x_4 - x_5 \geq 12 \quad (2)$$

$$\quad 3x_1 + x_2 + x_3 + x_4 + 5x_5 \geq 15 \quad (3)$$

$$\quad x_1 \quad + 4x_3 + x_4 \geq 6 \quad (4)$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

One can see that the above LP is exactly the corresponding dual of the LP from Example 4.2.1. The first tableau describes the corresponding SLP from Example 4.3.1, and the initial super optimal basis is $BV = \{6, 7, 8, 9\}$. Observe that this tableau is super optimal because the reduced cost for all nonbasic variables are negative and SLP is a minimization problem. In addition, $BV = \{6, 7, 8, 9\}$ is also infeasible because all right-hand side values are negative.

To begin, DPDSM selects the basic variables with the two most negative right-hand side values, which are x_6 and x_8 . Consequently, $p' = 1$ and $q' = 3$. The ratio algorithm solves the following 2CLP:

$$\text{minimize } z = 99x_1 + 40x_2 + 106x_3 + 60x_4 + 170x_5$$

$$\text{subject to } \quad x_1 + x_2 + 4x_3 + 2x_4 + 2x_5 \geq 20$$

$$\quad 3x_1 + x_2 + x_3 + x_4 + 5x_5 \geq 15$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0.$$

The ratio algorithm calculates $\beta = (3, 1, \frac{1}{4}, \frac{1}{2}, \frac{5}{2}, 2M, -2M)$ and finds $i^* = 4$ and $j^* = 2$ because $\beta_4 = \frac{1}{2} < \frac{b_8}{b_6} = \frac{15}{20} \leq \beta_2 = 1$. The algorithm verifies that $\bar{x} = (\bar{x}_4, \bar{x}_2) = (5, 10)$, and $z' = 60\bar{x}_4 + 40\bar{x}_2 = 700$ is the minimal feasible objective function value that one could obtain from S2CLP. Hence, RA returns $i^* = 4$, $j^* = 2$, $\beta_4 = \frac{1}{2}$, and $\beta_2 = 1$. In this case, the first element in BV is replaced with index 4, the third element in BV is replaced with index 2, and the resulting basis is $BV = \{4, 7, 2, 9\}$. The second tableau in Table 4.8 shows this double pivot.

Table 4.8: *Double pivot dual simplex tableau from Example 4.3.1*

BV	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	RHS
		1	-99	-40	-106	-60	-170	0	0	0	0
x_6	0	-1	-1	-4	-2	-2	1	0	0	0	-20
x_7	0	2	0	-9	-2	1	0	1	0	0	-12
x_8	0	-3	-1	-1	-1	-5	0	0	1	0	-15
x_9	0	-1	0	-4	-1	0	0	0	0	1	-6
	1	-19	0	-6	0	-30	-20	0	-20	0	700
x_4	0	-2	0	3	1	-3	-1	0	1	0	5
x_7	0	-2	0	-3	0	-5	-2	1	2	0	-2
x_2	0	5	1	-2	0	8	1	0	-2	0	10
x_9	0	-3	0	-1	0	-3	-1	0	1	1	-1
	1	-1	0	0	0	-12	-14	0	-26	-6	706
x_4	0	-11	0	0	1	-12	-4	0	4	3	2
x_7	0	7	0	0	0	4	1	1	-1	-3	1
x_2	0	11	1	0	0	14	3	0	-4	-2	12
x_3	0	3	0	1	0	3	1	0	-1	-1	1

Because $x_7 = -2$ and $x_9 = -1$, DPDSM continues with $p' = 2$ and $p' = 4$. The ratio algorithm solves the following 2CLP:

$$\begin{aligned}
 &\text{minimize } z = 19x_1 + 6x_3 + 30x_5 + 20x_6 + 20x_8 \\
 &\text{subject to } \quad 2x_1 + 3x_3 + 5x_5 + 2x_6 - 2x_8 \geq 2 \\
 &\quad \quad \quad 3x_1 + x_3 + 3x_5 + x_6 - x_8 \geq 1 \\
 &\quad \quad \quad x_1, x_3, x_5, x_6, x_8 \geq 0.
 \end{aligned}$$

The ratio algorithm determines $\beta = (\frac{3}{2}, \frac{1}{3}, \frac{3}{5}, \frac{1}{2}, 3M, 2M, -2M)$ and finds $\beta_3 = \frac{1}{3} < \frac{b_9}{b_7} = \frac{1}{2} \leq \alpha_6 = \frac{1}{2}$ such that $\bar{x} = (\bar{x}_3, \bar{x}_6) = (0, 1)$ with $z' = 6\bar{x}_3 + 20\bar{x}_6 = 20$. The algorithm runs its optimality procedure and eventually determines $\bar{x} = (\bar{x}_3, \bar{x}_{10}) = (1, 1)$ with $z' = 6\bar{x}_3 + 0\bar{x}_{10} = 6$ as the minimal feasible objective function value from S2CLP. Observe that x_{10} corresponds to one of the two surplus variables added to 2CLP in order to convert it to an S2CLP. Even though this variable index is out of the range of SLP, this does not cause issues to DPDSM

because index 10 never enters into the basis. This comment also follows for the other surplus variable, x_{11} . Finally, RA returns $i^* = 3$, $j^* = 10$, $\beta_3 = \frac{1}{3}$, and $\beta_{10} = 2M$.

Since $\beta_{j^*} = 2M$, the fourth element in BV is replaced with index 3 and the resulting basis is $BV = \{4, 7, 2, 3\}$. The third tableau in Table 4.8 shows this double pivot. Since $c_i^\pi < 0$ for all $i \in NBV$ (minimization problem), $x_4 \not\leftarrow 0$, $x_7 \not\leftarrow 0$, $x_2 \not\leftarrow 0$, and $x_3 \not\leftarrow 0$, thus $BV = \{4, 7, 2, 3\}$ is feasible and optimal. Therefore, DPDSM reports an optimal solution $z^* = 706$ and $x^* = (0, 12, 1, 2, 0, 0, 1, 0, 0)$.

Similar to DPSM in Example 4.2.1, DPDSM performs 50% fewer iterations than DSM in this example. This is because DSM begins with $BV = \{6, 7, 8, 9\}$ and successively moves to the following bases until an optimal basis is obtained: $BV = \{3, 7, 8, 9\}$, $BV = \{3, 7, 4, 9\}$, $BV = \{3, 2, 4, 9\}$, $BV = \{3, 2, 4, 7\}$. Additionally, the relative improvement in objective function value is 32.1% for the first double pivot and 50% for the second double pivot. These improvements are also identical to Example 4.2.1 because of the primal-dual relationship between both LPs.

This chapter demonstrated that two dimensional searches within a simplex framework substantially decrease the number of pivots and have the potential to more quickly solve LPs. Even though both SM and DSM are vastly used by commercial and open source solvers, interior point methods became popular over the last decades, and are occasionally faster than SM or DSM depending on the type of LPs solved (recall this discussion in Chapter 2). Hence, an obvious research direction is to implement two dimensional searches within an interior point framework. The next chapter describes the algorithmic, theoretical, and computational research created on this topic.

Chapter 5

Two Dimensional Searches in an Interior Point Framework

This chapter presents a new set of two dimensional search interior point algorithms. The primary goal is to demonstrate and validate through novel methods that these two dimensional search interior point algorithms solve LPs faster than the corresponding one dimensional search version. Chapter 5.1 discusses the classical one dimensional search interior point algorithms studied during this dissertation’s research. Chapter 5.2 presents novel primal and dual two dimensional search interior point algorithms derived from affine and logarithmic barrier search directions. Chapter 5.3 describes computational experiments that show the effectiveness of the newly developed algorithms.

The primary content of this chapter is based on the manuscript, “Projected Orthogonal Vectors in Two Dimensional Search Interior Point Algorithms for Linear Programming”, currently under peer-review (Vitor and Easton, 2019a). Some preliminary results to this manuscript can also be found in the conference paper, “A Two Dimensional Search Primal Affine Scaling Interior Point Algorithm for Linear Programs”, published in the *Proceedings of the 2018 IISE Annual Conference* (Vitor and Easton, 2018a).

5.1 One Dimensional Search Interior Point Algorithms

This section presents the main theoretical and algorithmic details of the classical one dimensional search primal affine scaling (Chapter 5.1.1), dual affine scaling (Chapter 5.1.2), primal logarithmic barrier (Chapter 5.1.3), and dual logarithmic barrier (Chapter 5.1.4) interior point algorithms. These methods are well documented in the literature and this section only presents a summary of each algorithm. Fundamental knowledge about each of these techniques is vital to understanding the newly developed two dimensional search algorithms in Chapter 5.2. Additional details about these methods can be found in Gondzio (2012), Gonzaga (1992), Lustig et al. (1994a), Fang and Puthenpura (1993), Bazarraa et al. (2010), and Vanderbei (2014).

5.1.1 One Dimensional Search Primal Affine Scaling Interior Point Algorithm

Formally, denote a primal linear program (PLP_x) as:

$$\begin{aligned} & \text{minimize} && z = c^T x \\ & \text{subject to} && Ax = b \\ & && x \geq 0, \end{aligned}$$

where $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{r \times n}$ is full row rank, and $b \in \mathbb{R}^r$. Denote $N = \{1, \dots, n\}$ as the set of variable indices and $R = \{1, \dots, r\}$ as the set of constraint indices of a PLP_x . The feasible region of a PLP_x is defined as $S = \{x \in \mathbb{R}_+^n : Ax = b\}$ and the relative interior of S is denoted by $S' = \{x \in \mathbb{R}^n : Ax = b, x > 0\}$. The optimal solution of a PLP_x is (z^*, x^*) where $x^* \in S$ and $z^* = c^T x^* \leq c^T x$ for all $x \in S$. Observe that the one dimensional search primal affine scaling interior point algorithm ($\text{1DIM}_{\text{aff}}^{\text{P}}$) converges to (z^*, x^*) by successively moving from an $x' \in S'$ to an $x'' \in S'$ with $c^T x'' < c^T x'$. Furthermore, PLP_x corresponds exactly to an SLP, but PLP_x is a minimization LP and A is simply defined as full row rank in this case.

From a feasible interior solution x_{1D}^k , each iteration k of $1\text{DIM}_{\text{aff}}^{\text{P}}$ has the following three primary steps: (1) determine a feasible and improving search direction dx_{1D}^k ; (2) find an appropriate step size $\lambda_{1D}^k > 0$ by solving a one dimensional subspace linear program (1DSLPL); (3) move to a new and improved feasible interior solution x_{1D}^{k+1} . In such a case, $1\text{DIM}_{\text{aff}}^{\text{P}}$ follows the spirit of [Karmarkar's](#) projective scaling algorithm ([Karmarkar, 1984](#)). Recall that [Karmarkar's](#) algorithm utilizes a projective transformation to place the feasible interior solution near the center of the LP's feasible region in the transformed space. The algorithm moves to an improved feasible interior solution in the transformed space by following the steepest descent, given by the objective function. The inverse of the projective transformation maps the improved feasible interior solution back to the original space and the algorithm repeats this process until a termination criterion is satisfied.

The core concept of $1\text{DIM}_{\text{aff}}^{\text{P}}$ is the replacement of the projective transformation from [Karmarkar's](#) algorithm with the affine scaling transformation. Given a feasible interior solution $x_{1D}^k > 0$ at the k^{th} iteration, the affine scaling transformation $y = X_k^{-1}x$ maps x_{1D}^k from the x -space to the y -space where X_k is an $n \times n$ nonsingular diagonal matrix with the elements of x_{1D}^k in the diagonal. Observe that mapping x_{1D}^k from the x -space to the y -space using the aforementioned transformation preserves all linear properties of x . Consequently, applying the affine scaling transformation to the corresponding PLP_x results in the following primal linear program in the y -space (PLP_y):

$$\begin{aligned} &\text{minimize } z = c^T X_k y \\ &\text{subject to } \quad AX_k y = b \\ &\quad \quad \quad y \geq 0. \end{aligned}$$

A search direction dy_{1D}^k to PLP_y is feasible if $AX_k y_{1D}^k = b$ and $AX_k(y_{1D}^k + \lambda_{1D}^k dy_{1D}^k) = b$, which implies $AX_k dy_{1D}^k = 0$. Moreover, dy_{1D}^k is an improving search direction to PLP_y if $c^T X_k(y_{1D}^k + \lambda_{1D}^k dy_{1D}^k) < c^T X_k y_{1D}^k$ for every $\lambda_{1D}^k > 0$, resulting in $c^T X_k dy_{1D}^k < 0$. From the aforementioned discussion about [Karmarkar's](#) algorithm, projecting $-(c^T X_k)^T = -X_k c$ onto the null space of AX_k is sufficient to have $AX_k dy_{1D}^k = 0$ and $c^T X_k dy_{1D}^k < 0$. Let

$I - X_k A^T (A X_k^2 A^T)^{-1} A X_k$ be the orthogonal projection matrix and the search direction $dy_{1D}^k = -(I - X_k A^T (A X_k^2 A^T)^{-1} A X_k) X_k c$ where I is an $n \times n$ identity matrix. Consequently, $dy_{1D}^k = -X_k (c - A^T (A X_k^2 A^T)^{-1} A X_k^2 c)$.

Since dy_{1D}^k is a feasible and improving search direction, the question is how far to move from y_{1D}^k along dy_{1D}^k to an improved feasible interior solution y_{1D}^{k+1} . Thus, one must determine $\lambda_{1D}^k > 0$ such that $y_{1D}^{k+1} = y_{1D}^k + \lambda_{1D}^k dy_{1D}^k \geq 0$. If $dy_{1D}^k > 0$, then $y_{1D}^k + \lambda_{1D}^k dy_{1D}^k > 0$ for all $\lambda_{1D}^k > 0$ and the problem is unbounded. If $dy_{1D}^k = 0$, the current solution is primal optimal. If not, then there exists some $i \in N$ such that $(dy_{1D}^k)_i \leq 0$, which implies $\lambda_{1D}^k \leq -\frac{(y_{1D}^k)_i}{(dy_{1D}^k)_i}$ or simply $\lambda_{1D}^k < -\frac{1}{(dy_{1D}^k)_i}$ because $dy_{1D}^k = -X_k (c - A^T (A X_k^2 A^T)^{-1} A X_k^2 c)$. Thus, optimally determining λ_{1D}^k , denoted by λ_{1D}^{k*} , corresponds to solving

$$\lambda_{1D}^{k*} = \min_i \left\{ -\frac{1}{(dy_{1D}^k)_i} : (dy_{1D}^k)_i < 0 \quad \forall i \in N \right\},$$

which is equivalent to solving the following 1DSLP:

$$\begin{aligned} & \text{minimize} && z = c^T X_k dy_{1D}^k \lambda_{1D}^k \\ & \text{subject to} && dy_{1D}^k \lambda_{1D}^k \geq -1 \\ & && \lambda_{1D}^k \geq 0. \end{aligned}$$

Finally, a new and improved feasible interior solution in the y -space, y_{1D}^{k+1} , is $y_{1D}^k + \alpha \lambda_{1D}^{k*} dy_{1D}^k$ where $\alpha \in (0, 1)$. Observe that by forcing $\alpha < 1$, the boundary of PLP_y is never reached and y_{1D}^{k+1} remains in the interior. The final step is to map the current y_{1D}^{k+1} back to the x -space by using the inverse of the affine scaling transformation, $x = X_k y$. Consequently, $x_{1D}^{k+1} = x_{1D}^k + \alpha \lambda_{1D}^{k*} X_k dy_{1D}^k$.

In summary, $\text{1DIM}_{\text{aff}}^P$ begins each iteration k with $x_{1D}^k \in P'$ and determines $x_{1D}^{k+1} = x_{1D}^k + \alpha \lambda_{1D}^{k*} dx_{1D}^k \in P'$ for some $\alpha \in (0, 1)$ until $\text{1DIM}_{\text{aff}}^P$ is sufficiently close to (z^*, x^*) , where $dx_{1D}^k = -X_k^2 (c - A^T (A X_k^2 A^T)^{-1} A X_k^2 c)$ and

$$\lambda_{1D}^{k*} = \min_i \left\{ -\frac{(x_{1D}^k)_i}{(dx_{1D}^k)_i} : (dx_{1D}^k)_i < 0 \quad \forall i \in N \right\}.$$

5.1.2 One Dimensional Search Dual Affine Scaling Interior Point Algorithm

The corresponding dual linear program (DLP_{w,s}) of a PLP_x can be defined as:

$$\begin{aligned}
 & \text{maximize } z = b^T w \\
 & \text{subject to } A^T w + s = c \\
 & \quad w \text{ is unrestricted} \\
 & \quad s \geq 0,
 \end{aligned}$$

where $w \in \mathbb{R}^r$ and $s \in \mathbb{R}^n$. The feasible region of a DLP_{w,s} is denoted as $T = \{w \in \mathbb{R}^r, s \in \mathbb{R}_+^n : A^T w + s = c\}$ and the relative interior of T is $T' = \{w \in \mathbb{R}^r, s \in \mathbb{R}^n : A^T w + s = c, s > 0\}$. The optimal solution of a DLP_{w,s} is (z^*, w^*, s^*) where $(w^*, s^*) \in T$ and $z^* = b^T w^* \geq b^T w$ for all $(w, s) \in T$. Since DLP_{w,s} is a maximization problem, the one dimensional search dual affine scaling interior point algorithm (1DIM_{aff}^D) converges to (z^*, w^*, s^*) by moving from a $(w', s') \in T'$ to a $(w'', s'') \in T'$ with $b^T w'' > b^T w'$ at each iteration.

Even though 1DIM_{aff}^D follows the same principle of 1DIM_{aff}^P and [Karmarkar's](#) interior point algorithm, s is the variable placed near the center of the dual LP's feasible region in the transformed space, because w is unrestricted. Therefore, 1DIM_{aff}^D begins each iteration k with a feasible interior dual solution (w_{1D}^k, s_{1D}^k) and determines a feasible improving search direction (dw_{1D}^k, ds_{1D}^k) , selects an appropriate step size $\rho_{1D}^k > 0$ by solving a 1DSL_P, and calculates a new and improved feasible interior dual solution $(w_{1D}^{k+1}, s_{1D}^{k+1})$.

To determine a search direction (dw_{1D}^k, ds_{1D}^k) that is feasible ($A^T dw_{1D}^k + ds_{1D}^k = 0$) and improving ($b^T dw_{1D}^k > 0$), the affine scaling transformation $v = S_k^{-1}s$ is applied to DLP_{w,s} and creates a DLP_{w,v}. From the affine scaling transformation, $ds_{1D}^k = S_k dv_{1D}^k$ and the feasibility condition becomes $A^T dw_{1D}^k + S_k dv_{1D}^k = 0$. Multiplying $A^T dw_{1D}^k + S_k dv_{1D}^k = 0$ with AS_k^{-2} results in $dw_{1D}^k = -(AS_k^{-2}A^T)^{-1}AS_k^{-1}dv_{1D}^k$. Since $b^T dw_{1D}^k > 0$, $dv_{1D}^k = -((AS_k^{-2}A^T)^{-1}AS_k^{-1})^T b$ and $dw_{1D}^k = (AS_k^{-2}A^T)^{-1}b$. Consequently, $ds_{1D}^k = S_k dv_{1D}^k = -A^T(AS_k^{-2}A^T)^{-1}b = -A^T dw_{1D}^k$.

Observe that $1\text{DIM}_{\text{aff}}^{\text{D}}$ optimally determines $\rho_{1D}^k > 0$, denoted by ρ_{1D}^{k*} , following a similar logic as how $1\text{DIM}_{\text{aff}}^{\text{P}}$ calculates λ_{1D}^{k*} . That is,

$$\rho_{1D}^{k*} = \min_i \left\{ -\frac{(s_{1D}^k)_i}{(ds_{1D}^k)_i} : (ds_{1D}^k)_i < 0 \quad \forall i \in N \right\}.$$

Since ds_{1D}^k has been already derived into the s -space, the new and improved feasible interior dual solution is computed as $w_{1D}^{k+1} = w_{1D}^k + \alpha \rho_{1D}^{k*} dw_{1D}^k$ and $s_{1D}^{k+1} = s_{1D}^k + \alpha \rho_{1D}^{k*} ds_{1D}^k$ for some $\alpha \in (0, 1)$.

5.1.3 One Dimensional Search Primal Logarithmic Barrier Interior Point Algorithm

Interior point methods, as many other iterative techniques, are sensitive to numerical errors. This numerical instability may cause these algorithms to perform poorly near the boundary of the LP's feasible region. That is, if a feasible interior solution is sufficiently close to the boundary of the LP's feasible region, the algorithm may behave incorrectly and lead to early termination, an infeasible solution, or even an incorrect solution. Recall from Chapter 2.2.2 that one possible method to resolve this issue is to introduce a barrier term to the objective function of the problem. Commonly, this term is a logarithmic barrier function. This section describes the one dimensional search primal logarithmic barrier interior point algorithm ($1\text{DIM}_{\text{log}}^{\text{P}}$), a technique that combines primal affine scaling and centering search directions.

From a PLP_x , $1\text{DIM}_{\text{log}}^{\text{P}}$ considers the following primal nonlinear program ($\text{PNLP}_{x,\mu}$):

$$\begin{aligned} \text{minimize } z &= f(x, \mu) = c^T x - \mu \sum_{i \in N} \ln x_i \\ \text{subject to } Ax &= b \\ x &> 0, \end{aligned}$$

where $\mu \in \mathbb{R}_+$ with $\lim_{k \rightarrow \infty} \mu^k = 0$. Observe that in $\text{PNLP}_{x,\mu}$, $x = 0$ is no longer allowed and the constraint $x > 0$ is incorporated by the portion $-\mu \sum_{i \in N} \ln x_i$ in the objective function.

This penalizes $f(x, \mu)$ when x approaches 0, which avoids x reaching the boundary of the LP's feasible region. Consequently, x^* is an optimal solution to PLP_x , if and only if, x^* is also an optimal solution to $\text{PNLP}_{x,\mu}$ when μ^k approaches zero as k approaches infinity.

To find a feasible and improving search direction dx_{1D}^k for $\mu_{1D}^k > 0$ at each iteration k , a quadratic approximation of f at a feasible interior solution x_{1D}^k is derived. Let

$$f(x, \mu) \approx f(x_{1D}^k, \mu_{1D}^k) + \nabla f(x_{1D}^k, \mu_{1D}^k)^T dx_{1D}^k + \frac{1}{2}(dx_{1D}^k)^T \nabla^2 f(x_{1D}^k, \mu_{1D}^k) dx_{1D}^k$$

be the quadratic approximation of f and the following primal quadratic program ($\text{PQP}_{dx_{1D}^k}$) determines the search direction dx_{1D}^k :

$$\begin{aligned} \text{minimize } z &= \nabla f(x_{1D}^k, \mu_{1D}^k)^T dx_{1D}^k + \frac{1}{2}(dx_{1D}^k)^T \nabla^2 f(x_{1D}^k, \mu_{1D}^k) dx_{1D}^k \\ \text{subject to } & \quad A dx_{1D}^k = 0. \end{aligned}$$

Observe that the resulting dx_{1D}^k is an improving search direction because $\text{PQP}_{dx_{1D}^k}$ minimizes the quadratic approximation of f , and dx_{1D}^k is a feasible search direction because $Ax_{1D}^k = b$ and $A(x_{1D}^k + \lambda_{1D}^k dx_{1D}^k) = b$ since $A dx_{1D}^k = 0$. One can use the method of Lagrange multipliers to solve $\text{PQP}_{dx_{1D}^k}$. Let $L(x_{1D}^k, \mu_{1D}^k, \gamma_{1D}^k) = f(x_{1D}^k, \mu_{1D}^k) - \gamma_{1D}^k A dx_{1D}^k$ be the Lagrangian for the corresponding $\text{PQP}_{dx_{1D}^k}$ where $\gamma_{1D}^k \in \mathbb{R}^n$ is the Lagrange multiplier. Since $\nabla f(x_{1D}^k, \mu_{1D}^k) = c - \mu_{1D}^k X_k^{-1} e$ and $\nabla^2 f(x_{1D}^k, \mu_{1D}^k) = \mu_{1D}^k X_k^{-2}$ where $e = (1, 1, \dots, 1)^T \in \mathbb{R}^n$, then:

$$\begin{aligned} \frac{\partial L(x_{1D}^k, \mu_{1D}^k, \gamma_{1D}^k)}{\partial dx_{1D}^k} &= \nabla f(x_{1D}^k, \mu_{1D}^k) + \nabla^2 f(x_{1D}^k, \mu_{1D}^k) dx_{1D}^k - A^T \gamma_{1D}^k \\ &= c - \mu_{1D}^k X_k^{-1} e + \mu_{1D}^k X_k^{-2} dx_{1D}^k - A^T \gamma_{1D}^k = 0. \end{aligned}$$

Coupling this fact with $A dx_{1D}^k = 0$ results in the following system of equations:

$$\begin{bmatrix} \mu_{1D}^k X_k^{-2} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} dx_{1D}^k \\ \gamma_{1D}^k \end{bmatrix} = \begin{bmatrix} c - \mu_{1D}^k X_k^{-1} e \\ 0 \end{bmatrix}$$

Solving the above system of equations results in:

$$\begin{aligned} dx_{1D}^k &= -\frac{1}{\mu_{1D}^k} X_k (I - X_k A^T (A X_k^2 A^T)^{-1} A X_k) (X_k c - \mu_{1D}^k e) \\ &= -\frac{1}{\mu_{1D}^k} X_k^2 (c - A^T (A X_k^2 A^T)^{-1} A X_k^2 c) + X_k (e - X_k A^T (A X_k^2 A^T)^{-1} A X_k e). \end{aligned}$$

Therefore, $dx_{1D}^k = \frac{1}{\mu_{1D}^k} \overline{dx}_{1D}^k + X_k (e - X_k A^T (A X_k^2 A^T)^{-1} A X_k e)$ where \overline{dx}_{1D}^k is the search direction determined by $1\text{DIM}_{\text{aff}}^{\text{P}}$. Consequently, the above dx_{1D}^k represents the combination of the primal affine scaling search direction with a centering search direction that pushes x_{1D}^{k+1} away from the boundary of the LP's feasible region. Since an optimal solution to $\text{PNLP}_{x,\mu}$ is achieved when μ_{1D}^k approaches zero as k approaches infinity, μ_{1D}^{k+1} is reduced by a constant $\beta \in (0, 1)$ and $\mu_{1D}^{k+1} = \beta \mu_{1D}^k$. While computing a search direction for $1\text{DIM}_{\text{log}}^{\text{P}}$ differs from $1\text{DIM}_{\text{aff}}^{\text{P}}$, determining λ_{1D}^{k*} and x_{1D}^{k+1} is similar to $1\text{DIM}_{\text{aff}}^{\text{P}}$.

5.1.4 One Dimensional Search Dual Logarithmic Barrier Interior Point Algorithm

The same logic used to derive $1\text{DIM}_{\text{log}}^{\text{P}}$ can be applied to solve a $\text{DLP}_{w,s}$. Thus, the one dimensional search dual logarithmic barrier interior point algorithm ($1\text{DIM}_{\text{log}}^{\text{D}}$) considers the dual nonlinear program ($\text{DNLP}_{w,\mu}$):

$$\begin{aligned} &\text{maximize } z = f(w, \mu) = b^T w + \mu \sum_{i \in N} \ln(c_i - A_i^T w) \\ &\text{subject to } \quad A^T w < c, \end{aligned}$$

where $\mu \in \mathbb{R}_+$. Observe that $s = c - A^T w$ and the term $\sum_{i \in N} \ln(c_i - A_i^T w)$ in $f(w, \mu)$ denotes the sum of each slack variable s_i from $\text{DLP}_{w,s}$, which penalizes s when it is sufficiently close to the boundary of the dual LP's feasible region. Similarly to $1\text{DIM}_{\text{log}}^{\text{P}}$, w^* is an optimal solution to $\text{DLP}_{w,s}$, if and only if, w^* is an optimal solution to $\text{DNLP}_{w,\mu}$ when μ^k approaches zero as k approaches infinity.

Let the Lagrangian for the corresponding DNLP $_{w,\mu}$ at w_{1D}^k be

$$L(w_{1D}^k, \mu_{1D}^k, \gamma_{1D}^k) = f(w_{1D}^k, \mu_{1D}^k) + \gamma_{1D}^k{}^T (c - A^T w_{1D}^k),$$

where $\gamma_{1D}^k \in \mathbb{R}^n$ is the Lagrange multiplier. The reader can trivially verify that $b - \mu_{1D}^k A S_k^{-1} e = 0$ and $s_{1D}^k > 0$ are the first order KKT conditions to $L(w_{1D}^k, \mu_{1D}^k, \gamma_{1D}^k)$. Because $\gamma_{1D}^k{}^T (c - A^T w_{1D}^k) = 0$ and $c - A^T w_{1D}^k > 0$, then $\gamma_{1D}^k{}^T = 0$. Thus, one step of Newton's method from an interior feasible solution (w_{1D}^k, s_{1D}^k) with $s_{1D}^k > 0$ at the k^{th} iteration results in $dw_{1D}^k = \frac{1}{\mu_{1D}^k} (A S_k^{-2} A^T)^{-1} b - (A S_k^{-2} A^T)^{-1} A S_k^{-1} e$ and $ds_{1D}^k = -A^T dw_{1D}^k$. Equivalently, $dw_{1D}^k = \frac{1}{\mu_{1D}^k} \hat{d}w_{1D}^k - (A S_k^{-2} A^T)^{-1} A S_k^{-1} e$ where $\hat{d}w_{1D}^k$ is the search direction determined by 1DIM $_{\text{aff}}^D$. Consequently, dw_{1D}^k is composed by the dual affine scaling search direction along with a centering search direction. Moreover, ρ_{1D}^{k*} and w_{1D}^{k+1} are calculated similarly to 1DIM $_{\text{aff}}^D$.

5.2 Two Dimensional Search Interior Point Algorithms

This section develops primal and dual two dimensional search interior point algorithms derived from affine and logarithmic barrier search directions. Prior to describing these newly created techniques, this chapter presents some general properties to determine effective search directions for two dimensional search methods.

5.2.1 Principles for Determining Two Search Directions

Each of the four techniques presented in Chapter 5.1, and every other one dimensional search interior point algorithm, have different procedures to determine search directions. Selecting a search direction is a vital step for each of these methods, because it impacts the algorithm's performance. Similarly, choosing two search directions appropriately is critical to creating effective two dimensional search interior point algorithms. This section provides two principles for selecting search directions.

At each iteration, two dimensional search interior point algorithms require more computational effort than the corresponding one dimensional search version. Consequently, the

improvement in objective function value per iteration should be significant in order to counteract this additional effort. The first principle involves selecting two search directions that lead to a feasible interior solution with an objective function value that is at least as good as the objective function value obtained with a single search direction.

One key observation is that a two dimensional subspace linear program (2DSLP) is guaranteed to improve the objective function value by at least as much as a 1DSLP, if the search direction from 1DSLP is contained in the nonnegative linear combination of the two search directions from 2DSLP. Theorem 5.2.1 formalizes this claim for two dimensional search interior point algorithms designed to solve any PLP_x .

Theorem 5.2.1. *Given a PLP_x , a feasible interior solution $x \in S'$, and search directions d_{1D} , d'_{2D} , and d''_{2D} where $d_{1D} = \theta'_{2D}d'_{2D} + \theta''_{2D}d''_{2D}$ for some $\theta'_{2D}, \theta''_{2D} \in \mathbb{R}_+$, then $z_{1D} \geq z_{2D}$ where $z_{1D} = \min\{c^T(x + \lambda_{1D}d_{1D}) : x + \lambda_{1D}d_{1D} \in S \text{ and } \lambda_{1D} \geq 0\}$, and $z_{2D} = \min\{c^T(x + \lambda'_{2D}d'_{2D} + \lambda''_{2D}d''_{2D}) : x + \lambda'_{2D}d'_{2D} + \lambda''_{2D}d''_{2D} \in S, \lambda'_{2D} \geq 0, \text{ and } \lambda''_{2D} \geq 0\}$.*

Proof. Given a PLP_x , assume $x \in S'$ is a feasible interior solution. Let d_{1D} , d'_{2D} , and d''_{2D} be search directions where $d_{1D} = \theta'_{2D}d'_{2D} + \theta''_{2D}d''_{2D}$ for some $\theta'_{2D}, \theta''_{2D} \in \mathbb{R}_+$. Denote $z_{1D} = \min\{c^T(x + \lambda_{1D}d_{1D}) : x + \lambda_{1D}d_{1D} \in S \text{ and } \lambda_{1D} \geq 0\}$ where λ_{1D}^* represents the optimal solution to 1DSLP. Define $\lambda'_{2D} = \theta'_{2D}\lambda_{1D}^*$ and $\lambda''_{2D} = \theta''_{2D}\lambda_{1D}^*$. Clearly, λ'_{2D} and $\lambda''_{2D} \geq 0$. Furthermore, $x + \lambda_{1D}^*d_{1D} \in S$ and

$$\begin{aligned} x + \lambda_{1D}^*d_{1D} &= x + \lambda_{1D}^*(\theta'_{2D}d'_{2D} + \theta''_{2D}d''_{2D}) \\ &= x + \lambda_{1D}^*\theta'_{2D}d'_{2D} + \lambda_{1D}^*\theta''_{2D}d''_{2D} \\ &= x + \lambda'_{2D}d'_{2D} + \lambda''_{2D}d''_{2D}. \end{aligned}$$

Therefore, $x + \lambda'_{2D}d'_{2D} + \lambda''_{2D}d''_{2D} \in S$, and so $x + \lambda'_{2D}d'_{2D} + \lambda''_{2D}d''_{2D}$ is a feasible solution to $z_{2D} = \min\{c^T(x + \lambda'_{2D}d'_{2D} + \lambda''_{2D}d''_{2D}) : x + \lambda'_{2D}d'_{2D} + \lambda''_{2D}d''_{2D} \in S, \lambda'_{2D} \geq 0, \text{ and } \lambda''_{2D} \geq 0\}$. Consequently, $z_{1D} = c^T(x + \lambda_{1D}^*d_{1D}) \geq z_{2D}$ and the result follows. \square

The following corollary extends Theorem 5.2.1 for two dimensional search interior point algorithms designed to solve any $DLP_{w,s}$. The reader can trivially derive the proof of Corollary 5.2.1 from the previous theorem.

Corollary 5.2.1. *Given a $DLP_{w,s}$, a feasible interior solution $(w, s) \in T'$, and search directions (dw_{1D}, ds_{1D}) , (dw'_{2D}, ds'_{2D}) , and (dw''_{2D}, ds''_{2D}) where $dw_{1D} = \theta'_{2D}dw'_{2D} + \theta''_{2D}dw''_{2D}$ and $ds_{1D} = \theta'_{2D}ds'_{2D} + \theta''_{2D}ds''_{2D}$ for some $\theta'_{2D}, \theta''_{2D} \in \mathbb{R}_+$, then $z_{1D} \leq z_{2D}$ where $z_{1D} = \max\{b^T(w + \rho_{1D}dw_{1D}) : (w + \rho_{1D}dw_{1D}, s + \rho_{1D}ds_{1D}) \in T \text{ and } \rho_{1D} \geq 0\}$, and $z_{2D} = \max\{b^T(w + \rho'_{2D}dw'_{2D} + \rho''_{2D}dw''_{2D}) : (w + \rho'_{2D}dw'_{2D} + \rho''_{2D}dw''_{2D}, s + \rho'_{2D}ds'_{2D} + \rho''_{2D}ds''_{2D}) \in T, \rho'_{2D} \geq 0, \text{ and } \rho''_{2D} \geq 0\}$. \square*

The previous theoretical results ensure that the movement obtained with two search directions improves the objective function value by at least as much as the movement obtained with a single search direction. Obviously, there may be an infinite number of search directions that satisfy the first principle. Thus, the next question is how to determine a suitable pair of search directions.

Observe that d'_{2D} and d''_{2D} have a direct impact to the area of the two dimensional subspace, and consequently, to the solution of 2DSLPL. One would expect that the larger the area of the two dimensional subspace, the better the objective function value obtained by the new and improved feasible interior solution. Consequently, one anticipates fewer iterations if the area of the two dimensional subspace formed by the two search directions and the LP's feasible region is sufficiently large.

A second principle focuses on selecting two search directions that increase the feasible region of 2DSLPLs. Example 5.2.1 describes a method to satisfy the first and second principles. The LP from the following example is not shown in standard form to better exemplify the aforementioned principle. However, the reader can easily convert this LP into a PLP $_x$ by augmenting an identity matrix to the constraint matrix A .

Example 5.2.1. Consider the following LP.

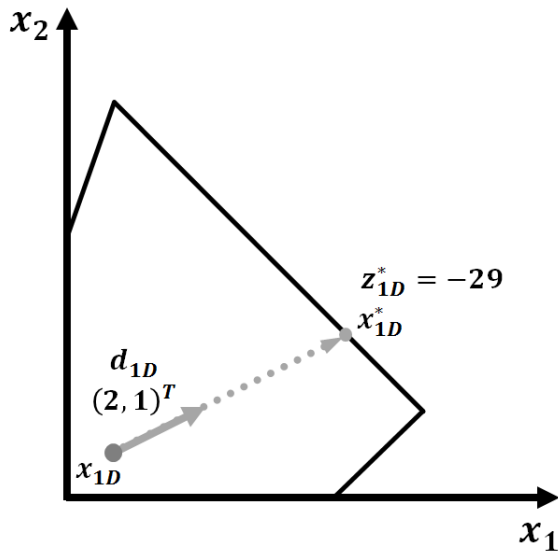
$$\begin{aligned}
& \text{minimize } z = -4x_1 - x_2 \\
& \text{subject to } \quad -3x_1 + x_2 \leq 6 \\
& \quad \quad \quad x_1 + x_2 \leq 10 \\
& \quad \quad \quad x_1 - x_2 \leq 6 \\
& \quad \quad \quad x_1, x_2 \geq 0
\end{aligned}$$

The feasible region of the above LP is shown in Figure 5.1. To solve this problem using either a one dimensional or two dimensional search interior point algorithm, let the initial feasible interior solution be $x_{1D} = x_{2D} = (1, 1)^T$. Figure 5.1(a) shows the movement to a new and improved feasible solution when the single search direction $d_{1D} = (2, 1)^T$ is considered. Solving a 1DSLP, defined by the line from x_{1D} to x_{1D}^* , determines the optimal step size, $\lambda_{1D}^* = \frac{8}{3}$, which leads to the new and improved feasible solution $x_{1D}^* = x_{1D} + \lambda_{1D}^* d_{1D} = (\frac{19}{3}, \frac{11}{3})$. The optimal objective function value of this movement is $z_{1D}^* = -29$. Observe that one can turn x_{1D}^* into an interior solution by multiplying λ_{1D}^* by some $\alpha \in (0, 1)$.

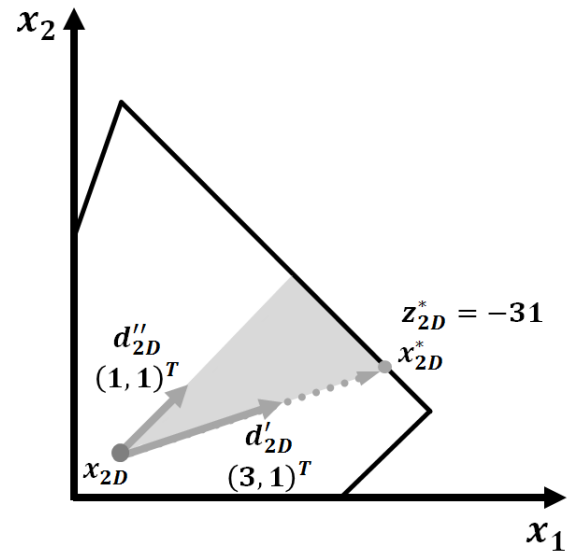
Figure 5.1(b) demonstrates the process when two search directions $d'_{2D} = (3, 1)^T$ and $d''_{2D} = (1, 1)^T$ are selected. In such a case, d_{1D} is contained in the nonnegative linear combination of d'_{2D} and d''_{2D} since $(\theta'_{2D}, \theta''_{2D}) = (\frac{1}{2}, \frac{1}{2})$ and $d_{1D} = \frac{1}{2}d'_{2D} + \frac{1}{2}d''_{2D}$. Solving a 2DSLP results in $(\lambda'_{2D}, \lambda''_{2D}) = (2, 0)$. Consequently, $x_{2D}^* = x_{2D} + \lambda'_{2D} d'_{2D} + \lambda''_{2D} d''_{2D} = (7, 3)$ and $z_{2D}^* = -31$. If $d'_{2D} = (5, 1)^T$ and $d''_{2D} = (1, 5)^T$ are chosen, as depicted in Figure 5.1(c), then $d_{1D} = \frac{3}{8}d'_{2D} + \frac{1}{8}d''_{2D}$, and 2DSLP returns $(\lambda'_{2D}, \lambda''_{2D}) = (\frac{4}{3}, 0)$ such that $x_{2D}^* = (\frac{23}{3}, \frac{7}{3})$ and $z_{2D}^* = -33$. In this case, not only the z value from Figure 5.1(c) is better than the z value from Figure 5.1(a), but it is superior to the z value from Figure 5.1(b).

Figure 5.1(d) shows the case when $d'_{2D} = (2, 0)^T$ and $d''_{2D} = (0, 1)^T$ are selected. The solution to the corresponding 2DSLP results in $(\lambda'_{2D}, \lambda''_{2D}) = (\frac{7}{2}, 1)$, and $x_{2D}^* = (8, 2)$ with $z_{2D}^* = -34$. Observe that this solution is also the optimal solution to the LP. In this case, d_{1D} is orthogonally partitioned so that $(\theta'_{2D}, \theta''_{2D}) = (1, 1)$ and $d_{1D} = d'_{2D} + d''_{2D}$. Consequently, by

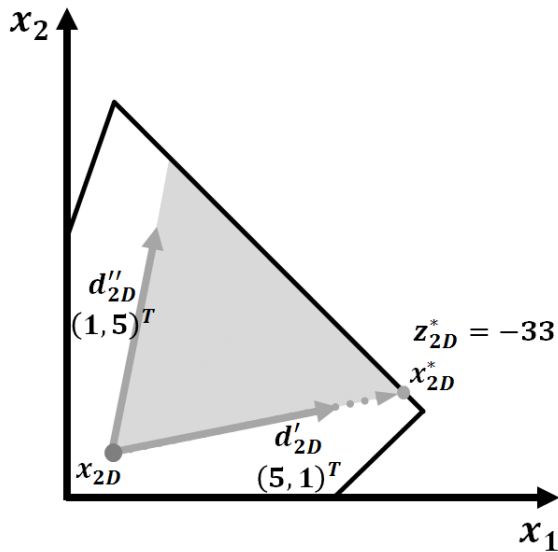
choosing two search directions that are orthogonal to each other, both principles presented in this dissertation are satisfied.



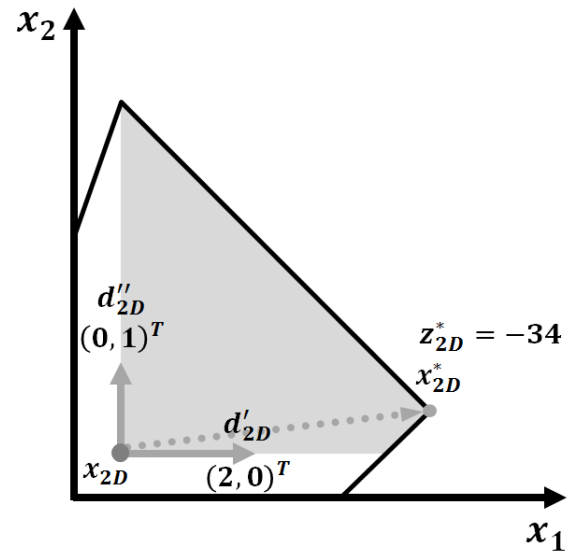
(a) One single search direction $d_{1D} = (2, 1)^T$



(b) Two search directions $d'_{2D} = (3, 1)^T$ and $d''_{2D} = (1, 1)^T$ where $(\theta'_{2D}, \theta''_{2D}) = (\frac{1}{2}, \frac{1}{2})$ and $d_{1D} = \frac{1}{2}d'_{2D} + \frac{1}{2}d''_{2D}$



(c) Two search directions $d'_{2D} = (5, 1)^T$ and $d''_{2D} = (1, 5)^T$ where $(\theta'_{2D}, \theta''_{2D}) = (\frac{3}{8}, \frac{1}{8})$ and $d_{1D} = \frac{3}{8}d'_{2D} + \frac{1}{8}d''_{2D}$



(d) Two search directions $d'_{2D} = (2, 0)^T$ and $d''_{2D} = (0, 1)^T$ where $(\theta'_{2D}, \theta''_{2D}) = (1, 1)$ and $d_{1D} = d'_{2D} + d''_{2D}$

Figure 5.1: Graphical representation of Example 5.2.1

While defining two orthogonal search directions using the affine scaling and logarithmic barrier search directions described in Chapter 5.1 satisfy both principles, these search directions would likely violate the feasibility conditions. Observe that the search directions from the previous section are derived from the objective function. Applying these principles to the objective function instead, results in two search directions that are feasible and potentially improving. The following sections present primal and dual affine scaling and logarithmic barrier two dimensional search interior point algorithms where the objective function is orthogonally partitioned to derive two search directions.

5.2.2 Two Dimensional Search Primal Affine Scaling and Logarithmic Barrier Interior Point Algorithms

The two dimensional search primal affine scaling interior point algorithm ($2\text{DIM}_{\text{aff}\perp}^{\text{P}}$) and the two dimensional search primal logarithmic barrier interior point algorithm ($2\text{DIM}_{\text{log}\perp}^{\text{P}}$) are first presented. These methods are denoted with a \perp because search directions are derived from orthogonal vectors. Since both algorithms are similar, they are described jointly in Algorithm 5.1 using the same framework. Similarly to $1\text{DIM}_{\text{aff}}^{\text{P}}$ and $1\text{DIM}_{\text{log}}^{\text{P}}$, the three primary steps of these methods are: (1) from a feasible interior solution x_{2D}^k , compute two feasible and potential improving search directions $d_{2D}^{k'}$ and $d_{2D}^{k''}$; (2) determine a step size $(\lambda_{2D}^{k'*}, \lambda_{2D}^{k''*})$ for each of the two search directions by solving a 2DSLP; (3) calculate a new and improved feasible interior solution x_{2D}^{k+1} from the 2DSLP's result.

Algorithm 5.1 requires a PLP $_x$, an initial solution $x_{2D}^0 > 0$ with $Ax_{2D}^0 = b$, an $\alpha \in (0, 1)$, and a sufficiently small number $\epsilon > 0$ as input. Additional input to $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ includes a sufficiently large positive number μ_{2D}^0 and some $\beta \in (0, 1)$. Both methods randomly partition the objective function vector c at each iteration k into two new vectors $c^{k'}$ and $c^{k''}$. When a particular element of $c^{k'}$ is assigned with an element from c , the corresponding element from $c^{k''}$ is necessarily assigned to 0 and vice versa.

Algorithm 5.1 : Two Dimensional Search Primal Affine Scaling Interior Point Algorithm (2DIM_{aff⊥}^P) and Two Dimensional Search Primal Logarithmic Barrier Interior Point Algorithm (2DIM_{log⊥}^P)

```

1: begin
2:   Let  $k = 0$ ,  $x_{2D}^0 > 0$  with  $Ax_{2D}^0 = b$ ,  $\alpha \in (0, 1)$ , and  $\epsilon > 0$  be sufficiently small. For
   2DIMlog⊥P, let  $\mu_{2D}^0 > 0$  and  $\beta \in (0, 1)$ ;
3:   for each  $i \in N$  do
4:     if  $u \sim U(0, 1) \leq \frac{1}{2}$  then
5:        $c_i^{k'} = c_i$  and  $c_i^{k''} = 0$ ;
6:     else
7:        $c_i^{k'} = 0$  and  $c_i^{k''} = c_i$ ;
8:   if either  $c_i^{k'} = 0$  or  $c_i^{k''} = 0$  for every  $i \in N$  then choose an arbitrary  $p \in N$  such
   that  $c_p^{k'} \neq 0$  or  $c_p^{k''} \neq 0$  and  $c_p^{k'} \leftrightarrow c_p^{k''}$ ;
9:   if 2DIMaff⊥P then  $d_{2D}^{k'} = dx_{2D}^{k'} = -X_k^2(c^{k'} - A^T(AX_k^2A^T)^{-1}AX_k^2c^{k'})$ ;
    $d_{2D}^{k''} = dx_{2D}^{k''} = -X_k^2(c^{k''} - A^T(AX_k^2A^T)^{-1}AX_k^2c^{k''})$ ;
10:  if 2DIMlog⊥P then  $d_{2D}^{k'} = \frac{1}{\mu_{2D}^k}dx_{2D}^{k'} + X_k(e - X_kA^T(AX_k^2A^T)^{-1}AX_k e)$ ;
    $d_{2D}^{k''} = \frac{1}{\mu_{2D}^k}dx_{2D}^{k''} + X_k(e - X_kA^T(AX_k^2A^T)^{-1}AX_k e)$ ;
11:  Solve the following 2DSLP:

           minimize  $z = c^T d_{2D}^{k'} \lambda_{2D}^{k'} + c^T d_{2D}^{k''} \lambda_{2D}^{k''}$ 
           subject to  $d_{2D}^{k'} \lambda_{2D}^{k'} + d_{2D}^{k''} \lambda_{2D}^{k''} \geq -x_{2D}^k$ 
                    $\lambda_{2D}^{k'}, \lambda_{2D}^{k''} \geq 0$ ;

12:  if 2DSLP is unbounded then
13:    return PLP $x$  is unbounded;
14:  else
15:     $x_{2D}^{k+1} = x_{2D}^k + \alpha(\lambda_{2D}^{k'*} d_{2D}^{k'} + \lambda_{2D}^{k''*} d_{2D}^{k''})$ ;
16:    if  $\frac{|c^T x_{2D}^{k+1} - c^T x_{2D}^k|}{|c^T x_{2D}^k|} < \epsilon$  then
17:      return PLP $x$  is optimal,  $x^* = x_{2D}^{k+1}$ , and  $z^* = c^T x_{2D}^{k+1}$ ;
18:    else
19:      if 2DIMlog⊥P then calculate  $\mu_{2D}^{k+1} = \beta \mu_{2D}^k$ ;
20:       $k \leftarrow k + 1$ ;
21:      goto 3;
22: end

```

Both $-c^{k'}$ and $-c^{k''}$ are projected onto the null space of A to generate search directions $d_{2D}^{k'}$ and $d_{2D}^{k''}$ that are feasible. Observe that both $d_{2D}^{k'}$ and $d_{2D}^{k''}$ are not necessarily improving because $-c$ is not entirely projected onto the null space of constraint matrix A . In this case, at least one of the two search directions is improving. The following 2DSLP,

$$\begin{aligned}
& \text{minimize } z = c^T d_{2D}^{k'} \lambda_{2D}^{k'} + c^T d_{2D}^{k''} \lambda_{2D}^{k''} \\
& \text{subject to } d_{2D}^{k'} \lambda_{2D}^{k'} + d_{2D}^{k''} \lambda_{2D}^{k''} \geq -x_{2D}^k \\
& \lambda_{2D}^{k'}, \lambda_{2D}^{k''} \geq 0,
\end{aligned}$$

is optimally solved, and its optimal solution, $(\lambda_{2D}^{k'*}, \lambda_{2D}^{k''*})$, determines the next feasible interior solution $x_{2D}^{k+1} = x_{2D}^k + \alpha(\lambda_{2D}^{k'*} d_{2D}^{k'} + \lambda_{2D}^{k''*} d_{2D}^{k''})$. The algorithms then repeat these steps until improvement in the objective function value becomes sufficiently small.

If $1\text{DIM}_{\text{aff}}^{\text{P}}$ and $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ are considered, then clearly search directions $d_{1D}^k = dx_{1D}^k = -X_k^2(I - A^T(AX_k^2 A^T)^{-1}AX_k^2)c$, $d_{2D}^{k'} = dx_{2D}^{k'} = -X_k^2(I - A^T(AX_k^2 A^T)^{-1}AX_k^2)c^{k'}$, and $d_{2D}^{k''} = dx_{2D}^{k''} = -X_k^2(I - A^T(AX_k^2 A^T)^{-1}AX_k^2)c^{k''}$, respectively. Thus, $d_{1D}^k = d_{2D}^{k'} + d_{2D}^{k''}$, Theorem 5.2.1 holds, and the first principle is satisfied. On the other hand, $d_{2D}^{k'}$ and $d_{2D}^{k''}$ are not orthogonal to each other because $d_{2D}^{k'} \cdot d_{2D}^{k''} \neq 0$. To partially accomplish the second principle and create a larger feasible region to 2DSL_P, the objective function c is orthogonally partitioned into $c^{k'}$ and $c^{k''}$, which results in $c = c^{k'} + c^{k''}$ and $c^{k'} \cdot c^{k''} = 0$, and both are used to derive search directions $d_{2D}^{k'}$ and $d_{2D}^{k''}$, respectively.

One may claim that partitioning d_{1D}^k directly into $d_{2D}^{k'}$ and $d_{2D}^{k''}$ using the same scheme presented in Algorithm 5.1 (lines 3-7) satisfies both principles. However, $Adx_{2D}^{k'} \neq 0$ and $Adx_{2D}^{k''} \neq 0$ in this case, which makes both search directions infeasible. A potential future research topic is to develop search directions that are orthogonal to each other in the projected space. Observe that this discussion also applies to both $1\text{DIM}_{\text{log}}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$.

Randomly partitioning c at each iteration is not necessary to satisfy both principles. However, random partitioning exhibited superior performance compared to a static partitioning scheme in preliminary computational experiments. The author believes that randomly partitioning c allows consecutive iterations to search over vastly different spaces in the LP's feasible region. This strategy should require fewer iterations.

The author is confident that all four two dimensional search techniques developed in this chapter correctly solve LPs. However, deriving a proof does not immediately follow the same methodology as other researchers used to prove the convergence of the one dimensional search interior point algorithms (see Chapter 2.2.1). The complicating factor involves the

lack of dual estimates for the corresponding new and improved feasible interior solution at each iteration. In one dimensional search interior point algorithms, these dual estimates are trivially obtained. Unfortunately, the dual estimates are difficult to generate because of the nonnegative linear combination of the two search directions. Since this chapter is primarily focused on the computational benefits of the newly created two dimensional search interior point algorithms, proving convergence for $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$, $2\text{DIM}_{\text{log}\perp}^{\text{P}}$, and also for the algorithms shown in the next section is left as a critical future research topic.

Since $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ improve the objective function value by at least as much as the corresponding one dimensional search version, one should question the additional theoretical and computational effort per iteration required by these techniques. Theoretically, $1\text{DIM}_{\text{aff}}^{\text{P}}$ and $1\text{DIM}_{\text{log}}^{\text{P}}$ compute $d_{1D}^k = dx_{1D}^k$, and the most time consuming step calculates $(AX_k^2 A^T)^{-1}$. As discussed in Chapter 4.2, the time to compute the inverse of an $r \times r$ matrix is $O(I(r))$. Solving a 1DSLIP takes $O(n)$ time. Updating x_{1D}^k also requires $O(n)$ effort and calculating other parameters can be done in $O(1)$ time. Consequently, the theoretical running time per iteration of $1\text{DIM}_{\text{aff}}^{\text{P}}$ and $1\text{DIM}_{\text{log}}^{\text{P}}$ is $O(I(r))$.

Both $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ partition c into $c^{k'}$ and $c^{k''}$ in $O(n)$ time. Computing $d_{2D}^{k'}$ and $d_{2D}^{k''}$ requires $O(I(r))$ each. Solving a 2DSLIP takes $O(n)$ time (Dyer, 1984; Megiddo, 1983) since interior point methods do not require an optimal basis to their subproblems. Moreover, updating the solution x_{2D}^{k+1} also requires $O(n)$ effort. Other operations can be done in $O(1)$ time, and the theoretical running time per iteration of $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ is $O(I(r))$. Therefore, both $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ have the same theoretical running time per iteration as $1\text{DIM}_{\text{aff}}^{\text{P}}$ and $1\text{DIM}_{\text{log}}^{\text{P}}$, respectively. An analysis to the additional computational effort of these algorithms is discussed in Chapter 5.3.

5.2.3 Two Dimensional Search Dual Affine Scaling and Logarithmic Barrier Interior Point Algorithms

The other two algorithms described in this chapter are the two dimensional search dual affine scaling interior point algorithm ($2\text{DIM}_{\text{aff}\perp}^{\text{D}}$) and the two dimensional search dual logarithmic barrier interior point algorithm ($2\text{DIM}_{\text{log}\perp}^{\text{D}}$). Both $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ are shown jointly in Algorithm 5.2 using a similar framework. The input to Algorithm 5.2 is a DLP $_{w,s}$, an initial solution (w_{2D}^0, s_{2D}^0) with $A^T w_{2D}^0 + s_{2D}^0 = c$ and $s_{2D}^0 > 0$, an $\alpha \in (0, 1)$, and $\epsilon > 0$. Moreover, $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ also requires a large positive number μ_{2D}^0 and a $\beta \in (0, 1)$.

Both $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ partition the right-hand side b into $b^{k'}$ and $b^{k''}$ at each iteration k such that when $b_j^{k'}$ is assigned with b_j , then $b_j^{k''} = 0$ for all $j \in R$, and vice versa. This guarantees that $b = b^{k'} + b^{k''}$ and $b^{k'} \cdot b^{k''} = 0$, which partially satisfies the second principle. Both vectors derive search directions such that $dw_{1D}^k = \hat{d}w_{2D}^{k'} + \hat{d}w_{2D}^{k''}$ (or $dw_{1D}^k = \hat{d}w_{2D}^{k'} + \hat{d}w_{2D}^{k''}$ for $2\text{DIM}_{\text{log}\perp}^{\text{D}}$) and $ds_{1D}^k = ds_{2D}^{k'} + ds_{2D}^{k''}$. Thus, Corollary 5.2.1 holds and the second principle is satisfied.

Algorithm 5.2 creates a 2DSLIP from search directions with respect to the slack variables. This is because s is the variable placed near the center of the LP's feasible region and w is unrestricted. However, each 2DSLIP is maximized with respect to the dual search directions. The optimal solution to 2DSLIP, $(\rho_{2D}^{k'*}, \rho_{2D}^{k''*})$, is computed and both w_{2D}^{k+1} and s_{2D}^{k+1} are updated.

The analysis performed in Chapter 5.2.2 with respect to the theoretical running time per iteration of $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ also follows for $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$. That is, $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ have the same theoretical running time as $1\text{DIM}_{\text{aff}}^{\text{D}}$ and $1\text{DIM}_{\text{log}}^{\text{D}}$, respectively, and this is $O(I(r))$. One should question whether these two dimensional search interior point algorithms can solve LPs faster than the corresponding one dimensional search version. The following section demonstrates that the additional computational effort is minimal and these novel two dimensional search techniques can improve both the number of iterations and solution time.

Algorithm 5.2 : Two Dimensional Search Dual Affine Scaling Interior Point Algorithm (2DIM_{aff⊥}^D) and Two Dimensional Search Dual Logarithmic Barrier Interior Point Algorithm (2DIM_{log⊥}^D)

1: **begin**
2: Let $k = 0$, (w_{2D}^0, s_{2D}^0) with $A^T w_{2D}^0 + s_{2D}^0 = c$ and $s_{2D}^0 > 0$, $\alpha \in (0, 1)$, and $\epsilon > 0$ be sufficiently small. For 2DIM_{log⊥}^D, let $\mu_{2D}^0 > 0$ and $\beta \in (0, 1)$;
3: **for each** $j \in R$ **do**
4: **if** $u \sim U(0, 1) \leq \frac{1}{2}$ **then**
5: $b_j^{k'} = b_j$ and $b_j^{k''} = 0$;
6: **else**
7: $b_j^{k'} = 0$ and $b_j^{k''} = b_j$;
8: **if** either $b_j^{k'} = 0$ or $b_j^{k''} = 0$ for every $j \in R$ **then** choose an arbitrary $q \in R$ such that $b_q^{k'} \neq 0$ or $b_q^{k''} \neq 0$ and $b_q^{k'} \leftrightarrow b_q^{k''}$;
9: **if** 2DIM_{aff⊥}^D **then** $\hat{d}w_{2D}^{k'} = (AS_k^{-2}A^T)^{-1}b^{k'}$, $d_{2D}^{k'} = ds_{2D}^k = -A^T \hat{d}w_{2D}^{k'}$;
 $\hat{d}w_{2D}^{k''} = (AS_k^{-2}A^T)^{-1}b^{k''}$, $d_{2D}^{k''} = ds_{2D}^k = -A^T \hat{d}w_{2D}^{k''}$;
10: **if** 2DIM_{log⊥}^D **then** $\hat{d}w_{2D}^{k'} = \frac{1}{\mu_{2D}^k} \hat{d}w_{2D}^{k'} - (AS_k^{-2}A^T)^{-1}AS_k^{-1}e$, $d_{2D}^{k'} = ds_{2D}^k = -A^T \hat{d}w_{2D}^{k'}$;
 $\hat{d}w_{2D}^{k''} = \frac{1}{\mu_{2D}^k} \hat{d}w_{2D}^{k''} - (AS_k^{-2}A^T)^{-1}AS_k^{-1}e$, $d_{2D}^{k''} = ds_{2D}^k = -A^T \hat{d}w_{2D}^{k''}$;
11: Let $z = b^T \hat{d}w_{2D}^{k'} \rho_{2D}^{k'} + b^T \hat{d}w_{2D}^{k''} \rho_{2D}^{k''}$ for 2DIM_{aff⊥}^D and $z = b^T \hat{d}w_{2D}^{k'} \rho_{2D}^{k'} + b^T \hat{d}w_{2D}^{k''} \rho_{2D}^{k''}$ for 2DIM_{log⊥}^D
12: Solve the following 2DSLP:

$$\begin{aligned} & \text{maximize } z \\ & \text{subject to } d_{2D}^{k'} \rho_{2D}^{k'} + d_{2D}^{k''} \rho_{2D}^{k''} \geq -s_{2D}^k \\ & \rho_{2D}^{k'}, \rho_{2D}^{k''} \geq 0; \end{aligned}$$

13: **if** the 2DSLP is unbounded **then**
14: **return** DLP_{w,s} is unbounded;
15: **else**
16: **if** 2DIM_{aff⊥}^D **then** $w_{2D}^{k+1} = w_{2D}^k + \alpha(\rho_{2D}^{k'*} \hat{d}w_{2D}^{k'} + \rho_{2D}^{k''*} \hat{d}w_{2D}^{k''})$;
17: **if** 2DIM_{log⊥}^D **then** $w_{2D}^{k+1} = w_{2D}^k + \alpha(\rho_{2D}^{k'*} \hat{d}w_{2D}^{k'} + \rho_{2D}^{k''*} \hat{d}w_{2D}^{k''})$;
18: $s_{2D}^{k+1} = s_{2D}^k + \alpha(\rho_{2D}^{k'*} d_{2D}^{k'} + \rho_{2D}^{k''*} d_{2D}^{k''})$;
19: **if** $\frac{|b^T w_{2D}^{k+1} - b^T w_{2D}^k|}{|b^T w_{2D}^k|} < \epsilon$ **then**
20: **return** DLP_{w,s} is optimal, $(w_{2D}^*, s_{2D}^*) = (w_{2D}^{k+1}, s_{2D}^{k+1})$, and $z^* = b^T w_{2D}^{k+1}$;
21: **else**
22: **if** 2DIM_{log⊥}^D **then** calculate $\mu_{2D}^{k+1} = \beta \mu_{2D}^k$;
23: $k \leftarrow k + 1$;
24: **goto** 3;
25: **end**

5.3 Computational Study

To validate the computational effectiveness of the two dimensional search interior point algorithms, all eight techniques presented in this chapter were implemented in Python 3.6.6. Computational experiments were tested on an Intel® Core™ i7-6700 3.40GHz processor with 32 GB of RAM. The following sections describe the major implementation details along with the instances and computational results.

5.3.1 Implementation Details

To generate an initial feasible interior solution, the Big- M method was implemented (see [Fang and Puthenpura \(1993\)](#) for additional details). For the primal affine scaling and logarithmic barrier methods, an artificial variable associated with a large positive number in the objective function was added to PLP_x , so the problem had $n + 1$ variables and r constraints. Thus, the initial feasible interior solution became $(1, 1, \dots, 1)^T \in \mathbb{R}^{n+1}$ and the residuals were assigned to the constraint values of the artificial variable. This modified PLP_x was solved with $1DIM_{\text{aff}}^P$, $2DIM_{\text{aff}\perp}^P$, $1DIM_{\text{log}}^P$, and $2DIM_{\text{log}\perp}^P$ until these algorithms achieved an optimal solution. Observe that this solution is also an optimal solution to the original PLP_x .

For the dual affine scaling and logarithmic barrier techniques, the Big- M method added an additional constraint associated with a large positive number in the right-hand side of PLP_x , which corresponds to adding an artificial variable to $DLP_{w,s}$. In such a case, this artificial variable began with a negative value and the modified problem was solved with $1DIM_{\text{aff}}^D$, $2DIM_{\text{aff}\perp}^D$, $1DIM_{\text{log}}^D$, and $2DIM_{\text{log}\perp}^D$ until the artificial variable became nonnegative. The solution obtained became the initial feasible interior solution to the original problem.

To avoid numerical instability issues, the large positive number in the primal and dual versions of the Big- M method was determined differently for each instance. While the majority of instances converged with a reasonable M value, some other problems required a larger number. To map this number with each instance, M was determined as a function of the largest objective function coefficient (primal) or the largest right-hand side coefficient

(dual). However, M was identical when solving a particular instance with both the one and two dimensional search interior point algorithms.

As previously discussed, the most time consuming step of all eight interior point algorithms is to compute $(AX_k^2 A^T)^{-1}$ or $(AS_k^{-2} A^T)^{-1}$. To perform this operation, three methods were considered: (1) inverse calculator contained in the linear algebra package from NumPy (*numpy.linalg.inv*), which utilizes Lapack’s LU factorization; (2) Cholesky decomposition also contained in the linear algebra package from NumPy (*numpy.linalg.cholesky*); (3) Cholesky decomposition in the linear algebra package from SciPy (*scipy.linalg.cholesky*). Experiments demonstrated that not only methods (2) and (3) were numerically more stable than method (1), but also these techniques were about 50% faster. Furthermore, the Cholesky decomposition from NumPy provided a more accurate convergence (small error) than the Cholesky decomposition from SciPy. Consequently, the Cholesky decomposition method from NumPy computed the matrix inverse for all algorithms.

Observe that the algorithm of [Boggs et al. \(1989\)](#) solved 2DSLPS using the general implementation of the simplex method. A small contribution of this research is to implement faster methods to solve 2DSLPS. While the slope algorithm could be easily used to solve these 2DSLPS, this computational study implemented the algorithm from [Dyer \(1984\)](#) in order to avoid the additional sorting step.

Every instance in this computational study was solved by $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$, $2\text{DIM}_{\text{log}\perp}^{\text{P}}$, $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$, and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ 10 different times in order to avoid random anomalies associated with the partitioning of the objective function. Consequently, the number of iterations, solution time, and relative improvement in objective function value reported in Tables 5.2–5.6 correspond to the average of all 10 runs. Furthermore, each instance was solved with $\alpha = 0.99$, $\alpha = 0.95$, and $\alpha = 0.90$. Experiments determined that $\alpha = 0.95$ led the majority of instances to a smoother convergence. Therefore, results reported in this dissertation have $\alpha = 0.95$. For the logarithmic barrier algorithms, $\beta = 0.5$ and $\mu = 1 \times 10^4$.

Since the optimal objective function value is known for all instances presented in this dissertation, the computational study calculated the error with respect to the solution obtained by each of the eight algorithms. Error is defined as:

$$\tau = \frac{|z - z^*|}{|z^*|},$$

where z is the final objective function value obtained with the one dimensional and two dimensional search interior point algorithms, and z^* is the optimal objective function value. Every instance reported has $\tau < 1 \times 10^{-6}$. This implies that all instances in this study converged to an optimal solution.

[Boggs et al.](#) considered two strategies: fully solve each instance with two search directions; start solving each problem with two search directions and switch to a single search direction in the last few iterations. This research also implemented both strategies because the closer $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$, $2\text{DIM}_{\text{log}\perp}^{\text{P}}$, $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$, and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ are to the optimal solution, the smaller the two dimensional subspace is. In this case, the two search directions may behave similar to the single search direction. Experiments showed that solving instances with two search directions from beginning to end performed computationally better. This result supports the conclusion from [Boggs et al.](#), and this computational study only reports results where instances are fully solved with two search directions.

5.3.2 Instances and Computational Results

Similar to the computational study developed to test the double pivot simplex method, this research tested the newly created two dimensional search interior point algorithms with instances from Netlib ([Gay, 1985](#)). From all instances in Netlib, 60 problems were selected. These problems have different number of variables and constraints, more dense and sparse structures, and different convergence profiles. Slack and surplus variables were added to each instance when necessary to make matrix A full row rank. Moreover, additional constraints were added to each instance to represent variables with lower bounds, upper bounds, and fixed bounds. Table [5.1](#) presents the characteristics of these instances with the aforementioned transformations. Observe that several of the problems shown in Table [5.1](#) overlap with Tables [4.5-4.6](#).

Table 5.1: *Number of constraints, variables, and nonzero elements in instances tested from Netlib*

Name	Cont	Var	Nz	Name	Cont	Var	Nz
25FV47 [‡]	1,853	3,424	13,285	SC50A	50	78	160
ADLITTLE	56	138	424	SC50B	50	78	148
AFIRO	27	51	102	SCAGR7	129	185	465
AGG2 [†]	516	758	4740	SCAGR25	471	671	1,725
BANDM [†]	305	472	2,494	SCFXM1 [†]	330	600	2,732
BEACONFD	173	295	3,408	SCFXM2 [†]	660	1,200	5,469
BLEND	74	114	522	SCFXM3 [‡]	990	1,200	5,469
BNL1	1,107	2,282	6,692	SCORPION	948	1,306	2,934
BNL2 [‡]	2,324	4,486	14,996	SCRS8 [‡]	490	1,275	3,288
BORE3D [†]	675	990	2,546	SCSD1 [†]	77	760	2,388
BRANDY	552	801	3,032	SCSD6	147	1,350	4,316
CZPROB [†]	1,158	3,791	11,166	SCTAP1 [†]	300	660	1,872
D2Q06C [‡]	2,171	5,831	33,081	SCTAP2	1,090	2,500	7,334
D6CUBE	1,246	7,430	39,781	SCTAP3	1,480	3,340	9,734
DEGEN2	886	1,420	5,306	SHARE1B [†]	117	253	1,179
DEGEN3	2,937	4,755	29,017	SHARE2B	96	162	777
E226 [‡]	223	472	2,768	SHIP04L	1,110	3,238	8,150
ETAMACRO ^{†‡}	689	1,105	3,115	SHIP04S	1,110	2,568	6,170
FIT1D	1,050	2,075	15,479	SHIP08L	2,174	6,457	16,372
FIT1P [‡]	1,026	2,076	10,666	SHIP08S	2,174	4,561	10,684
GANGES [‡]	1,713	2,110	7,745	SHIP12L	3,241	8,668	21,501
GROW7 [†]	420	581	3,172	SHIP12S	3,241	6,004	13,509
GROW15	900	1,245	6,820	STANDATA	490	1,405	3,492
GROW22	1,320	1,826	10,012	STANDGUB [‡]	816	2,000	4,410
ISRAEL [†]	174	316	2,443	STANDMPS	598	1,405	4,140
KB2	52	77	331	STOCFOR1	117	165	501
QAP8	2,736	4,368	11,856	STOCFOR2	2,157	3,045	9,357
RECIPELP [‡]	207	320	919	TRUSS	1,000	8,806	27,836
SC105	105	163	340	WOOD1P	730	3,324	71,431
SC205 [‡]	205	317	665	WOODW	1,098	8,418	37,487

† and ‡ denote instances solved with the primal (†) and/or dual (‡) logarithmic barrier methods

The study tracked two primary results: number of iterations and solution time. To calculate improvement in the number of iterations, define:

$$\delta_{\text{Iter}} = \left(\frac{\text{Iter}_{1D} - \text{Iter}_{2D}}{\text{Iter}_{1D}} \right) \times 100\%,$$

where Iter_{1D} and Iter_{2D} are the number of iterations obtained with the one dimensional and two dimensional search interior point algorithms, respectively. Similarly, improvement in solution time is defined as:

$$\delta_{\text{Time}} = \left(\frac{\text{Time}_{1D} - \text{Time}_{2D}}{\text{Time}_{1D}} \right) \times 100\%,$$

where Time_{1D} and Time_{2D} are the solution times from the one dimensional and two dimensional search techniques, respectively. These improvements are identical to $\delta_{\frac{\text{DPSM}}{\text{SM}}}$ and $\delta_{\frac{\text{DPSM}}{\text{CPLEX}}}$ defined in Chapters 4.2.2.1 and 4.2.2.3, respectively.

Some instances presented issues with convergence because all eight implemented algorithms depend on many parameters and can be extremely sensitive to numerical errors. There were instances that solved with $1\text{DIM}_{\text{aff}}^{\text{P}}$ but not with $1\text{DIM}_{\text{log}}^{\text{P}}$, or solved with $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ but not with $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$, and vice versa. Consequently, Tables 5.1-5.6 present the results for each instance with either the affine scaling methods or the logarithmic barrier techniques. To distinguish between each algorithm, let a † denote instances solved with $1\text{DIM}_{\text{log}}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$. Similarly, let a ‡ denote problems solved with $1\text{DIM}_{\text{log}}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$. Finally, all remaining instances were solved with $1\text{DIM}_{\text{aff}}^{\text{P}}$, $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$, $1\text{DIM}_{\text{aff}}^{\text{D}}$, and $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$.

Tables 5.2 and 5.3 show the number of iterations and solution time for all instances solved with the primal algorithms. In addition, δ_{Iter} and δ_{Time} are presented. Because the large number of instances, 30 out of the 60 results are presented in Table 5.2 and the other 30 are shown in Table 5.3. On average, the number of iterations is improved by approximately 16% and solution time is improved by nearly 14%. When analyzing problems solved with $1\text{DIM}_{\text{aff}}^{\text{P}}$ and $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ only, this improvement is around 14% and 13%, respectively. This improvement is approximately 21% and 16%, respectively, when only considering problems solved with $1\text{DIM}_{\text{log}}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$.

Table 5.2: Improvement in the number of iterations and solution time of $2DIM_{\text{aff}\perp}^P$ and $2DIM_{\text{log}\perp}^P$ over $1DIM_{\text{aff}}^P$ and $1DIM_{\text{log}}^P$

Name	$1DIM^P$		$2DIM_{\perp}^P$		Improvement (δ)	
	Iter.	Time (s)	Iter.	Time (s)	δ_{Iter}	δ_{Time}
25FV47 [‡]	207	335.36	196	320.73	5.3%	4.4%
ADLITTLE	31	0.20	29	0.17	6.5%	16.4%
AFIRO	16	0.06	14	0.05	12.5%	16.7%
AGG2 [†]	60	5.06	50	4.41	16.7%	12.9%
BANDM [†]	38	1.26	28	0.97	26.3%	23.3%
BEACONFD	17	0.28	13	0.18	23.5%	37.5%
BLEND	34	0.17	24	0.11	29.4%	35.1%
BNL1	97	53.45	91	51.21	6.2%	4.2%
BNL2 [‡]	233	726.98	224	715.11	3.9%	1.6%
BORE3D [†]	50	7.62	32	5.15	36.0%	32.4%
BRANDY	116	10.23	85	7.70	26.7%	24.7%
CZPROB [†]	57	78.10	45	64.51	21.1%	17.4%
D2Q06C [‡]	455	1,912.05	398	1,723.09	12.5%	9.9%
D6CUBE	17	59.66	16	56.87	5.9%	4.7%
DEGEN2	15	3.78	14	3.49	6.7%	7.7%
DEGEN3	34	154.40	33	149.19	2.9%	3.4%
E226 [‡]	44	1.12	40	1.04	9.1%	7.1%
ETAMACRO ^{†‡}	41	7.39	39	7.13	4.9%	3.5%
FIT1D	100	46.19	78	36.68	22.0%	20.6%
FIT1P [‡]	190	87.94	145	66.93	23.7%	23.9%
GANGES [‡]	184	157.30	170	145.94	7.6%	7.2%
GROW7 [†]	26	1.40	22	1.25	15.4%	10.8%
GROW15	229	54.09	170	40.83	25.8%	24.5%
GROW22	208	107.47	213	110.52	-2.4%	-2.8%
ISRAEL [†]	153	2.36	82	1.35	46.4%	43.0%
KB2	105	0.45	72	0.29	31.4%	35.2%
QAP8	13	47.28	12	43.01	7.7%	9.0%
RECIPELP [‡]	50	0.74	43	0.70	14.0%	6.0%
SC105	69	0.46	58	0.37	15.9%	19.3%
SC205 [‡]	163	2.48	89	1.41	45.4%	43.2%
SC50A	34	0.12	30	0.12	11.8%	2.5%

† and ‡ denote instances solved with the primal (†) and/or dual (‡) logarithmic barrier methods

Table 5.3: Improvement in the number of iterations and solution time of $2DIM_{\text{aff}\perp}^P$ and $2DIM_{\text{log}\perp}^P$ over $1DIM_{\text{aff}}^P$ and $1DIM_{\text{log}}^P$ - continued

Name	$1DIM^P$		$2DIM_{\perp}^P$		Improvement (δ)	
	Iter.	Time (s)	Iter.	Time (s)	δ_{Iter}	δ_{Time}
SC50B	39	0.14	35	0.13	10.3%	3.9%
SCAGR7	220	1.95	186	1.74	15.5%	10.8%
SCAGR25	169	10.91	138	9.17	18.3%	16.0%
SCFXM1 [†]	54	2.48	35	1.74	35.2%	29.6%
SCFXM2 [†]	54	10.17	44	8.76	18.5%	13.8%
SCFXM3 [‡]	316	118.05	201	75.47	36.4%	36.1%
SCORPION	218	56.62	209	54.42	4.1%	3.9%
SCRS8 [‡]	99	14.06	88	13.14	11.1%	6.6%
SCSD1 [†]	20	0.66	20	0.72	0.0%	-8.3%
SCSD6	18	1.66	15	1.47	16.7%	11.4%
SCTAP1 [†]	26	1.28	24	1.22	7.7%	4.5%
SCTAP2	245	152.28	239	149.43	2.4%	1.9%
SCTAP3	260	315.07	254	314.20	2.3%	0.3%
SHARE1B [†]	74	0.67	59	0.58	20.3%	14.1%
SHARE2B	35	0.19	24	0.14	31.4%	27.3%
SHIP04L	133	118.76	124	112.33	6.8%	5.4%
SHIP04S	140	91.00	131	86.06	6.4%	5.4%
SHIP08L	162	780.88	149	725.37	8.0%	7.1%
SHIP08S	162	468.99	152	441.92	6.2%	5.8%
SHIP12L	150	1,810.26	134	1,630.13	10.7%	10.0%
SHIP12S	142	1,003.58	133	928.52	6.3%	7.5%
STANDATA	40	6.45	32	5.38	20.0%	16.7%
STANDGUB [‡]	28	9.96	23	7.76	17.9%	22.1%
STANDMPS	48	9.01	36	6.82	25.0%	24.2%
STOCFOR1	67	0.38	52	0.32	22.4%	15.4%
STOCFOR2	404	705.24	369	654.71	8.7%	7.2%
TRUSS	380	1,628.78	351	1,573.51	7.6%	3.4%
WOOD1P	80	53.89	54	38.00	32.5%	29.5%
WOODW	94	386.28	92	385.83	2.1%	0.1%
Average					15.5%	13.9%

[†] and [‡] denote instances solved with the primal ([†]) and/or dual ([‡]) logarithmic barrier methods

Tables 5.4 and 5.5 describe the results for the dual algorithms. Differently than Boggs et al., this computational study implemented the two dimensional search techniques in both Phase 1 and Phase 2 of the Big- M method. Thus, the results in Tables 5.4–5.5 include the combined number of iterations and solution time of both phases. The number of iterations improves by nearly 13% and solution time by around 10%, on average. Problems solved with only $1\text{DIM}_{\text{aff}}^{\text{D}}$ and $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ improve the number of iterations and solution time by approximately 12% and 10%, respectively. When considering only instances solved with $1\text{DIM}_{\text{log}}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$, iterations and solution time are improved by about 16% and 11%, respectively.

5.3.3 Analysis of Results

Overall, the majority of problems showed some improvement. Only four instances had either a negative improvement or no improvement when the primal algorithms are implemented. Furthermore, only four other problems did not improve the number of iterations and/or solution time when the dual algorithms are considered.

One should compare the performance of these two dimensional search interior point algorithm with Boggs et al.’s results. Unfortunately, the work from Boggs et al. did not report solution times. Instead, Boggs et al. only reported the number of iterations for 29 out of the 60 instances shown in Table 5.1. Their method improved the number of iterations by approximately 9%; whereas, $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ decreased the number of iterations by about 12%. Similarly, $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ improved the number of iterations by nearly 16%. Consequently, orthogonally partitioning the objective function to create two search directions reduces the number of iterations by more than the rank-one update or Newton recentering search direction.

Table 5.4: Improvement in the number of iterations and solution time of $2DIM_{\text{aff}\perp}^D$ and $2DIM_{\text{log}\perp}^D$ over $1DIM_{\text{aff}}^D$ and $1DIM_{\text{log}}^D$

Name	$1DIM^D$		$2DIM_{\perp}^D$		Improvement (δ)	
	Iter.	Time (s)	Iter.	Time (s)	δ_{Iter}	δ_{Time}
25FV47 [‡]	59	71.90	55	67.03	6.8%	6.8%
ADLITTLE	28	0.13	27	0.13	3.6%	3.0%
AFIRO	19	0.06	18	0.06	5.3%	3.2%
AGG2 [†]	25	1.45	23	1.38	8.0%	4.6%
BANDM [†]	19	0.49	19	0.52	0.0%	-5.1%
BEACONFD	17	0.20	14	0.18	17.6%	10.4%
BLEND	24	0.13	21	0.10	12.5%	21.1%
BNL1	115	39.11	104	35.52	9.6%	9.2%
BNL2 [‡]	51	114.09	46	103.81	9.8%	9.0%
BORE3D [†]	26	2.45	25	2.39	3.8%	2.3%
BRANDY	33	2.16	29	1.92	12.1%	11.3%
CZPROB [†]	83	53.10	76	49.26	8.4%	7.2%
D2Q06C [‡]	57	165.80	52	154.66	8.8%	6.7%
D6CUBE	23	39.26	19	34.19	17.4%	12.9%
DEGEN2	20	3.65	19	3.43	5.0%	5.9%
DEGEN3	29	74.38	25	64.57	13.8%	13.2%
E226 [‡]	25	0.57	22	0.53	12.0%	7.3%
ETAMACRO ^{†‡}	30	3.82	25	3.35	16.7%	12.4%
FIT1D	74	21.91	66	19.90	10.8%	9.1%
FIT1P [‡]	30	10.98	23	8.38	23.3%	23.7%
GANGES [‡]	18	11.83	15	9.66	16.7%	18.3%
GROW7 [†]	39	1.47	38	1.47	2.6%	-0.3%
GROW15	40	6.52	40	6.65	0.0%	-2.0%
GROW22	41	13.68	40	13.38	2.4%	2.2%
ISRAEL [†]	32	0.47	30	0.44	6.3%	4.5%
KB2	20	0.07	19	0.07	5.0%	3.4%
QAP8	33	69.23	29	61.25	12.1%	11.5%
RECIPELP [‡]	27	0.35	18	0.31	33.3%	11.1%
SC105	17	0.10	16	0.10	5.9%	5.9%
SC205 [‡]	33	0.48	26	0.43	21.2%	10.6%
SC50A	14	0.05	12	0.05	14.3%	11.5%

† and ‡ denote instances solved with the primal (†) and/or dual (‡) logarithmic barrier methods

Table 5.5: Improvement in the number of iterations and solution time of $2DIM_{\text{aff}\perp}^D$ and $2DIM_{\text{log}\perp}^D$ over $1DIM_{\text{aff}}^D$ and $1DIM_{\text{log}}^D$ - continued

Name	1DIM ^D		2DIM _⊥ ^D		Improvement (δ)	
	Iter.	Time (s)	Iter.	Time (s)	δ_{Iter}	δ_{Time}
SC50B	15	0.06	11	0.05	26.7%	19.6%
SCAGR7	33	0.25	27	0.22	18.2%	10.2%
SCAGR25	43	2.03	36	1.74	16.3%	14.2%
SCFXM1 [†]	26	0.79	20	0.65	23.1%	18.0%
SCFXM2 [†]	26	2.88	19	2.13	26.9%	26.1%
SCFXM3 [‡]	23	6.64	18	5.49	21.7%	17.4%
SCORPION	37	6.56	34	6.18	8.1%	5.8%
SCRS8 [‡]	39	4.47	37	4.43	5.1%	0.9%
SCSD1 [†]	14	0.34	9	0.25	35.7%	25.7%
SCSD6	16	1.06	10	0.75	37.5%	29.7%
SCTAP1 [†]	24	0.87	22	0.76	8.3%	12.5%
SCTAP2	24	8.75	23	8.73	4.2%	0.2%
SCTAP3	24	16.23	23	15.60	4.2%	3.9%
SHARE1B [†]	23	0.17	19	0.15	17.4%	12.4%
SHARE2B	62	0.33	49	0.28	21.0%	14.4%
SHIP04L	58	28.72	55	27.82	5.2%	3.1%
SHIP04S	54	19.83	52	19.38	3.7%	2.3%
SHIP08L	56	134.90	54	131.96	3.6%	2.2%
SHIP08S	54	85.15	52	83.21	3.7%	2.3%
SHIP12L	67	385.83	65	369.89	3.0%	4.1%
SHIP12S	59	221.67	57	217.05	3.4%	2.1%
STANDATA	162	17.42	128	14.24	21.0%	18.2%
STANDGUB [‡]	39	10.89	34	9.75	12.8%	10.4%
STANDMPS	210	25.56	165	20.93	21.4%	18.1%
STOCFOR1	21	0.12	19	0.11	9.5%	5.1%
STOCFOR2	67	73.89	57	63.10	14.9%	14.6%
TRUSS	36	72.82	30	62.15	16.7%	14.7%
WOOD1P	24	9.47	18	6.99	25.0%	26.2%
WOODW	40	77.29	35	72.40	12.5%	6.3%
Average					12.6%	9.9%

† and ‡ denote instances solved with the primal (†) and/or dual (‡) logarithmic barrier methods

Even though all eight techniques discussed in this chapter have the same theoretical running time per iteration, one would expect some additional computational effort by the newly created techniques at each step. Partitioning the objective function is an extra computational task incorporated in these algorithms. Although two search directions are computed instead of one, the most time consuming step can be calculated once for both of them. This can be done by expressing both search directions in terms of the partitioned vectors, and calculating the identical term once for both of them. Thus, only an extra matrix-vector multiplication is required, and this time can be reduced due to the many zeros in both orthogonal vectors. Solving 2DSLPS is computationally more expensive in practice than 1DSLPS, even though their theoretical running times are the same. Finally, calculating a new and improved solution doubles the time because of the additional search direction.

Despite the fact that the newly created two dimensional search interior point algorithms require more computational effort than their corresponding one dimensional search interior point algorithms, this additional computational effort is somewhat simple. Observe that Tables 5.2–5.5 show that the average improvement in solution time closely follows the average improvement in the number of iterations. This result implies that the additional computational effort is minimal since it only affected the solution time of $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ by approximately 2%, and $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ by only 3%.

The author believes that the improvement in solution time is primarily linked to a significant improvement in objective function value per iteration (Theorem 5.2.1 and Corollary 5.2.1). To check this claim, the objective function value obtained by moving through a one dimensional search direction (z_{1D}) and a two dimensional search direction (z_{2D}) were computed from the same feasible interior solution at each iteration k . Similar to Δ described in Chapter 4.2.2.3, relative improvement in the objective function value at each iteration k is defined as:

$$\Delta^k = \left(\frac{z_{2D} - z_{\text{current}}}{z_{1D} - z_{\text{current}}} - 1 \right) \times 100\%,$$

Table 5.6: Average relative improvement in the objective function value per iteration ($\bar{\Delta}$)

Name	Improvement ($\bar{\Delta}$)		Name	Improvement ($\bar{\Delta}$)	
	$\frac{2\text{DIM}^{\dagger}_{\text{P}}}{1\text{DIM}^{\dagger}_{\text{P}}}$	$\frac{2\text{DIM}^{\ddagger}_{\text{D}}}{1\text{DIM}^{\ddagger}_{\text{D}}}$		$\frac{2\text{DIM}^{\dagger}_{\text{P}}}{1\text{DIM}^{\dagger}_{\text{P}}}$	$\frac{2\text{DIM}^{\ddagger}_{\text{D}}}{1\text{DIM}^{\ddagger}_{\text{D}}}$
25FV47 [‡]	18.7%	26.8%	SC50A	7.0%	9.5%
ADLITTLE	12.6%	28.7%	SC50B	6.3%	14.4%
AFIRO	19.5%	12.6%	SCAGR7	18.5%	45.1%
AGG2 [†]	4.5%	9.7%	SCAGR25	27.2%	40.1%
BANDM [†]	26.4%	44.7%	SCFXM1 [†]	18.1%	14.6%
BEACONFD	6.3%	33.9%	SCFXM2 [†]	38.9%	9.0%
BLEND	16.4%	10.0%	SCFXM3 [‡]	14.2%	13.4%
BNL1	13.8%	54.8%	SCORPION	2.1%	53.2%
BNL2 [‡]	10.8%	16.1%	SCRS8 [‡]	16.3%	8.1%
BORE3D [†]	30.7%	25.8%	SCSD1 [†]	12.2%	96.5%
BRANDY	31.3%	24.3%	SCSD6	14.3%	6.2%
CZPROB [†]	12.0%	64.3%	SCTAP1 [†]	13.6%	21.9%
D2Q06C [‡]	19.4%	27.5%	SCTAP2	8.1%	16.7%
D6CUBE	5.4%	8.1%	SCTAP3	8.6%	9.5%
DEGEN2	6.8%	3.3%	SHARE1B [†]	36.6%	57.7%
DEGEN3	6.7%	3.6%	SHARE2B	37.0%	89.8%
E226 [‡]	9.2%	13.8%	SHIP04L	6.0%	11.3%
ETAMACRO ^{†‡}	4.2%	8.1%	SHIP04S	1.8%	11.9%
FIT1D	40.7%	44.6%	SHIP08L	6.5%	87.0%
FIT1P [‡]	31.6%	62.2%	SHIP08S	7.6%	46.8%
GANGES [‡]	2.7%	1.3%	SHIP12L	14.5%	49.6%
GROW7 [†]	8.0%	6.7%	SHIP12S	12.4%	7.1%
GROW15	6.4%	3.0%	STANDATA	51.9%	6.5%
GROW22	5.7%	6.0%	STANDGUB [‡]	40.3%	68.8%
ISRAEL [†]	24.5%	50.0%	STANDMPS	8.9%	5.9%
KB2	31.5%	17.5%	STOCFOR1	26.0%	10.4%
QAP8	0.7%	0.0%	STOCFOR2	22.3%	7.7%
RECIPELP [‡]	8.7%	6.4%	TRUSS	14.3%	40.0%
SC105	5.5%	16.3%	WOOD1P	82.0%	64.9%
SC205 [‡]	30.8%	16.7%	WOODW	45.6%	45.3%
Average			17.8%	26.9%	

† and ‡ denote instances solved with the primal (†) and/or dual (‡) logarithmic barrier methods

where z_{current} is the z value of the current solution. The average relative improvement in objective function value per iteration ($\overline{\Delta}$) is obtained by averaging all values of Δ^k . That is, $\overline{\Delta}$ represents how much more, on average, the newly created two dimensional search interior point algorithms improve the objective function value per iteration than the classical one dimensional search interior point algorithms.

Table 5.6 shows that on average, $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{P}}$ improve the objective function value at each iteration by approximately 18% more than $1\text{DIM}_{\text{aff}}^{\text{P}}$ and $1\text{DIM}_{\text{log}}^{\text{P}}$. This improvement becomes 27% when comparing $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$ and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ with the corresponding one dimensional search version. When analyzing just instances solved with the affine scaling techniques, these improvements are 18% and 28%, respectively. Similarly, when analyzing only the problems solved with the logarithmic barrier methods, these improvements are 19% and 22%, respectively.

To summarize, $2\text{DIM}_{\text{aff}\perp}^{\text{P}}$, $2\text{DIM}_{\text{log}\perp}^{\text{P}}$, $2\text{DIM}_{\text{aff}\perp}^{\text{D}}$, and $2\text{DIM}_{\text{log}\perp}^{\text{D}}$ are shown to be more effective than $1\text{DIM}_{\text{aff}}^{\text{P}}$, $1\text{DIM}_{\text{log}}^{\text{P}}$, $1\text{DIM}_{\text{aff}}^{\text{D}}$, and $1\text{DIM}_{\text{log}}^{\text{D}}$ based on the results of this computational study. Overall, the two dimensional search interior point algorithms improved the number of iterations by about 15% and solution time by approximately 12%. This improvement may be attributed to the minimal extra computational effort required by these newly developed algorithms and a significant average relative improvement in objective function value per iteration of approximately 23%.

Chapter 6

Conclusions and Future Research

This dissertation introduces a change of paradigm in optimization methods. The majority of algorithms designed to solve optimization models can be categorized as one dimensional search techniques. That is, these iterative methods determine an improved solution at each iteration by solving a one dimensional subspace problem. In contrast, this dissertation's research creates several algorithms to solve linear programming problems where the movement between solutions is determined by solving a two dimensional subspace linear program. These novel two dimensional search algorithms, when tested over numerous random and benchmark linear programs, demonstrate the potential of multidimensional searches in optimization methods. Overall, the number of iterations and solution time are reduced by an average of approximately 25%. The following sections describe the major conclusions obtained with this dissertation and potential topics for future research.

6.1 Conclusions

While one dimensional search techniques are limited by all solutions contained in a single ray, two dimensional search methods benefit from the addition of a second search direction. This second search direction allows these algorithms to search over a plane. Consequently, two dimensional search methods improve the objective function value at each iteration by at least

as much the corresponding one dimensional search version under broad conditions. On the other hand, solving a two dimensional subspace problem likely requires more computational effort than solving a one dimensional subspace problem. Therefore, effective techniques to solve this two dimensional subspace problem must be created in order to counteract the additional effort.

To accomplish this critical result, this dissertation presents the slope algorithm (SA), an effective technique to find an optimal basis and an optimal solution to linear programs with only two variables. The slope algorithm evaluates the “slope” of each constraint in a two variable linear program and contrasts these values with the slope formed by both cost coefficients. Sufficient conditions guarantee that the slope algorithm either identifies a ray of unboundedness, or finds an optimal solution and also an optimal basis. By deriving theoretical results on nonsupportive constraints, the slope algorithm’s running time is bounded by a linear logarithmic function in terms of the number of constraints.

While the slope algorithm is an interesting procedure to quickly solve two variable linear programs, its true benefit is realized as a pivoting technique incorporated into a primal simplex framework. This development results in the double pivot simplex method (DPSM), an advancement to the well-known simplex method. The double pivot simplex method pivots up to two nonbasic entering variables with two basic leaving variables at each iteration instead of one single variable from the simplex method. In this case, the slope algorithm replaces the minimum ratio test and determines both leaving basic variables. Because of the slope algorithm’s low complexity, an iteration of the double pivot simplex method has the same theoretical running time as an iteration of the simplex method. This dissertation also proves that given a nondegenerate linear program, the double pivot simplex method solves the problem within a finite number of steps. In addition, the double pivot simplex method also diminishes some of the negative effects caused by degenerate linear programs.

Computational experiments tested the double pivot simplex method on random linear programs and benchmark instances from Netlib and MIPLIB. When compared to the simplex method, computational results show an improvement in the number of pivots and solution time of nearly 17% in random dense instances and 30% in random sparse problems, on av-

erage. Furthermore, the number of pivots are improved by about 41% when tested over benchmark instances. When comparing to the primal simplex implementation of CPLEX, a state-of-the-art high performance mathematical programming solver, the double pivot simplex method reduces the number of pivots by approximately 22%, on average, in benchmark linear programs.

Similar to the slope algorithm, this dissertation's research also creates the ratio algorithm (RA), a method to determine an optimal basis and an optimal solution of linear programs with only two constraints. The ratio algorithm compares the ratio of all variables, which are formed by the constraint values of the two constraints, with the ratio formed by the right-hand side values. The ratio algorithm determines two variables that define an optimal basis to such a problem following infeasibility conditions and optimality properties. The complexity of the ratio algorithm is bounded by sorting a list of numbers, which size equals the number of variables, and is theoretically faster than all other existing techniques capable of finding an optimal basis of such simple problems. When the ratio algorithm is implemented within a dual simplex framework, it results in the double pivot dual simplex method (DPDSM), where two variables are exchanged at each iteration instead of one.

All of the aforementioned algorithms are designed from the context of a simplex framework. This dissertation's research also creates two dimensional search techniques using an interior point framework. In this case, one of the critical steps is to determine two effective search directions. Hence, this dissertation provides two principles for determining search directions. First, the selected two search directions should lead to a solution with an objective function value that is at least as good as the objective function value obtained with a single search direction. The second principle focuses on selecting two search directions that increase the feasible region of two dimensional subspace linear programs.

Using both of these principles, novel two dimensional search interior point algorithms are presented. First, the two dimensional search primal affine scaling interior point algorithm ($2\text{DIM}_{\text{aff}\perp}^{\text{P}}$) and the two dimensional search primal logarithmic barrier interior point algorithm ($2\text{DIM}_{\text{log}\perp}^{\text{P}}$) are created. From a feasible interior solution, each step of these techniques randomly partitions the objective function into two orthogonal vectors. These vectors are

used to generate two feasible and potentially improving search directions. Because of this partitioning scheme, one can represent a one dimensional search direction as a nonnegative linear combination of both two dimensional search directions, which satisfies the first principle. Furthermore, the orthogonality condition satisfies the second principle. When intersected with the feasible region of the linear program, these search directions create a two dimensional subspace linear program that is used to determine a new and improved feasible interior solution.

Computational experiments tested both primal two dimensional search interior point algorithms jointly versus the corresponding one dimensional search version. Experiments were tested on benchmark linear programs from Netlib. On average, the number of iterations is improved by 16% and solution time is reduced by 14%. When analyzing the average relative improvement in the objective function value per iteration, both primal methods determine successive feasible interior solutions that are 18% better than the classical one dimensional search primal affine scaling and logarithmic barrier interior point algorithms.

This dissertation's research also creates the two dimensional search dual affine scaling interior point algorithm ($2\text{DIM}_{\text{aff}\perp}^{\text{D}}$) and the two dimensional search dual logarithmic barrier interior point algorithm ($2\text{DIM}_{\text{log}\perp}^{\text{D}}$). These techniques follow a similar procedure to the primal methods, but search directions are computed by solving the corresponding dual problem, and the right-hand side vector is randomly and orthogonally partitioned at each step. Computational experiments in this case show an improvement of approximately 13% in the number of iterations and 10% in solution time. The average relative improvement in objective function value per iteration is approximately 27%.

When all computational results presented in this dissertation's computational studies are averaged, one can see that the number of iterations and solution time are reduced by nearly 25%. This substantial improvement demonstrates the potential of two dimensional searches in linear programming. If implemented into state-of-the-art commercial and open source solvers, these and other future developed multidimensional search algorithms could potentially decrease the solution time to solve not only linear programs, but also other critical classes of optimization models.

6.2 Future Research

This dissertation sets the stage for other important research in multidimensional searches applied to optimization methods. During the development of this dissertation's research, some critical questions emerged. Answering these research questions could substantially improve the understanding of multidimensional search algorithms. These ideas are theoretical, algorithmic, and computational in nature. The following sections describe some of these future research ideas.

6.2.1 Future Simplex Framework Research

One potential future research topic is to determine the benefit of double pivots in current state-of-the-art commercial and/or open source mathematical programming solvers. Consequently, the primary research task on this topic should develop an efficient method to update the basis factorization with two variables simultaneously. A complete implementation on these solvers would enable experiments to provide new computational results to the double pivot simplex method. Furthermore, this implementation would more accurately test the computational effectiveness of the double pivot dual simplex method.

Research could also extend double pivots to other simplex framework results. For instance, creating a primal and dual double pivot simplex method with lower and upper bounds is a potential future study. If computationally more effective than the corresponding single pivot version, these newly developed techniques could be implemented within the branch and bound algorithm to solve integer and mixed integer programs. That is, these methods could solve the linear relaxation problem of each child from the branching tree. In this case, double pivots would benefit not only linear programming but also integer programming.

Another research topic could investigate whether or not double pivots prevent cycling on degenerate linear programs. To answer this critical question, one should find a small instance where the double pivot simplex method or the double pivot dual simplex method cycle, or theoretically prove that these techniques do not cycle. If the first case is shown to

be true, then generating novel pivoting rules to avoid cycling with double pivots is another potential future research topic.

The development of a triple or higher dimensional pivot simplex method is another promising research idea. In this case, research must create a fast technique to find an optimal basis to linear programs with three or more decision variables. Future studies should also determine whether or not increasing the size of the multidimensional search subproblems is beneficial. If so, research could investigate an n dimensional pivot simplex method.

Future work could also examine how the double pivot simplex method or the double pivot dual simplex method can benefit from parallel computing. Since both of these techniques require more computational effort per iteration than their corresponding single pivot simplex method, can a portion of this step be performed in parallel? Along the same lines, if one answers the aforementioned triple or more pivot problem, a parallel implementation should become more effective.

Expanding the applicability of both the slope algorithm and the ratio algorithm can also benefit the field of multidimensional searches. For instance, the slope and ratio algorithms could be implemented to find an optimal basis to the subproblems of decomposition, column generation, and block pivot techniques. Furthermore, the ratio algorithm could be implemented within the network simplex method to solve network flow problems with exactly two linear side constraints. In this case, research should investigate conditions to merge both techniques and still maintain the basis structure of the network.

6.2.2 Future Interior Point Framework Research

An obvious research question is whether or not the two dimensional search primal and dual affine scaling and logarithmic barrier interior point algorithms presented in this dissertation theoretically converge to an optimal solution. One potential research idea is to develop a nontrivial relationship between the two search directions and the feasible region of the linear program. The goal is to determine the dual estimates for the primal algorithms and

the primal estimates for the dual techniques. Thus, one could prove convergence using complementary slackness.

This dissertation presents two principles for determining search directions. The two dimensional search interior point algorithms created in this dissertation's research partially achieve the second principle by orthogonally partitioning the objective function and right-hand side vectors. Future research could investigate another technique to better accomplish the second principle. In this case, one could find a method to generate two search directions that are orthogonal to each other in the projected space.

Another promising research topic is to create the first primal-dual two dimensional search interior point algorithm. This dissertation, along with other previous research, demonstrate the potential of multidimensional searches to more quickly solve linear programming problems. Since primal-dual interior point methods are frequently used to solve large scale linear programs rather than the primal and dual algorithms, expanding the knowledge of multidimensional searches to create novel and effective primal-dual techniques would be extremely useful. To further complement existing methods, these newly developed primal-dual algorithms could also incorporate predictor-corrector steps and initial infeasible interior solutions.

Similar to a simplex framework, future research could also investigate three or higher dimensional search interior point algorithms. Furthermore, developing two dimensional search methods to solve nonlinear programming problems is another possible future study. Both of these research topics would enhance the applicability of multidimensional searches, and could improve current state-of-the-art nonlinear optimization methods.

In summary, this dissertation sets the stage for exciting future research topics in multidimensional search algorithms for linear programming and other classes of optimization problems. Only time will be able to answer whether or not these multidimensional search methods will create a completely different perspective of optimization algorithms and change the current paradigm. The author believes that these algorithms have the potential to substantially advance operations research solution techniques and will contribute to a more efficient society.

Bibliography

- I. Adler, M. G. C. Resende, G. Veiga, and N. Karmarkar. An implementation of Karmarkar's algorithm for linear programming. *Mathematical Programming*, 44(1-3):297–335, 1989. doi: 10.1007/BF01587095.
- R. Agarwala and D. Fernández-Baca. Weighted multidimensional search and its application to convex optimization. *SIAM Journal on Computing*, 25(1):83–99, 1996. doi: 10.1137/S0097539792241928.
- K. Ahmadi, F. Hasani, and B. Kheirfam. A full-Newton step infeasible interior-point algorithm based on Darvay directions for linear optimization. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 13(2):191–208, 2014. doi: 10.1007/s10852-013-9227-7.
- R. Alterovitz, E. Lessard, J. Pouliot, I. J. Hsu, J. F. O'Brien, and K. Goldberg. Optimization of HDR brachytherapy dose distributions using linear programming with penalty costs. *Medical Physics*, 33(11):4012–4019, 2006. doi: 10.1118/1.2349685.
- K. M. Anstreicher. On long step path following and SUMT for linear and quadratic programming. *SIAM Journal on Optimization*, 6(1):33–46, 1996. doi: 10.1137/0806003.
- K. M. Anstreicher and R. A. Bosch. Long steps in an $O(n^3L)$ algorithm for linear programming. *Mathematical Programming*, 54(1-3):251–265, 1992. doi: 10.1007/BF01586053.
- L. H. Appelgren. A column generation algorithm for a ship scheduling problem. *Transportation Science*, 3(1):53–68, 1969. doi: 10.1287/trsc.3.1.53.
- A. Asadi and C. Roos. Infeasible interior-point methods for linear optimization based on large neighborhood. *Journal of Optimization Theory and Applications*, 170(2):562–590, 2016. doi: 10.1007/s10957-015-0826-5.

- Y. Q. Bai, M. El Ghami, and C. Roos. A new efficient large-update primal-dual interior-point method based on a finite barrier. *SIAM Journal on Optimization*, 13(3):766–782, 2003. doi: 10.1137/S1052623401398132.
- Y. Q. Bai, M. El Ghami, and C. Roos. A comparative study of kernel functions for primal-dual interior-point algorithms in linear optimization. *SIAM Journal on Optimization*, 15(1):101–128, 2004. doi: 10.1137/S1052623403423114.
- Y. Q. Bai, G. Lesaja, C. Roos, G. Q. Wang, and M. El Ghami. A class of large-update and small-update primal-dual interior-point algorithms for linear optimization. *Journal of Optimization Theory and Applications*, 138(3):341–359, 2008. doi: 10.1007/s10957-008-9389-z.
- J. P. Bailey, T. Easton, and F. Vitor. Octanary polyhedral branch and bound for integer programs. Manuscript submitted for publication, 2018.
- E. R. Barnes. A variation on Karmarkars algorithm for solving linear programming problems. *Mathematical Programming*, 36(2):174–182, 1986. doi: 10.1007/BF02592024.
- R. H. Bartels. A stabilization of the simplex method. *Numerische Mathematik*, 16(5):414–434, 1971. doi: 10.1007/BF02169151.
- R. H. Bartels and G. H. Golub. The simplex method of linear programming using LU decomposition. *Communications of the ACM*, 12(5):266–268, 1969. doi: 10.1145/362946.362974.
- F. Bartolini, G. M. Bazzani, V. Gallerani, M. Raggi, and D. Viaggi. The impact of water and agriculture policy scenarios on irrigated farming systems in Italy: An analysis based on farm level multi-attribute linear programming models. *Agricultural Systems*, 93(1-3): 90–114, 2007. doi: 10.1016/j.agsy.2006.04.006.
- V. Baryamureeba and T. Steihaug. On the convergence of an inexact primal-dual interior point method for linear programming. In I. Lirkov, S. Margenov, and J. Waśniewski,

- editors, *International Conference on Large-Scale Scientific Computing*, volume LSSC 2005 of *Lecture Notes in Computer Science*, pages 629–637. Springer, 2006. doi: 10.1007/11666806_72.
- D. A. Bayer and J. C. Lagarias. The nonlinear geometry of linear programming. I affine and projective scaling trajectories. *Transactions of the American Mathematical Society*, 314(2):499–526, 1989a. doi: 10.2307/2001396.
- D. A. Bayer and J. C. Lagarias. The nonlinear geometry of linear programming. II legendre transform coordinates and central trajectories. *Transactions of the American Mathematical Society*, 314(2):527–581, 1989b. doi: 10.2307/2001397.
- M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, Hoboken NJ, USA, 4th edition, 2010.
- J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962. doi: 10.1007/BF01386316.
- M. Benichou, J. M. Gauthier, G. Hentges, and G. Ribiere. The efficient solution of large-scale linear programming problems—some algorithmic techniques and computational results. *Mathematical Programming*, 13(1):280–322, 1977. doi: 10.1007/BF01584344.
- H. Y. Benson and D. F. Shanno. An exact primal-dual penalty method approach to warm-starting interior-point methods for linear programming. *Computational Optimization and Applications*, 38(3):371–399, 2007. doi: 10.1007/s10589-007-9048-6.
- D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont MA, USA, 1997.
- R. E. Bixby. Commentary—progress in linear programming. *ORSA Journal on Computing*, 6(1):15–22, 1994. doi: 10.1287/ijoc.6.1.15.
- R. E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50(1):3–15, 2002. doi: 10.1287/opre.50.1.3.17780.

- R. E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, Extra Volume: Optimization Stories:107–121, 2012.
- R. G. Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2):103–107, 1977. doi: 10.1287/moor.2.2.103.
- P. T. Boggs, P. D. Domich, J. R. Donaldson, and C. Witzgall. Algorithmic enhancements to the method of centers for linear programming problems. *ORSA Journal on Computing*, 1(3):159–171, 1989. doi: 10.1287/ijoc.1.3.159.
- M. Bouafia, D. Benterki, and A. Yassine. An efficient primal-dual interior point method for linear programming problems based on a new kernel function with a trigonometric barrier term. *Journal of Optimization Theory and Applications*, 170(2):528–545, 2016. doi: 10.1007/s10957-016-0895-0.
- C. W. Carroll. The created response surface technique for optimizing nonlinear, restrained systems. *Operations Research*, 9(2):169–185, 1961. doi: 10.1287/opre.9.2.169.
- K. C. Chalermkraivuth, S. Bollapragada, M. C. Clark, J. Deaton, L. Kiaer, J. P. Murdzek, W. Neeves, B. J. Scholz, and D. Toledano. GE asset management, Genworth financial, and GE insurance use a sequential-linear-programming algorithm to optimize portfolios. *INFORMS Journal on Applied Analytics*, 35(5):370–380, 2005. doi: 10.1287/inte.1050.0164.
- A. Charnes. Optimality and degeneracy in linear programming. *Econometrica*, 20(1):160–170, 1952. doi: 10.2307/1907845.
- P. C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998. doi: 10.1023/A:1009642405419.
- J. Clausen. A note on the Edmonds-Fukuda pivoting rule for simplex algorithms. *European Journal of Operational Research*, 29(3):378–383, 1987. doi: 10.1016/0377-2217(87)90251-7.

- M. Colombo and J. Gondzio. Further development of multiple centrality correctors for interior point methods. *Computational Optimization and Applications*, 41(3):277–305, 2008. doi: 10.1007/s10589-007-9106-0.
- A. J. Conejo, E. Castillo, R. Mínguez, and R. García-Bertrand. *Decomposition Techniques in Mathematical Programming*. Engineering and Science Applications. Springer-Verlag, Berlin Heidelberg, Germany, 2006. doi: 10.1007/3-540-27686-6.
- Z. Csizmadia, T. Illés, and A. Nagy. The s-monotone index selection rules for pivot algorithms of linear programming. *European Journal of Operational Research*, 221(3):491–500, 2012. doi: 10.1016/j.ejor.2012.02.008.
- G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, 1951, pages 339–347. John Wiley & Sons, New York NY, USA, 1947.
- G. B. Dantzig. Reminiscences about the origins of linear programming. *Operations Research Letters*, 1(2):43–48, 1982. doi: 10.1016/0167-6377(82)90043-8.
- G. B. Dantzig. Linear programming. In J. K. Lenstra, A. H. G. Rinnooy Kan, and A. Schrijver, editors, *History of Mathematical Programming: A Collection of Personal Reminiscences*. Elsevier Science Publishers, Amsterdam, Netherlands, 1991. doi: 10.1287/opre.50.1.42.17798.
- G. B. Dantzig and W. Orchard-Hays. The product form for the inverse in the simplex method. *Mathematical Tables and Other Aids to Computation*, 8(46):64–67, 1954. doi: 10.2307/2001993.
- G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi: 10.1287/opre.8.1.101.
- G. B. Dantzig and P. Wolfe. The decomposition algorithm for linear programs. *Econometrica*, 29(4):767–778, 1961. doi: 10.2307/1911818.

- K. Darby-Dowman and J. M. Wilson. Developments in linear and integer programming. *Journal of the Operational Research Society*, 53(10):1065–1071, 2002. doi: 10.1057/palgrave.jors.2601435.
- Z. Darvay and P. R. Takács. New method for determining search directions for interior-point algorithms in linear optimization. *Optimization Letters*, 12(5):1099–1116, 2018. doi: 10.1007/s11590-017-1171-4.
- C. de la Vallée Poussin. Sur la méthode de l'approximation minimum. *Annales de Société Scientifique de Bruxelles*, 35:1–16, 1911.
- A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002. doi: 10.1023/A:1012470815092.
- I. I. Dikin. Iterative solution of problems of linear and quadratic programming. *Doklady Akademii Nauk SSSR*, 174(4):747–748, 1967.
- I. I. Dikin. On the convergence of an iterative process. *Upravlyaemye Sistemy*, 12(1):54–60, 1974.
- P. D. Domich, P. T. Boggs, J. E. Rogers, and C. Witzgall. Optimizing over three-dimensional subspaces in an interior-point method for linear programming. *Linear Algebra and Its Applications*, 152(1):315–342, 1991. doi: 10.1016/0024-3795(91)90280-A.
- J. Dongarra and F. Sullivan. Guest editors' introduction: The top 10 algorithms. *Computing in Science & Engineering*, 2(1):22–23, 2000. doi: 10.1109/MCISE.2000.814652.
- R. Dorfman. The discovery of linear programming. *Annals of the History of Computing*, 6(3):283–295, 1984. doi: 10.1109/MAHC.1984.10026.
- M. E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM Journal on Computing*, 13(1):31–45, 1984. doi: 10.1137/0213003.
- M. E. Dyer. On a multidimensional search technique and its application to the euclidean one-centre problem. *SIAM Journal on Computing*, 15(3):725–738, 1986. doi: 10.1137/0215052.

- M. E. Dyer and A. M. Frieze. A randomized algorithm for fixed-dimensional linear programming. *Mathematical Programming*, 44(1-3):203–212, 1989. doi: 10.1007/BF01587088.
- M. E. Dyer and S. Sen. Fast and optimal parallel multidimensional search in PRAMs with applications to linear programming and related problems. *SIAM Journal on Computing*, 30(5):1443–1461, 2000. doi: 10.1137/S0097539797325727.
- T. Easton. *The When Diet: Mathematically Optimizing Eating and Exercise for Weight Loss*. Ithaca Press, Dryden NY, USA, 2009.
- K. Edlund, L. E. Sokoler, and J. B. Jorgensen. A primal-dual interior-point linear programming algorithm for MPC. In J. Baillieul and L. Guo, editors, *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 351–356, Shanghai, China, 2009. doi: 10.1109/CDC.2009.5400440.
- M. El Ghami and C. Roos. Generic primal-dual interior point methods based on a new kernel function. *International Journal RAIRO-Operations Research*, 42(2):199–213, 2008. doi: 10.1051/ro:2008009.
- M. El Ghami, I. D. Ivanov, C. Roos, and T. Steihaug. A polynomial-time algorithm for LO based on generalized logarithmic barrier functions. *International Journal of Applied Mathematics*, 21(1):99–115, 2008.
- M. El Ghami, I. Ivanov, J. B. M. Melissen, C. Roos, and T. Steihaug. A polynomial-time algorithm for linear optimization based on a new class of kernel functions. *Journal of Computational and Applied Mathematics*, 224(2):2009, 2009. doi: 10.1016/j.cam.2008.05.027.
- M. El Ghami, Z. A. Guennoun, S. Bouali, and T. Steihaug. Interior-point methods for linear optimization based on a kernel function with a trigonometric barrier term. *Journal of Computational and Applied Mathematics*, 236(15):3613–3623, 2012. doi: 10.1016/j.cam.2011.05.036.

- J. M. Elble and N. V. Sahinidis. A review of LU factorisation in the simplex algorithm. *International Journal of Mathematics in Operational Research*, 4(4):319–365, 2012a. doi: 10.1504/IJMOR.2012.048900.
- J. M. Elble and N. V. Sahinidis. A review of the LU update in the simplex algorithm. *International Journal of Mathematics in Operational Research*, 4(4):366–399, 2012b. doi: 10.1504/IJMOR.2012.048901.
- S. K. Eldersveld and M. A. Saunders. A block-LU update for large-scale linear programming. *SIAM Journal on Matrix Analysis and Applications*, 13(1):191–201, 1992. doi: 10.1137/0613016.
- I. Elhallaoui, D. Villeneuve, F. Soumis, and G. Desaulniers. Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research*, 53(4):632–645, 2005. doi: 10.1287/opre.1050.0222.
- I. Elhallaoui, G. Desaulniers, A. Metrane, and F. Soumis. Bi-dynamic constraint aggregation and subproblem reduction. *Computers & Operations Research*, 35(5):1713–1724, 2008. doi: 10.1016/j.cor.2006.10.007.
- I. Elhallaoui, A. Metrane, G. Desaulniers, and F. Soumis. An improved primal simplex algorithm for degenerate linear programs. *INFORMS Journal on Computing*, 23(4):569–577, 2010a. doi: 10.1287/ijoc.1100.0425.
- I. Elhallaoui, A. Metrane, F. Soumis, and G. Desaulniers. Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123(2):345–370, 2010b. doi: 10.1007/s10107-008-0254-5.
- J. B. E. Etoa. New optimal pivot rule for the simplex algorithm. *Advances in Pure Mathematics*, 6(1):647–658, 2016. doi: 10.4236/apm.2016.610054.
- S. Fang and S. Puthenpura. *Linear Optimization and Extensions: Theory and Algorithms*. Prentice Hall, Englewood Cliffs NJ, USA, 1993.

- R. W. Farebrother. A linear programming procedure based on de la Vallée Poussin's minimax estimation procedure. *Computational Statistics & Data Analysis*, 51(2):453–456, 2006. doi: 10.1016/j.csda.2005.10.005.
- A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, New York NY, USA, 1968.
- R. Fletcher and S. P. J. Matthews. Stable modification of explicit LU factors for simplex updates. *Mathematical Programming*, 30(3):267–284, 1984. doi: 10.1007/BF02591933.
- L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101, 1958. doi: 10.1287/mnsc.1040.0269.
- J. J. Forrest and D. Goldfarb. Steepest-edge simplex algorithms for linear programming. *Mathematical Programming*, 57(1-3):341–374, 1992. doi: 10.1007/BF01581089.
- J. J. H. Forrest and J. A. Tomlin. Updated triangular factors of the basis to maintain sparsity in the product form simplex method. *Mathematical Programming*, 2(1):263–278, 1972. doi: 10.1007/BF01584548.
- J. Fourier. Solution d'une question particulière du calcul des inégalités. *Nouveau Bulletin des Sciences par la Société philomathique de Paris*, pages 99–100, 1826.
- R. M. Freund. An infeasible-start algorithm for linear programming whose complexity depends on the distance from the starting point to the optimal solution. *Annals of Operations Research*, 62(1):29–57, 1996. doi: 10.1007/BF02206810.
- R. Frisch. The logarithmic potential method for solving linear programming problems. Memorandum, Institute of Economics, Oslo, Norway, May 1955.
- K. Fukuda. *Oriented Matroid Programming*. Ph.D. dissertation, Waterloo University, Waterloo, ON, Canada, 1982.

- J. García, J. E. Florez, Á. Torralba, D. Borrajo, C. L. López, Á. García-Olaya, and J. Sáenz. Combining linear programming and automated planning to solve intermodal transportation problems. *European Journal of Operational Research*, 227(1):216–226, 2013. doi: 10.1016/j.ejor.2012.12.018.
- S. I. Gass and S. Vinjamuri. Cycling in linear programming problems. *Computers & Operations Research*, 31(2):303–311, 2004. doi: 10.1016/S0305-0548(02)00226-5.
- J. B. Gauthier, J. Desrosiers, and M. E. Lübbecke. Vector space decomposition for solving large-scale linear programs. *Operations Research*, 66(5):1376–1389, 2018. doi: 10.1287/opre.2018.1728.
- A. Gautier, B. F. Lamond, D. Paré, and F. Rouleau. The Québec Ministry of natural resources uses linear programming to understand the wood-fiber market. *INFORMS Journal on Applied Analytics*, 30(6):32–48, 2000. doi: 10.1287/inte.30.6.32.11625.
- D. M. Gay. Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newsletter*, 13:10–12, 1985.
- D. M. Gay. A variant of Karmarkar’s linear programming algorithm for problems in standard form. *Mathematical Programming*, 37(1):81–90, 1987. doi: 10.1007/BF02591685.
- P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Sparse matrix methods in optimization. *SIAM Journal on Scientific and Statistical Computing*, 5(3):562–589, 1984. doi: 10.1137/0905041.
- P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright. On the projected Newton barrier methods for linear programming and an equivalence to Karmarkar’s projective method. *Mathematical Programming*, 36(2):183–209, 1986. doi: 10.1007/BF02592025.
- P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961. doi: 10.1287/opre.9.6.849.

- P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem-part II. *Operations Research*, 11(6):863–888, 1963. doi: 10.1287/opre.11.6.863.
- D. Goldfarb and J. K. Reid. A practicable steepest-edge simplex algorithm. *Mathematical Programming*, 12(1):361–371, 1977. doi: 10.1007/BF01593804.
- D. Goldfarb and M. J. Todd. Chapter II linear programming. In G. L. Nemhauser, A. H. G. Rinooy Kan, and M. J. Todd, editors, *Handbooks in Operations Research and Management Science*, volume 1, pages 73–170. Elsevier, Amsterdam, Netherlands, 1989. doi: 10.1016/S0927-0507(89)01003-0.
- A. M. Gomes and J. F. Oliveira. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, 171(3):811–829, 2006. doi: 10.1016/j.ejor.2004.09.008.
- J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6(2):137–156, 1996. doi: 10.1007/BF00249643.
- J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012. doi: 10.1016/j.ejor.2011.09.017.
- C. C. Gonzaga. Conical projection algorithms for linear programming. *Mathematical Programming*, 43(1-3):151–173, 1989a. doi: 10.1007/BF01582287.
- C. C. Gonzaga. An algorithm for solving linear programming problems in $O(n^3L)$ operations. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 1–28. Springer-Verlag, New York NY, USA, 1989b. doi: 10.1007/978-1-4613-9617-8_1.
- C. C. Gonzaga. Search directions for interior linear-programming methods. *Algorithmica*, 6(1-6):153–181, 1991a. doi: 10.1007/BF01759039.

- C. C. Gonzaga. Large step path-following methods for linear programming, part I: Barrier function method. *SIAM Journal on Optimization*, 1(2):268–279, 1991b. doi: 10.1137/0801018.
- C. C. Gonzaga. Large step path-following methods for linear programming, part II: Potential reduction method. *SIAM Journal on Optimization*, 1(2):280–292, 1991c. doi: 10.1137/0801019.
- C. C. Gonzaga. Path-following methods for linear programming. *SIAM Review*, 34(2):167–224, 1992. doi: 10.1137/1034048.
- I. Grattan-Guinness. Joseph Fourier’s anticipation of linear programming. *Operational Research Quarterly (1970-1977)*, 21(3):361–364, 1970. doi: 10.2307/3008492.
- H. J. Greenberg. Pivot selection tactics. In H. J. Greenberg, editor, *Design and Implementation of Optimization Software*, volume 28 of *Nato Science Series E*, pages 41–102. Springer, Alphen aan den Rijn, Netherlands, 1978.
- H. J. Greenberg. An analysis of degeneracy. *Naval Research Logistics*, 33(1):635–655, 1986. doi: 10.1002/nav.3800330409.
- G. Gu, H. Mansouri, M. Zangiabadi, Y. Q. Bai, and C. Roos. Improved full-Newton step $O(nL)$ infeasible interior-point method for linear optimization. *Journal of Optimization Theory and Applications*, 145(2):271–288, 2010. doi: 10.1007/s10957-009-9634-0.
- O. Güler and Y. Ye. Convergence behavior of interior-point algorithms. *Mathematical Programming*, 60(1-3):215–228, 1993. doi: 10.1007/BF01580610.
- J. Haeberly, M. V. Nayakkankuppam, and M. L. Overton. Extending Mehrotra and Gondzio higher order methods to mixed semidefinite-quadratic-linear programming. *Optimization Methods and Software*, 11(1-4):67–90, 1999. doi: 10.1080/10556789908805748.

- W. W. Hager and H. Zhang. An affine scaling method for optimization problems with polyhedral constraints. *Computational Optimization and Applications*, 59(1-2):163–183, 2014. doi: 10.1007/s10589-013-9535-x.
- P. M. J. Harris. Pivot selection methods of the Devex LP code. *Mathematical Programming*, 5(1):1–28, 1973. doi: 10.1007/BF01580108.
- F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, New York NY, USA, 10th edition, 2015.
- R. Howard. *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge MA, USA, 1960.
- J. Hu, J. E. Mitchell, J. Pang, and B. Yu. On linear programs with linear complementarity constraints. *Journal of Global Optimization*, 53(1):29–51, 2012. doi: 10.1007/s10898-010-9644-3.
- Q. Huangfu and J. A. J. Hall. Novel update techniques for the revised simplex method. *Computational Optimization and Applications*, 60(3):587–608, 2015. ISSN 1573-2894. doi: 10.1007/s10589-014-9689-1.
- Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. doi: 10.1007/s12532-017-0130-5.
- P. Huard. Resolution of mathematical programming with nonlinear constraints by the method of centers. In J. Abadie, editor, *Nonlinear Programming*, pages 207–219. North-Holland, Amsterdam, Netherlands, 1967.
- IBM. CPLEX Optimizer. Version 12.7, 2016. URL <https://www.ibm.com/analytics/cplex-optimizer>.
- T. Illés and T. Terlaky. Pivot versus interior point methods: Pros and cons. *European Journal of Operational Research*, 140(2):170–190, 2002. doi: 10.1016/S0377-2217(02)00061-9.

- B. Jansen, C. Roos, T. Terlaky, and J. P. Vial. Primal-dual algorithms for linear programming based on the logarithmic barrier method. *Journal of Optimization Theory and Applications*, 83(1):1–26, 1994. doi: 10.1007/BF02191759.
- D. L. Jensen. *Coloring and Duality: Combinatorial Augmentation Methods*. Ph.D. dissertation, Cornell University, Ithaca, NY, USA, 1985.
- M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer-Verlag, Berlin Heidelberg, Germany, 1st edition, 2010. doi: 10.1007/978-3-540-68279-0.
- L. V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6(4):366–422, 1939. doi: 10.1287/mnsc.6.4.366.
- N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984. doi: 10.1007/BF02579150.
- N. Karmarkar and R. Ramakrishnan. Further developments in the new polynomial-time algorithm for linear programming. In *ORSA/TIMS Joint National Meeting*, Boston MA, USA, 1985.
- L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. doi: 10.1016/0041-5553(80)90061-0.
- V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities-III: Proceedings of the Third Symposium on Inequalities*, pages 159–175. Academic Press, New York NY, USA, 1972.
- A. Koberstein and U. H. Suhl. Progress in the dual simplex method for large scale LP problems: Practical dual phase 1 algorithms. *Computational Optimization and Applications*, 37(1):49–65, 2007. doi: 10.1007/s10589-007-9022-3.

- T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. MIPLIB 2010: Mixed integer programming library version 5. *Mathematical Programming Computation*, 3(2):103–163, 2011. doi: 10.1007/s12532-011-0025-9.
- M. Kojima, S. Mizuno, and A. Yoshise. A primal-dual interior point algorithm for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 29–47. Springer-Verlag, New York NY, USA, 1989. doi: 10.1007/978-1-4613-9617-8_2.
- M. Kojima, N. Megiddo, and S. Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Mathematical Programming*, 61(1-3):263–280, 1993. doi: 10.1007/BF01582151.
- N. Komodakis and J. Pesquet. Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *IEEE Signal Processing Magazine*, 32(6):31–54, 2015. doi: 10.1109/MSP.2014.2377273.
- T. C. Koopmans. Optimum utilization of the transportation system. *Econometrica*, 17 (Supplement: Report of the Washington Meeting):136–146, 1949. doi: 10.2307/1907301.
- S. Kunnumkal, K. Talluri, and H. Topaloglu. A randomized linear programming method for network revenue management with product-specific no-shows. *Transportation Science*, 46 (1):90–108, 2012. doi: 10.1287/trsc.1110.0386.
- A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. doi: 10.2307/1910129.
- K. D. Lawrence and J. J. Burbridge. A multiple goal linear programming model for coordinated production and logistics planning. *International Journal of Production Research*, 14(2):215–222, 2007. doi: 10.1080/00207547608956595.

- E. K. Lee, R. J. Gallagher, and D. A. Patterson. A linear programming approach to discriminant analysis with a reserved-judgment region. *INFORMS Journal on Computing*, 15(1):23–41, 2003. doi: 10.1287/ijoc.15.1.23.15158.
- C. E. Lemke. The dual method of solving the linear programming problem. *Naval Research Logistics*, 1(1):36–47, 1954. doi: 10.1002/nav.3800010107.
- W. W. Leontief. Quantitative input and output relations in the economic systems of the United States. *The Review of Economics and Statistics*, 18(3):105–125, 1936. doi: 10.2307/1927837.
- L. Liao. A study of the dual affine scaling continuous trajectories for linear programming. *Journal of Optimization Theory and Applications*, 163(2):548–568, 2014. doi: 10.1007/s10957-013-0495-1.
- Y. Liao. The improvement on R. G. Blands method. In E. Qi, J. Shen, and R. Dou, editors, *The 19th International Conference on Industrial Engineering and Engineering Management*, pages 799–803, Changsha, China, 2013. doi: 10.1007/978-3-642-38391-5_84.
- S. Lim and S. Park. A new admissible pivot method for linear programming. *Asia-Pacific Journal of Operational Research*, 21(4):421–434, 2004. doi: 10.1142/S0217595904000394.
- C. Liu, H. Liu, and W. Cong. An $O(\sqrt{n}L)$ iteration primal-dual second-order corrector algorithm for linear programming. *Optimization Letters*, 5(4):729–743, 2011. doi: 10.1007/s11590-010-0242-6.
- M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234.
- I. J. Lustig. Feasibility issues in a primal-dual interior-point method for linear programming. *Mathematical Programming*, 49(1-3):145–162, 1990. doi: 10.1007/BF01588785.

- I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra and its Applications*, 152(1):191–222, 1991. doi: 10.1016/0024-3795(91)90275-2.
- I. J. Lustig, R. E. Marsten, and D. F. Shanno. On implementing Mehrotra’s predictor-corrector interior-point method for linear programming. *SIAM Journal on Optimization*, 2(3):435–449, 1992. doi: 10.1137/0802022.
- I. J. Lustig, R. E. Marsten, and D. F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6(1):1–14, 1994a. doi: 10.1287/ijoc.6.1.1.
- I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a globally convergent primal-dual predictor-corrector algorithm for linear programming. *Mathematical Programming*, 66(1-3):123–135, 1994b. doi: 10.1007/BF01581140.
- X. Ma and H. Liu. A superlinearly convergent wide-neighborhood predictor-corrector interior-point algorithm for linear programming. *Journal of Applied Mathematics and Computing*, 55(1-2):669–682, 2017. doi: 10.1007/s12190-016-1055-2.
- R. Mansini, W. Ogryczak, and M. G. Speranza. Conditional value at risk and related linear programming models for portfolio optimization. *Annals of Operations Research*, 152(1):227–256, 2007. doi: 10.1007/s10479-006-0142-4.
- H. Mansouri and M. Zangiabadi. An adaptive infeasible interior-point algorithm with full-Newton step for linear optimization. *Optimization*, 62(2):285–297, 2013. doi: 10.1080/02331934.2011.611881.
- H. Mansouri, M. Zangiabadi, and M. Arzani. A modified infeasible-interior-point algorithm for linear optimization problems. *Journal of Optimization Theory and Applications*, 166(2):605–618, 2015. doi: 10.1007/s10957-015-0719-7.

- H. Marchand, A. Martin, R. Weismantel, and L. Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1-3):397–446, 2002. doi: 10.1016/S0166-218X(01)00348-1.
- I. Maros. A general pricing scheme for the simplex method. *Annals of Operations Research*, 124(1-4):193–203, 2003a. doi: 10.1023/B:ANOR.0000004769.36807.cf.
- I. Maros. *Computational Techniques of the Simplex Method*, volume 61 of *International Series in Operations Research & Management Science*. Springer, New York NY, USA, 1st edition, 2003b. doi: 10.1007/978-1-4615-0257-9.
- K. A. McShane, C. L. Monma, and D. Shanno. An implementation of a primal-dual interior point method for linear programming. *ORSA Journal on Computing*, 1(2):70–83, 1989. doi: 10.1287/ijoc.1.2.70.
- N. Megiddo. Linear-time algorithms for linear programming in \mathbb{R}^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983. doi: 10.1137/0212052.
- N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, 31(1):114–127, 1984. doi: 10.1145/2422.322418.
- N. Megiddo. A note on degeneracy in linear programming. *Mathematical Programming*, 35(3):365–367, 1986. doi: 10.1007/BF01580886.
- N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 131–158. Springer-Verlag, New York NY, USA, 1989. doi: 10.1007/978-1-4613-9617-8_8.
- S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992. doi: 10.1137/0802028.
- R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms. part I: Linear programming. *Mathematical Programming*, 44(1-3):27–41, 1989. doi: 10.1007/BF01587075.

- R. D. C. Monteiro, T. Tsuchiya, and Y. Wang. A simplified global convergence proof of the affine scaling algorithm. *Annals of Operations Research*, 46(2):443–482, 1993. doi: 10.1007/BF02023109.
- S. Nadarajah, F. Margot, and N. Secomandi. Relaxations of approximate linear programs for the real option management of commodity storage. *Management Science*, 61(12):3054–3076, 2015. doi: 10.1287/mnsc.2014.2136.
- R. K. Nayak, M. P. Biswal, and S. Padhy. An affine scaling method for solving network flow problems. *Journal of Discrete Mathematical Sciences and Cryptography*, 15(1):13–29, 2012. doi: 10.1080/09720529.2012.10698361.
- G. L. Nemhauser. Column generation for linear and integer programming. *Documenta Mathematica*, Extra Volume: Optimization Stories:65–73, 2012.
- Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied and Numerical Mathematics. SIAM, Philadelphia PA, USA, 1994. doi: 10.1137/1.9781611970791.
- W. Orchard-Hays. *Advanced Linear Programming Computing Techniques*. McGraw-Hill, New York NY, USA, 1968.
- W. Orchard-Hays. History of mathematical programming systems. In H. J. Greenberg, editor, *Design and Implementation of Optimization Software*, volume 28 of *Nato Science Series E*, pages 1–26. Springer, Alphen aan den Rijn, Netherlands, 1978a.
- W. Orchard-Hays. Scope of mathematical programming software. In H. J. Greenberg, editor, *Design and Implementation of Optimization Software*, volume 28 of *Nato Science Series E*, pages 27–40. Springer, Alphen aan den Rijn, Netherlands, 1978b.
- W. Orchard-Hays. Anatomy of mathematical programming system. In H. J. Greenberg, editor, *Design and Implementation of Optimization Software*, volume 28 of *Nato Science Series E*, pages 41–102. Springer, Alphen aan den Rijn, Netherlands, 1978c.

- W. Orchard-Hays. History of mathematical programming systems. *Annals of the History of Computing*, 6(3):296–312, 1984. doi: 10.1109/MAHC.1984.10032.
- W. Orchard-Hays. History of the development of LP solvers. *INFORMS Journal on Applied Analytics*, 20(4):61–73, 1990. doi: 10.1287/inte.20.4.61.
- M. Padberg. *Linear Optimization and Extensions*, volume 12 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin Heidelberg, Germany, 2nd edition, 1999. doi: 10.1007/978-3-662-12273-0.
- F. Palacios-Gomez, L. Lasdon, and M. Engquist. Nonlinear optimization by successive linear programming. *Management Science*, 28(10):264–274, 1982. doi: 10.1287/mnsc.28.10.1106.
- P. Pan. A basis-deficiency-allowing variation of the simplex method for linear programming. *Computers & Mathematics with Applications*, 36(3):33–53, 1998. doi: 10.1016/S0898-1221(98)00127-8.
- P. Pan. A new perturbation simplex algorithm for linear programming. *Journal of Computational Mathematics*, 17(3):233–242, 1999.
- P. Pan. Primal perturbation simplex algorithms for linear programming. *Journal of Computational Mathematics*, 18(6):587–596, 2000.
- P. Pan. A largest-distance pivot rule for the simplex algorithm. *European Journal of Operational Research*, 187(2):393–402, 2008a. doi: 10.1016/j.ejor.2007.03.026.
- P. Pan. Efficient nested pricing in the simplex algorithm. *Operations Research Letters*, 36(3):309–313, 2008b. doi: 10.1016/j.orl.2007.10.001.
- P. Pan. A fast simplex algorithm for linear programming. *Journal of Computational Mathematics*, 28(6):837–847, 2010.
- P. Pan. An affine-scaling pivot algorithm for linear programming. *Optimization*, 62(4):431–445, 2013. doi: 10.1080/02331934.2011.606576.

- P. Pan. *Linear Programming Computation*. Springer-Verlag, Berlin Heidelberg, Germany, 1st edition, 2014. doi: 10.1007/978-3-642-40754-3.
- N. Ploshkas and N. Samaras. *Linear Programming Using MATLAB®*, volume 127 of *Springer Optimization and Its Applications*. Springer, Cham, Switzerland, 1st edition, 2017. doi: 10.1007/978-3-319-65919-0.
- F. A. Potra. A quadratically convergent predictor-corrector method for solving linear programs from infeasible starting points. *Mathematical Programming*, 67(1-3):383–406, 1994. doi: 10.1007/BF01582228.
- F. A. Potra. An infeasible-interior-point predictor-corrector algorithm for linear programming. *SIAM Journal on Optimization*, 6(1):19–32, 1996. doi: 10.1137/0806002.
- W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes*. Cambridge University Press, New York NY, USA, 3rd edition, 2007.
- V. Raymond, F. Soumis, and D. Orban. A new version of the improved primal simplex for degenerate linear programs. *Computers & Operations Research*, 37(1):91–98, 2010. doi: 10.1016/j.cor.2009.03.020.
- J. K. Reid. A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases. *Mathematical Programming*, 24(1):55–69, 1982. doi: 10.1007/BF01585094.
- J. Renegar. A polynomial-time algorithm, based on Newton’s method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988. doi: 10.1007/BF01580724.
- H. E. Romeijn, R. K. Ahuja, J. F. Dempsey, and A. Kumar. A new linear programming approach to radiation therapy treatment planning problems. *Operations Research*, 54(2): 201–216, 2006. doi: 10.1287/opre.1050.0261.

- A. Rong and R. Lahdelma. Fuzzy chance constrained linear programming model for optimizing the scrap charge in steel production. *European Journal of Operational Research*, 186(3):953–964, 2008. doi: 10.1016/j.ejor.2007.02.017.
- C. Roos. An improved and simplified full-Newton step $O(n)$ infeasible interior-point method for linear optimization. *SIAM Journal on Optimization*, 25(1):102–114, 2015. doi: 10.1137/140975462.
- D. M. Ryan and M. R. Osborne. On the solution of highly degenerate linear programmes. *Mathematical Programming*, 41(1-3):385–392, 1988. doi: 10.1007/BF01580776.
- R. Saigal. A simple proof of a primal affine scaling method. *Annals of Operations Research*, 62(1):303–324, 1996a. doi: 10.1007/BF02206821.
- R. Saigal. The primal power affine scaling method. *Annals of Operations Research*, 62(1):375–417, 1996b. doi: 10.1007/BF02206824.
- R. Sedgewick and K. Wayne. *Algorithms*. Addison-Wesley, Boston MA, USA, 4th edition, 2011.
- R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, 6(3):423–434, 1991. doi: 10.1007/BF02574699.
- R. Shamir. The efficiency of the simplex method: A survey. *Management Science*, 33(3):301–334, 1987. doi: 10.1287/mnsc.33.3.301.
- M. I. Shamos and D. Hoey. Geometric intersection problems. In *17th Annual Symposium on Foundations of Computer Science*, pages 208–215, Houston TX, USA, 1976. doi: 10.1109/SFCS.1976.16.
- D. Shanno. Who invented the interior-point method? *Documenta Mathematica*, Extra Volume: Optimization Stories:55–64, 2012.
- D. F. Shanno and A. Bagchi. A unified view of interior point methods for linear programming. *Annals of Operations Research*, 22(1):55–70, 1990. doi: 10.1007/BF02023048.

- D. X. Shaw and D. Goldfarb. A path-following projective interior point method for linear programming. *SIAM Journal on Optimization*, 4(1):65–85, 1994. doi: 10.1137/0804003.
- D. A. Spielman and S. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004. doi: 10.1145/990308.990310.
- J. M. Spitter, C. A. J. Hurkens, A. G. de Kok, J. K. Lenstra, and E. G. Negenman. Linear programming models with planned lead times for supply chain operations planning. *European Journal of Operational Research*, 163(3):706–720, 2005. doi: 10.1016/j.ejor.2004.01.019.
- L. M. Suhl and U. H. Suhl. A fast LU update for linear programming. *Annals of Operations Research*, 43(1):33–47, 1993. doi: 10.1007/BF02025534.
- U. H. Suhl and L. M. Suhl. Computing sparse LU factorizations for large-scale linear programming bases. *ORSA Journal on Computing*, 2(4):325–335, 1990. doi: 10.1287/ijoc.2.4.325.
- P. R. Takács and Z. Darvay. A primal-dual interior-point algorithm for symmetric optimization based on a new method for finding search directions. *Optimization*, 67(6):889–905, 2018. doi: 10.1080/02331934.2018.1432610.
- L. Tang, J. Liu, A. Rong, and Z. Yang. A mathematical programming model for scheduling steelmaking-continuous casting production. *European Journal of Operational Research*, 120(2):423–435, 2000. doi: 10.1016/S0377-2217(99)00041-7.
- T. Terlaky. A finite crisscross method for oriented matroids. *Journal of Combinatorial Theory, Series B*, 42(3):319–327, 1987. doi: 10.1016/0095-8956(87)90049-9.
- T. Terlaky and S. Zhang. Pivot rules for linear programming: A survey on recent theoretical developments. *Annals of Operations Research*, 46(1):203–233, 1993. doi: 10.1007/BF02096264.

- M. J. Todd. Linear and quadratic programming in oriented matroids. *Journal of Combinatorial Theory, Series B*, 39(2):105–133, 1985. doi: 10.1016/0095-8956(85)90042-5.
- M. J. Todd. The many facets of linear programming. *Mathematical Programming*, 91(3):417–436, 2002. doi: 10.1007/s101070100261.
- P. Tolla. A stable and sparsity exploiting LU factorization of the basis matrix in linear programming. *European Journal of Operational Research*, 24(2):247–251, 1986. doi: 10.1016/0377-2217(86)90046-9.
- J. A. Tomlin. Pivoting for size and sparsity in linear programming inversion routes. *IMA Journal of Applied Mathematics*, 10(3):289–295, 1972. doi: 10.1093/imamat/10.3.289.
- P. Tseng and Z. Luo. On the convergence of the affine-scaling algorithm. *Mathematical Programming*, 56(1-3):301–319, 1992. doi: 10.1007/BF01580904.
- T. Tsuchiya. Global convergence property of the affine scaling methods for primal degenerate linear programming problems. *Mathematics of Operations Research*, 17(3):527–557, 1992. doi: 10.1287/moor.17.3.527.
- C. van Dooren. A review of the use of linear programming to optimize diets, nutritiously, economically and environmentally. *Frontiers in Nutrition*, 5(48):1–15, 2018. doi: 10.3389/fnut.2018.00048.
- R. J. Vanderbei. *Linear Programming: Foundations and Extensions*, volume 196 of *International Series in Operations Research & Management Science*. Springer, New York NY, USA, 4th edition, 2014. doi: 10.1007/978-1-4614-7630-6.
- R. J. Vanderbei and J. C. Lagarias. I. I. Dikin’s convergence result for the affine-scaling algorithm. *Contemporary Mathematics*, 114(1):109–119, 1990.
- R. J. Vanderbei, M. S. Meketon, and B. A. Freedman. A modification of Karmarkar’s linear programming algorithm. *Algorithmica*, 1(1-4):395–407, 1986. doi: 10.1007/BF01840454.

- F. Vitor. The ratio algorithm to solve the optimal basis of two constraint linear programs. In K. Barker, D. Berry, and C. Rainwater, editors, *Proceedings of the 2018 IISE Annual Conference*, pages 1949–1954, Orlando FL, USA, 2018.
- F. Vitor and T. Easton. Merged knapsack cover inequalities for the multiple knapsack problem. In H. Yang, Z. Kong, and MD Sarder, editors, *Proceedings of the 2016 Industrial and Systems Engineering Research Conference*, pages 607–612, Anaheim CA, USA, 2016.
- F. Vitor and T. Easton. A two dimensional search primal affine scaling interior point algorithm for linear programs. In K. Barker, D. Berry, and C. Rainwater, editors, *Proceedings of the 2018 IISE Annual Conference*, pages 1961–1966, Orlando FL, USA, 2018a.
- F. Vitor and T. Easton. The double pivot simplex method. *Mathematical Methods of Operations Research*, 87(1):109–137, 2018b. doi: 10.1007/s00186-017-0610-4.
- F. Vitor and T. Easton. Projected orthogonal vectors in two dimensional search interior point algorithms for linear programming. Manuscript submitted for publication, 2019a.
- F. Vitor and T. Easton. Approximate and exact merging of knapsack constraints with cover inequalities. Manuscript submitted for publication, 2019b.
- F. T. Vitor. *Improving the Solution Time of Integer Programs by Merging Knapsack Constraints with Cover Inequalities*. M.S. thesis, Kansas State University, Manhattan, KS, USA, 2015.
- J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928. doi: 10.1007/bf01448847.
- H. P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, Chichester, England, 5th edition, 2013.
- W. L. Winston. *Operations Research: Applications and Algorithms*. Thomson Learning, Belmont CA, USA, 4th edition, 2004.

- L. B. Winternitz, S. O. Nicholls, A. L. Tits, and D. P. O’Leary. A constraint-reduced variant of Mehrotra’s predictor-corrector algorithm. *Computational Optimization and Applications*, 51(3):1001–1036, 2012. doi: 10.1007/s10589-010-9389-4.
- P. Wolfe. A technique for resolving degeneracy in linear programming. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):205–211, 1963. doi: 10.1137/0111016.
- H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, volume 27 of *International Series in Operations Research & Management Science*. Springer, New York NY, USA, 1st edition, 2000. doi: 10.1007/978-1-4615-4381-7.
- S. J. Wright. *Primal-Dual Interior-Point Methods*. Other Titles in Applied Mathematics. SIAM, Philadelphia PA, USA, 1997. doi: 10.1137/1.9781611971453.
- J. H. Wu. Long-step primal path-following algorithm for monotone variational inequality problems. *Journal of Optimization Theory and Applications*, 99(2):509–531, 1998. doi: 10.1023/A:1021786630040.
- X. Yang, Y. Zhang, and H. Liu. A wide neighborhood infeasible-interior-point method with arc-search for linear programming. *Journal of Applied Mathematics and Computing*, 51(1-2):209–225, 2016. doi: 10.1007/s12190-015-0900-z.
- Y. Yang. A polynomial arc-search interior-point algorithm for linear programming. *Journal of Optimization Theory and Applications*, 158(3):859–873, 2013. doi: 10.1007/s10957-013-0281-0.
- Y. Yang. Two computationally efficient polynomial-iteration infeasible interior-point algorithms for linear programming. *Numerical Algorithms*, 79(3):957–992, 2018. doi: 10.1007/s11075-018-0469-3.
- Y. Ye. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011. doi: 10.1287/moor.1110.0516.

- Y. Ye, M. J. Todd, and S. Mizuno. An $O(\sqrt{n}L)$ -iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19(1):53–67, 1994. doi: 10.1287/moor.19.1.53.
- Y. Zhang. On the convergence of a class of infeasible interior-point methods for the horizontal linear complementarity problem. *SIAM Journal on Optimization*, 4(1):208–227, 1994. doi: 10.1137/0804012.
- P. Zhou and B. W. Ang. Linear programming models for measuring economy-wide energy efficiency performance. *Energy Policy*, 36(8):2911–2916, 2008. doi: 10.1016/j.enpol.2008.03.041.