University of Macedonia

Doctoral Thesis

Design and development of a hybrid mathematical programming algorithm

Author:
Themistoklis Glavelis

Supervisor:
Prof. Nikolaos Samaras

A thesis submitted in fulfilment of the requirements for the degree of PhD Thesis

in the

Algorithmic Operations Research Group (algOR)
Department of Applied Informatics

July 2019

Declaration of Authorship

I, Themistoklis Glavelis, declare that this thesis titled, 'Design and development of a hybrid mathematical programming algorithm' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"The Purpose of Mathematical Programming is Insight, not Numbers."

Arthur Geoffrion (1976)

University of Macedonia

Abstract

Department of Applied Informatics

PhD Thesis

Design and development of a hybrid mathematical programming algorithm

by Themistoklis Glavelis

One of the most significant and well-studied optimization problems is the Linear Programming problem (LP). LP consists of optimizing, (minimizing or maximizing) a linear function over a certain domain. The domain is given by a set of linear constraints. The presence of effective presolve techniques is of great importance for every linear programming solver. The main goal of the presolve session is to reduce the problem's size and to determine whether the problem is unbounded or infeasible. Computational results with a set of optimal benchmark problems from NETLIB are also presented. Finally, simplex algorithm has been used in order to solve benchmarks before and after a new proposed presolve technique has been performed to LPs.

Moreover, an experimental investigation of a variation of Primal-Dual Exterior Point Simplex Algorithm (PDEPSA) is presented and it is called Primal-Dual Interior Point Simplex Algorithm (PDIPSA). In order to gain an insight into the practical behaviour of the proposed algorithm, we have performed some computational experiments. Our computational results demonstrate that PDIPSA is faster comparable with simplex algorithm and primal exterior point algorithm.

The main aim of this thesis is to present a hybrid algorithm which combines the strengths of Interior Point Methods (IPMs) and Exterior Point Simplex Algorithms (EPSAs). The hybrid algorithm takes full advantage of IPM at the first iterations which lead to significant enhancement of the objective function's value. The hybrid algorithm uses the Primal Dual Interior Point Simplex Algorithm (PDIPSA) which is a variation of EPSAs and computes a direction to the feasible region according to the interior point that was calculated by IPM. Furthermore, a computational study is presented with experiments over the Netlib set (optimal, Kennington and infeasible LPs) and the Misc section of Mészáros collection.

Supervisor: Samaras Nikolaos
Title: Professor

Acknowledgements

I owe my deepest gratitude to my supervisor Prof. Nikolaos Samaras for his encouragement, guidance, and support throughout this entire process. I am grateful to him for his endless patience and enthusiasm for the topic from our first meeting many years ago to our final editing days. His great experience and expert knowledge in the field of Linear Programming was always a great motivation and without his invaluable advice and guidance the completion of this thesis would be impossible. He is not only a valuable associate and instructor, but also a lifetime friend.

I am also grateful to Prof. Konstantinos Paparrizos who was always willing to take time out of his busy schedule to help me with his ideas whenever I needed them. I would also like to thank Prof. Andreas Georgiou for agreeing to serve on my committee, reviewing my work and his comments were really valuable.

Throughout this journey I have had the pleasure of working with excellent colleagues from all over the globe. Each and every one of them has helped in my understanding of the linear programming. In particular, I would like to point out a few people that helped me along the way and I spent much time with them on a daily basis as being member of the Algorithmic Operations Research Group (AlgOR). I would like to thank especially my colleague Nikolaos Ploskas, I know him for the last fifteen years and he is an invaluable friend and his support was significant even when the situations were no so easy for me. Moreover, I would like to express my thanks to my colleagues Charalampos Triantafyllidis, George Geranis and Dorothea Petraki for their cooperation and partnership.

Finally, and perhaps most important, this thesis would not have been possible without the unquestioning support, sacrifice, and patience of my family. Thanks in particular to my father, George, my mother Kondylio, my siblings Fillipos and Anastasios. Last but not least, I owe a special thanks to Anastasia without her endless support and understanding not only this Thesis but nothing would be completed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Contributions of this thesis

The current thesis provides contributions to the following fields:

- Despite the explosion in computational power of hardware, the presence of effective presolve techniques is of great importance for every linear programming solver. The main goal of the presolve session is to reduce the problem's size and to determine whether the problem is unbounded or infeasible. The implementation of presolve techniques can lead to significant reduction of data size in a Linear Programming problem (LP) and as a result to significant reduction of execution time. One of the goals of the current thesis is to present a new presolve technique and to perform a computational study. Computational results with a set of optimal benchmark problems from NETLIB are also presented. The computational study includes statistics of benchmarks before the application of the presolve session, statistics after the application of the presolve session and the impact of the proposed new presolve technique. Finally, simplex algorithm has been used in order to solve benchmarks before and after the new presolve technique has been performed. On the average the new presolve technique has achieved 5.47% reduction in the number of iterations and 5.99% reduction in the cpu time.

- Another aim of this thesis is to present an experimental investigation of a Primal-Dual Exterior Point Simplex Algorithm (PDEPSA) for LPs. There was a huge gap between the theoretical worst case complexity and practical performance of simplex type algorithms. PDEPSA traverses across the interior of the feasible region in an attempt to avoid combinatorial complexities of vertex following algorithms. In order to gain an insight into the practical behaviour of the proposed algorithm,

we have performed some computational experiments. Our computational results demonstrate that PDEPSA is faster than the simplex algorithm and the primal exterior point algorithm. Moreover, a clear advantage is obtained by PDEPSA in the number of iterations and the CPU execution time.

- Furthermore, the main aim of this thesis is to present a hybrid algorithm which combines the strengths of Interior Point Methods (IPMs) and Exterior Point Simplex Algorithms (EPSAs). The hybrid algorithm takes full advantage of IPM at the first iterations which lead to significant enhancement of the objective function's value. However, it has been observed that after of some iterations, IPMs do not converge fast to the optimal solution; at this stage the EPSA is applied. IPMs move inside the feasible region from one interior point to another. Our variation of EPSA demands an initial interior point, this starting point is crucial and it is computed by IPM. The hybrid algorithm uses the Primal Dual Interior Point Simplex Algorithm (PDIPSA) which is a variation of EPSA and computes a direction to the feasible region according to the interior point that was calculated by IPM. Also a computational study is presented with experiments over the Netlib set (optimal, Kennington and infeasible LPs) and the Misc section of Mészáros collection. Computational results show algorithm's superiority over the Simplex Algorithm and EPSA.

1.2 Operations Research

Operations Research (OR) is the science which is responsible for problem formulation and solutions which can conclude to improvements to the process of appropriate decision making. A common misunderstanding is that OR is a collection of mathematical tools. Although OR uses a variety of mathematical techniques and models, it is applicable to a much broader field. Since its first steps, OR has significantly contributed in many factors such as industry, managerial and decision-making procedures, which is the main activity of an engineer or a manager. The main goal of OR is to analyse and optimize the performance of real life problems.

OR is a relatively new area of research and it puts its fundamental principles in the last fifty years while its origins stems from the latter half of World War II. Despite the fact, that it started its first steps in World War II, OR met rapid expansion and it is applied to a vast number of problems during the last decades. Nowadays, OR is an independent well-defined research field and its procedures are used in a wide range of application areas.

It is well known that the origins of OR is in England during World War II when Royal Air Force attempted to implement an optimal solution for its' radar defence systems. The main goal of this project was to improve the operational efficiency of these systems. Consequently, this procedure of making a research of how to make a process to run more efficiently soon started to expand into other areas of the war. USA took the opportunity and USA Navy used the principles of OR in the antisubmarine warfare.

With the end of World War II, OR spread rapidly among the researchers who saw many common characteristics between the problems that military faced and the problems of real life in industry. Industrialization encouraged and gave a significant boost to OR to evolve its principles and methods while the allocation of limited resources to various activities arose as one of the most significant industrial problems. Furthermore, the increasing specialisation fed and made more difficult the attempt to locate the optimal assignment of industry resources in order to increase the productivity. Consequently, over the next years OR introduced and increased its role in a wide variety of issues in transportation, production planning, computer operations, financial assets, risk management and many other fields in business productivity. Except for the industrial sector, OR techniques has been applied in sectors like health care, energy policy, defence and water resource planning.

Consequently, OR is one of the most significant tools in improving the process of decision making and its framework should be clear in order to be applied to a generic problem. This framework is constituted of seven sequential steps: Orientation, Problem Definition, Data Collection, Model Formulation, Solution, Model Validation and Output Analysis, and Implementation and Monitoring. These seven steps is a circle and each step feeds information to the next step and as a result a continuous feedback as it is shown in the next Figure.

This work concentrates on the stage of computing the optimal solution when the problem is represented by a model. In the past, researchers have spent a lot of their efforts in this field and as a result nowadays there is a large amount of methods for this purpose. As it is mentioned before, in OR the problem is expressed in the form of a model and linear programming is a technique for identifying the best choice based on certain criteria. More specific, the goal of linear programming is to optimise some criteria within some constraints. The model consists of an objective function which represents profit, loss or return on investment and a number of constraints in the allocation of the available resources. There is a huge number of methods to solve a LP. The main goal of this thesis is to introduce a new approach and method for solving a LP, at the same time of improving the computational performance in contrast to well-known techniques.

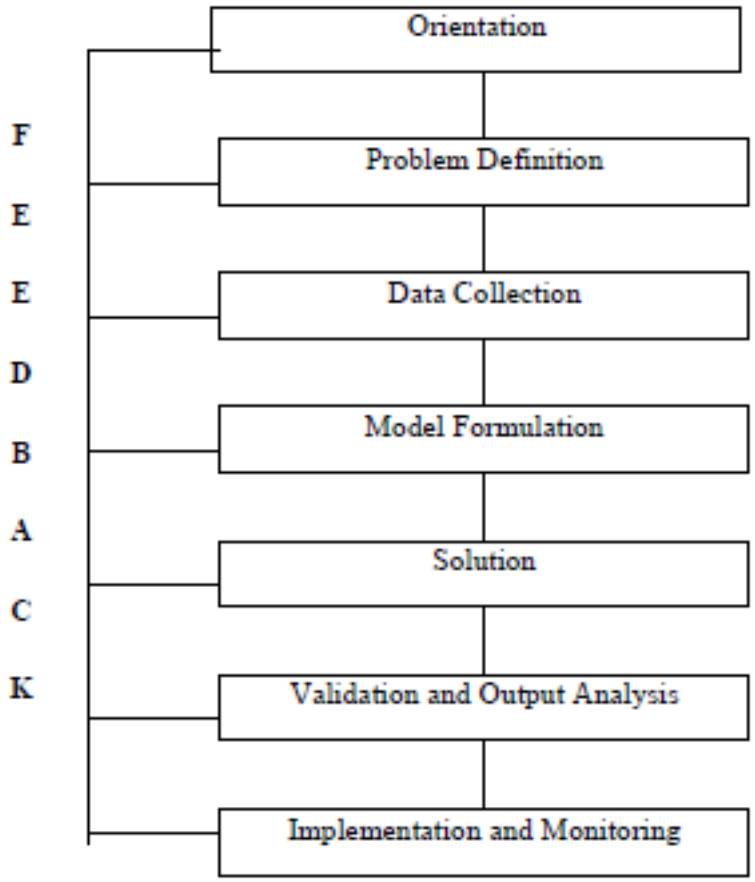

FIGURE 1.1: The Operations Research Approach.

1.3 Overview

In Chapter 2, there is the presentation of the fundamentals of linear programming and its evolution through the last decades. Moreover, the mathematical model of LPs and notations are included. We discuss the significance of linear programming in the attempt of solving real life problems. In addition, we describe the crucial characteristics of duality and how it contributed to the improvements of linear programming algorithms. At last, we mention the significant geometric interpretations of LPs which can reveal many interesting attributes of linear programming.

In Chapter 3, we present the main categories of linear programming algorithms which are mainly separated into 2 different families: (i) simplex-type algorithms, and (ii) interior point methods. Moreover, we discuss the primal revised simplex algorithm, the most widely-used method to solve a LP and we present the primal exterior point simplex algorithm, a simplex-type algorithm that uses two paths to reach the optimal solution. Furthermore, a brief overview of basic concepts of interior point methods is included with the Mehrotra's Predictor-Corrector method, a primal-dual infeasible interior point

method. For each one of these three algorithms, we present its main steps and some illustrative examples in order to demonstrate theirs steps and processes.

In Chapter 4, we describe the significance of preprocessing in linear programming. The main reason for making the presence of preprocessing necessary is the numerical difficulties that a software application may face while solving a large scale LP. Consequently, we implement three different scaling techniques and nine presolve methods: (i) elimination of empty rows, (ii) elimination of empty columns, (iii) elimination of singleton equality rows, (iv) elimination of singleton inequality rows, (v) elimination of dual singleton inequality rows, (vi) elimination of implied free singleton columns, (vii) elimination of implied bounds from constraints, (viii) making coefficient matrix A structurally full rank, (ix) elimination of linearly dependent rows that aim to reduce the size of a LP. Moreover, the scaling process includes 3 different methods, the equilibration, the geometric mean and the hybrid technique which combines the first two methods. Finally, a new presolve technique is introduced with significant computational results from the analysis of primal feasibility conditions.

In Chapter 5, PDIPSA is presented, a variation of Primal Dual Exterior Simplex Algorithms which have been proved extremely computational effective in practice. PDIPSA's main difference stems from the fact that it traverses across the interior of the feasible region in an attempt to avoid combinatorial complexities of vertex following algorithms. Furthermore, this difference is clearly demonstrated through a geometric example which illustrates the new approach which is included by PDIPSA in contrast to others PDEPSAs. Moreover, we include the proof of correctness of the algorithm and we present the method for solving general LPs. Finally, a computational study is presented to highlight the efficiency of the algorithm.

In Chapter 6, we present a hybrid algorithm which is a combination of an interior point method and an exterior point simplex algorithm. The idea of combining these two different types of methods stemmed from the observation that IPMs are able to spot very fast feasible solutions with good objective values, but they need a relatively long time to converge to the optimal solution. In order to take full advantage of EPSA, we used the PDIPSA which is described in Chapter 5. Primal-dual algorithms can deal more effectively with the problems of stalling and cycling and as a result, they are able to improve the performance of EPSA. The IPM, which we use in our hybrid algorithm, is Mehrotra's Predictor-Corrector method, an infeasible primal-dual IPM. The main advantage of this hybrid algorithm is that it exploits the strengths of both IPM and PDIPSA. In the first iterations, IPM moves inside the feasible area and computes interior points. Finally, in order to gain an insight into the practical behaviour of the proposed

algorithm, we have performed a computational study on a set of benchmark problems (Netlib, Kennington, Mészáros).

In the last Chapter, we conclude the main results of the current thesis and we present some ideas for further work in the future.

Chapter 2

Linear Programming

2.1 Introduction

The transformation of a real-life problem into a mathematical model is the main idea of linear programming. Moreover, linear programming focus on the process of analysing and solving the model with designing algorithms which can conclude to the solution of initial real-life problem. The main parts of the mathematical model is the objective function which has to be optimized(minimized or maximized), a finite number of constraints and a finite number of decision variables.

Linear programming is the main part of management science and operations research. Furthermore, decision support systems base on linear programming models for the quantitative analysis of real-life problems and this is the main reason for being so widely known among the scientific community. Some of the most interesting issues that benefit from linear programming are the economic optimization problems, strategic planning problems and algorithm analysis. Apart from its practical applications, linear programming includes significant aspects from theoretical point of view referring to algorithmic and algebraic concepts with geometric angles of incidences. Consequently, many researchers focused their attention on linear programming and how they could enhance algorithms and computational methods in order to formulate and solve problems from very different scientific areas. However, many concerns arise especially when the problem's size grows and they are referring to accuracy errors, optimal data structures and implementation issues. Except for this, linear programming also includes significant geometric interpretations, for example in small size problems, a graphical representation of the objective function and the feasible region can easily reveal the optimal solution of the initial model.

As it mentioned previously, linear programming refers to an optimization problem with three essential ingredients: a variable vector x consisting of a set of unknowns to be determined, an objective function of x to be optimized, and a set of constraints to be satisfied by x and all of them are linear inequalities and/or equalities. Consequently, linear programming is the subject of studying and solving LPs. The needs of the second World War and the necessity of solving military logistic problems introduced the fundamental principles of linear programming. However, its applications are still used by a great number of sectors both in scientific community and in industrial area.

2.2 History of Linear Problem

The search of the origin and the development of every research area through decades are always interesting and beneficial for everyone. Likewise, the first steps of linear programming were made in early 1820s from the famous French mathematician J.B.J. Fourier who gave his name to mathematical series and from V. Poussin in early 1910s but both of their works went unnoticed. Another significant academician was L.V. Kantorovich who published the book *Mathematical Method of Production Management and Planning* in 1939 [58] and he presented mathematical models and how real-life problems could be solved. Moreover, F.L. Hitchcock introduced on his paper the transportation problem in 1941 [52]. All of these research work remained unnoticed until the late 1940s and early 1950s when simplex method triggered the spread of linear programming.

World War II was a crucial point for the development of linear programming, Operations Research and Optimization have greatly benefited. Despite the fact that World War II was the deadliest military conflict in history in absolute terms of total dead and over 60 million people were killed, which was about 3% of the 1940 world population, it gave rise to a large number of scientific areas.

After the end of the World War II, George B. Dantzig who already had received his PhD, was a mathematical advisor to U.S. Air Force Controller in 1946 [22]. His main challenge was to create a process for human assignment to assembly line and to optimize the logistical supply program. After the analysis stage, he created a model without an objective function. In 1947, Dantzig proposed an algorithm with the name *Simplex* in tableau format which is recognized as one of the top algorithms of the 20^{th} century [24] [25] [26] and it is marked as the birth of linear programming. The original example of Dantzig included the best assignment of 70 people to an equal number of jobs. From one point of view, the problem is a 1-to-1 correspondence between people and the jobs. In practice, there are 70! activities and it is impossible to determine the best among them by comparing all, although the different assignments are a finite number. The

specific example clearly shows the difficulties that decision makers faced before 1947, where experience and mature judgement were the only tools for a 'good' solution which was not the optimal in many cases.

As it is known, the electronic computer provoked dramatical changes in all scientific areas and of course to linear programming and the simplex method. The first attempts took place at National Bureau of Standards in USA by A. Hoffman and his team. In 1954 Orchard-Hays[85] focused on introducing a simplex method based on commercial software and his work were further developed by man researchers during the next years. From its first steps, linear programming has significant applications like in military, economic and industrial. Some of the most well known problems which can be solved with the linear programming are the transportation[52] and the diet problem[108].

In this two decades, 50's and 60's, three main new areas attract the academic attention, the matrix factorization theory, network flows and the duality theory [69] [71] [1]. Later, the matrix inversion became the subject for serious research work and the handling of sparse matrices arose and finally became part of all the commercial software [103] [104].

Consequently, a big discussion about the practical performance of the Simplex method started and there were many attempts to estimate the execution time of the algorithm proportionally to the problem size [61] [62] [84]. The simplex algorithm has exponential time computational complexity [64], when a polynomial time one is the desirable complexity. In contrast to simplex, the former Soviet Union mathematician L.G. Khachiyan [63] proposed a first polynomial time LP solver, the ellipsoid algorithm. However, it's computational performance were poorer than the Simplex's, due to the fact that the polynomial time complexity is the worst case complexity and in practice it is rare.

In 1984, Indian mathematician Narendra Karmarkar (1984) introduced an Interior Point Method (IPM) which was promisingly fast in practice and its complexity was a polynomial time. Karmarkar's method was based on previous of affine scaling algorithm [28], [29], the method of logarithmic barriers of Frisch [35] [36] and the method of the central path of Huard [54] [55]. The specific algorithm uses a set of interior points and it computes the optimal solution within few iterations in most cases.

The most common variation of IPMs is the primal dual algorithms where both the primal and the dual problem are solved simultaneously. Moreover, it has been observed that most of the IPMs are infeasible in nature, they start from an infeasible point and they enter the feasible region after of some iterations. Lustig [70] were the first who presented infeasible IPMs. Moreover, the primal-dual predictor-corrector path by Mehrotra [79] is the most well known variant of IPMs and the LinProg optimization package of Matlab is based on one of these variants.

Meanwhile, the development of Simplex method continued and Harris [51] presented new pivot rules. After of few years, recurrence formulas with promising numerical results were presented by Forrest and Goldfarb [33]. As it is mentioned previously, the worst case complexity is not very common in real experience and Borgwardt [14] [15] showed that the average time complexity of Simplex method is closer to polynomial rather than to exponential. In addition to this, Smale [106] [107] using another pivot rule agreed with the previous statement.

In the late 80's Exterior Point Simplex Algorithms (EPSA) arose in literature, they were a new type of Simplex and they based on two paths by Paparrizos [91]. However, after of few years Paparizzos presented an EPSA for a network optimization problem, applicable to the assignment problem. There are many research papers which describe variants of EPSA family algorithms [6] [20]. Morever, Paparrizos introduced the generalization of EPSA for LP later [92] and a full analysis of EPSA family there is in Paparrizos et al. [94]. EPSA in contrast to Primal Simplex Algorithm uses two paths to converge to the optimal solution, the first path constitutes of infeasible points and the other of feasible points. The most significant advantage of the two paths method is that EPSA does not move from one adjacent vertex to another but it visits infeasible points and as a result it can calculate the optimal solution in less steps (iterations).

Furthermore, there are two crucial points for improving the computational behaviour of Simplex Algorithm, the initial basic partition and the pivoting rule which refer to the choice of potential entering and leaving variables of the LP. The pivoting rule affects dramatically the number of iterations which needed in order to reach the optimal solution and it is of great value, especially for large scale LPs. There are many and great papers [76] [87] [111] about pivoting rules which focused on enhancing the computational performance of Simplex. Pan presented great works [88] [89] with promising computational performance against other pivoting rules like Dantzig's. Moreover, Arsham et al. [7] proposed a total new technique which is independent of artificial variables. Through decades, there were many new methods for pivoting [90] and some of them consisted of interior and boundary points and they were the first immature attempts to combine completely different approaches. In addition, Malakooti and Najjar [75] [83] presented a combination of IMPs and boundary methods like Simplex with significant results and its main advantage is its capability to avoid large number of boundary points.

The current thesis presents variations of EPSA and it attempts to enhance the computational performance of previous exterior algorithms which focus on both the primal and the dual LP, Primal Dual Exterior Point Simplex Algorithm (PDEPSA). Apart from that, the proposed enhanced version of PDEPSA is combined with an IPM in order to

examine a completely different approach of exploiting the strengths of PDEPSAs and IMPs while avoiding their weaknesses.

2.3 The Linear Problem

The main purpose of linear programming is to choose the best way to allocate limited sources in a specific problem. For example, in the case of a manufacturer who is producing several different types of goods, he must decide how many of each type to produce. However, there are some limitations or constraints that manufacturer should take them under consideration, like the available labor and the fact that the machines can only run for a certain number of hours a day before they overheat. Moreover, his main target is to produce enough goods to meet the demands of his customers and maximize his earnings at the same time of reducing the costs of his company.

Likewise, lets assume the case of an airline which is interested of assigning crews to flights to minimize layover time. In addition, the workers' contracts limit the number of hours per day that employees are allowed to work. Apart from that, the contract specifies that an individual in a crew may not fly more than three flights per day. Consequently, the airline must decide under these and other constraints for the assignments that it should make. All of these types of problems are frequently met in linear programming where the goals are to maximize profits or minimize costs, or maximize output or minimize waste.

An illustrative example

One of the most well-known LPs is the diet problem. Considering the following example, a dietician is preparing a diet consisting of two foods, A and B. Each unit of food A contains 10 grams of protein, 7 grams of fat and 25 grams of carbohydrate and costs 70 cents. Each unit of food B contains 40 grams of protein, 8 grams of fat and 17 grams of carbohydrate and costs 30 cents. Moreover, the diet being prepared should contain the following minimum requirements: at least 50 grams of protein, at least 24 grams of fat and at least 30 grams of carbohydrate. Consequently, the dietician has to calculate the units of each food in order to satisfy all the minimal requirements and to minimize the cost for his customer.

Data problem

All the appropriate information is on the next table in order to make the formulation easier:

TABLE 2.1: Diet problem

	Food A	**Food B**	**Minimum Requirements**
Protein	10	40	50
Fat	7	8	20
Carbohydrate	25	17	35
Cost	0.7	0.3	

The numbers on the right side, 50, 20, 35 represent the minimum amount in grams of protein, fat and carbohydrate in the diet. The numbers inside the table represent the number of grams of either protein, fat, or carbohydrate in each unit of food. There are in fact an infinite number of ways of combining foods A and B in order to meet the minimum requirements and it is obvious that this type of problems is not easy, especially when the number of foods is much larger.

Set up the model

Since, the number of units of foods A and B are the quantities which must be calculated, let $x_1 =$ the number of units of food A used in diet and let $x_2 =$ the number of units of food B used in diet. Since, each unit of food A costs €0.70, the x_1 units of food A that are used will cost €$0.7x_1$ and for food B the cost is €$0.3x_1$. Hence, the total cost is

$$C = 0.7x_1 + 0.3x_2 \tag{2.1}$$

which is what it should be minimized.

The next step refers to minimum requirements. According to the table above, each unit of food A contains 10 grams of protein, so the x_1 units of food A contains $10x_1$ grams of protein. Similarly, the food B contains $40x_1$ grams of protein. Consequently, the total number of grams of protein used in a diet containing x_1 units of food A and x_2 units of food B is

$$10x_1 + 40x_2$$

In addition, according to minimum requirements must be at least 50 grams. This leads to the inequality

$$10x_1 + 40x_2 \geq 50 \tag{2.2}$$

Similarly, for the fat requirement:

$$7x_1 + 8x_2 \geq 20 \tag{2.3}$$

And for the carbohydrate requirement:

$$25x_1 + 17x_2 \geq 35 \tag{2.4}$$

There are certain obvious but not explicitly stated restrictions in the variables, that the number of units of food A and the number of units of food B used must both be non-negative. That is,

$$x_1, x_2 \geq 0 \tag{2.5}$$

Combining (2.1)-(2.5):

$$\begin{array}{lrcrcl} \min & c = & 0.7x_1 & + & 0.3x_2 & \\ s.t. & & 10x_1 & + & 40x_2 & \geq 50 \\ & & 7x_1 & + & 8x_2 & \geq 20 \\ & & 25x_1 & + & 17x_2 & \geq 35 \\ x_j \geq 0, & (j = 1, 2) & & & & \end{array}$$

The constraints (2.1)-(2.4) are called the main constraints of the LP and constraints (2.5) are called non-negative or physical constraints. The cost function (2.1) is called the objective.

The Standard Form of a LP model

Decision Variables

A LP model contains real decision variables, denoted by $x_1, \ldots, x_n$ where n is a finite positive integer. In above example there are 2 decision variables x_1 and x_2.

Objective Function

The objective function $c_1x_1 + \ldots + c_nx_n$ is a linear function in the n decision variables with $c_1, \ldots, c_n$ real numbers, called the *objective coefficients*. Depending on whether the objective function has to be maximized or minimized, the objective function of the model is written as:

$$max(c_1x_1 + \ldots + c_nx_n) \text{ or}$$
$$min(c_1x_1 + \ldots + c_nx_n)$$

respectively. In the diet example, the objective function is $min(0.7x_1 + 0.3x_2)$. The values of the objective function are called the *objective values*.

Constraints

A constraint of a LP model is either a $\leq, \geq or =$ and the symbol $\oplus$ is used to represent one of them:

$$a_{i1}x_1 + \ldots + a_{in}x_n \oplus b_i$$

The entry a_{ij} is the coefficient of the $j-th$ decision variable x_j in the $i-th$ constraint. Let m be the number of constraints. All (left-hand sides of the) constraints are linear in $x_1, \ldots, x_n$. For $i \in 1, \ldots, m$ and $j \in 1, \ldots, n$, the entries a_{ij}, b_j and c_i are real numbers and are called the parameters of the LP.

Non-negativities

A non-negativity of a LP-model is an inequality of the form:

$$x_i \geq 0$$

The set of vectors in $\Re^n$ satisfying the constraints and the non-negativities of the model is called the *feasible region*. A LP model is called feasible if its feasible region is not empty, otherwise is called infeasible. An *optimal solution* of a maximizing (respectively, minimizing) LP model is a point in the feasible region with maximal (or minimal) objective value, this objective value is called the *optimal objective value*.

Consequently, a minimizing LP model, with '$\geq$' constraints and non negativities can be written:

$$\begin{array}{lrcccccc}
\min & c_1x_1 & + & \ldots & + & c_nx_n & & \\
s.t. & a_{11}x_1 & + & \ldots & + & a_{1n}x_n & \geq & b_1 \\
 & \vdots & + & \ldots & + & \vdots & \geq & \vdots \\
 & a_{m1}x_1 & + & \ldots & + & a_{mn}x_n & \geq & b_m \\
x_j \geq 0, & (j = 1, \ldots, n) & & & & & &
\end{array}$$

A more general form of the standard model of LP is the following:

$$\begin{array}{ll}
min & c^T x \\
s.t. & Ax \oplus b \\
 & x \geq 0
\end{array}$$

The Typical Form of a LP model

LPs are presented in two different forms, there is the general and the typical form. In typical form all of the constraints are converted to absolute equalities and as a result we can express the LP with the following form:

$$\begin{aligned} &min && c^T x \\ &s.t. && Ax = b \\ & && x \geq 0 \end{aligned} \qquad (LP.1)$$

In order to transform a problem of standard form to typical form, slack variables are used. In the diet problem the 1^{st} constraint $10x_1+40x_2 \geq 50$ can be written $10x_1+40x_2-x_3 = 50$ if variable x_3 is added and it is called the slack variable of the constraint (2.2).

Including slack variables for all constraints of the diet problem, the LP can be written as:

$$\begin{aligned} \min \quad c = \; & 0.7x_1 + 0.3x_2 \\ s.t. \quad & 10x_1 + 40x_2 - x_3 = 50 \\ & 7x_1 + 8x_2 - x_4 = 20 \\ & 25x_1 + 17x_2 - x_5 = 35 \\ x_j \geq 0, \quad & (j = 1, 2, 3, 4, 5) \end{aligned}$$

The vast majority of LPs are real-life problems and this is the reason that concludes to the non negativity constraint for the decision variable x, another term for the variable x is the natural constraint. From now on we assume every LP as minimization problem, there are maximization problems too, but we can easily transform them to minimization problems, if we use the opposite objective function in the mathematical model. With the term feasible points we refer to the set of points that satisfy all the constraints and they are also known as the feasible region of the LP.

2.4 Duality

J. Von Neumann [117] presented for first time the dual form of a LP. There are strong connections between the dual and the initial problem, the variables of the dual problem are associated with the constraints and non-negativities of the original problem. The initial problem is called primal model. The most significant issue is that the dual problem includes optimality criteria for feasible solutions of the primal problem. Moreover, the

sensitivity analysis of the primal model is based on the dual problem. The primal problem refers for example to the optimization of the amount of profit obtainable from the production activity, in contrast the dual is associated with economic worth of the limited resources and capacities.

Consider the next 2 LPs:

$$\begin{aligned} \max \quad & 5x_1 + 2x_2 \\ s.t. \quad & x_1 + 3x_2 \leq 12 \\ & 3x_1 - 4x_2 \leq 9 \\ & 7x_1 + 8x_2 \leq 20 \\ x_j \geq 0, \quad & (j = 1, 2) \end{aligned} \tag{2.6}$$

$$\begin{aligned} \min \quad & 12w_1 + 9w_2 + 20w_3 \\ s.t. \quad & w_1 + 3w_2 + 7w_3 \geq 5 \\ & 3w_1 - 4w_2 + 8w_3 \geq 2 \\ w_j \geq 0, \quad & (j = 1, 2, 3) \end{aligned} \tag{2.7}$$

The two LPs 2.6 and 2.7 are very different each other. They have a different number of variables and a different number of constraints. However, with a closer look, it seems that there are some relationships between them.

First of all, the number of constraints in 2.6 is equal to the number of variables in 2.7 and the number of constraints in 2.7 is equal to the number of variables in 2.6.

Secondly, the constants on the right hand side of the main constraints of 2.6 are precisely the coefficients in the objective function of 2.7,while the constants on the right hand side of the main constraints of 2.7 are the coefficients of the objective function of 2.6.

Moreover, the coefficients of the first variable x in 2.6 are precisely the coefficients of the first constraint in 2.7, while the coefficients in the second column of 2.6 are precisely the coefficients of the second constraint of 2.7.

Furthermore, all of the inequalities have been reserved.

Last but not least, the original LP was a maximum program, while this new program is a minimum program.

While these relationships might show that the two linear programs are related, the fact remains that these programs are very different and many people would probably feel that there really should be no connection between the optimal solution of 2.6 and that of 2.7. The fact is that there is an extremely close relationship between the optimal values of the objective functions of these two LPs.

Programs 2.6 and 2.7 are called dual linear programs. More specifically, given a maximum program then its dual program is a minimum program formed in the following way. Associate with each main constraint in the original program a variable. These variables are called dual variables. So in 2.6 the first constraint is associated with the variable w_1, the second constraint with the variable w_2 and the third with the variable w_3. Moreover, the constants on the right hand side of the main constraints of the original program become the coefficients of these dual variables in the new objective function and this new dual objective function is to be minimized. The coefficients in the first column of the original program become the coefficients of these dual variables in the first constraint of the new LP. Similarly, the coefficients of the second column of the original LP become the coefficients of the second constraint in the new LP, etc. In addition, all the inequalities are reserved.

2.5 Geometry

In previous sections, we discussed the formulation of several types of LPs. In the current section, by examining matters geometrically [102], we are able to lay some of the groundwork for the very powerful and interesting aspects of linear programming.

Consider the next LP:

$$\begin{array}{rrcrcr} \max & 2x & + & 3y & & \\ s.t. & 2x & + & 4y & \geq & 8 \\ & 2x & + & 5y & \leq & 18 \\ & 3x & + & y & \geq & 5 \\ & x & - & 2y & \leq & 2 \\ x, y \geq 0 & & & & & \end{array} \tag{2.8}$$

Using the graphical method, the constraints and non-negativities are drawn in a rectangular coordinate-system.

In Fiqure 2.1 the decision variables x and y are non-negative and we conclude to the shaded area determined by all the constraints. Moreover, we end up with the feasible region which includes all the points which satisfy all the constraints and non-negativities.

Moreover, it may happen that a LP has more than one optimal solution, then the optimal solution is called multiple. There are three different types of feasible regions:

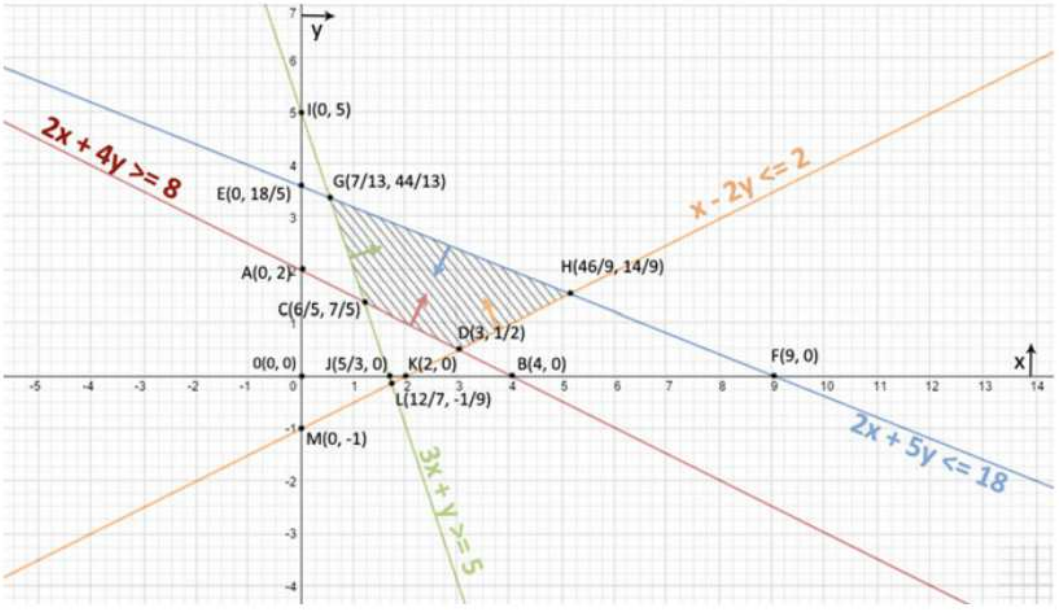

FIGURE 2.1: Example of feasible region

Feasible Region Bounded. A non empty feasible region is called *bounded*, if all decision variables are bounded on the feasible region. Consequently, the objective values are then also bounded on the feasible region, see Fiqure 2.2.

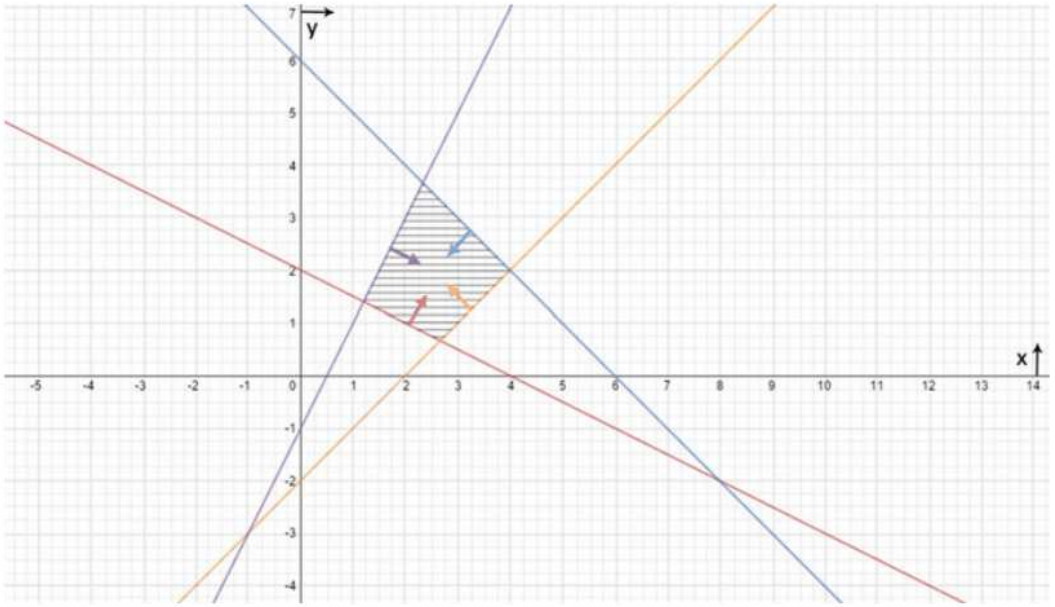

FIGURE 2.2: Example of Feasible Region Bounded

Feasible Region Unbounded. A non empty feasible region is called *unbounded* if it is not bounded, at least one of the decision variables can take on arbitrarily large values on the feasible region. It depends on the objective function whether an optimal

solution exists, in case that an optimal solution does not exist then the model is called *unbounded*, see Fiqure 2.3.

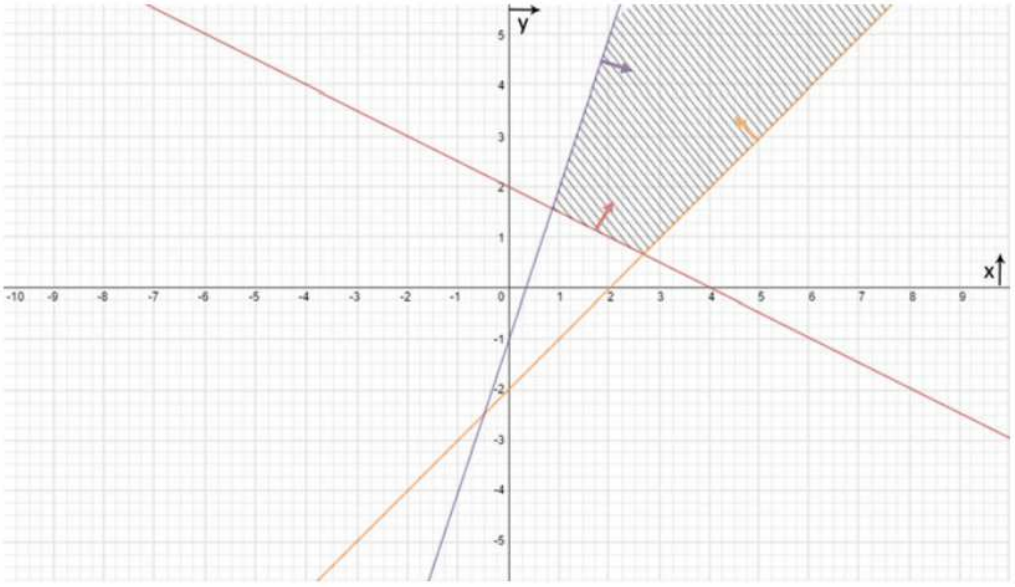

FIGURE 2.3: Example of Feasible Region Unbounded

Feasible region empty. If the feasible region does not include any point then the LP is called *infeasible*, see Fiqure 2.4.

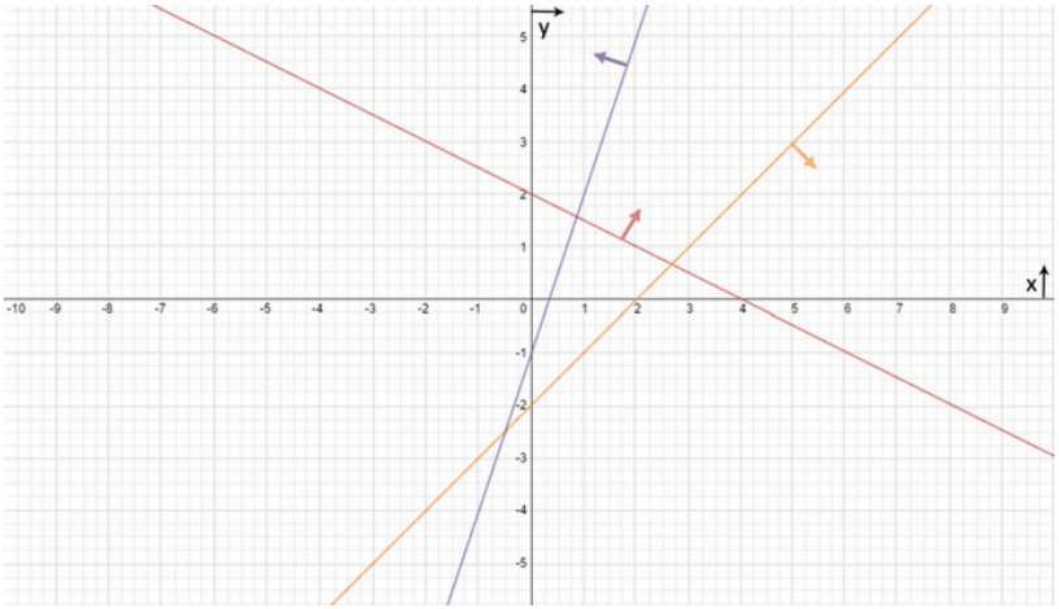

FIGURE 2.4: Example of Feasible region empty

Chapter 3

Linear Programming Algorithms

3.1 Introduction

There are mainly two-different families of linear programming algorithms: (i) simplex-type algorithms, and (ii) interior point methods. This Chapter will give a brief overview to these different families of algorithms. More specifically, Section 3.2 presents the primal revised simplex algorithm, the most widely-used method to solve a LP. Section 3.3 presents the primal exterior point simplex algorithm, a simplex-type algorithm that uses two paths to reach the optimal solution. Section 3.4 gives a brief overview of basic concepts of interior point methods and presents the Mehrotra's Predictor-Corrector method, a primal-dual infeasible interior point method. For each one of these three algorithms, we present its main steps and also provide an example to demonstrate its execution. Finally, conclusions are presented in Section 3.5.

3.2 Simplex Algorithm

The most widely-used method for solving LPs is the simplex algorithm, proposed by Dantzig [25] [27]. The simplex algorithm is one of the top ten most important algorithms in the 20th century [32]. The simplex algorithm starts with a feasible basic solution and reaches an optimum solution by moving from one feasible solution to another, along the edges of the feasible region. The simplex algorithm guarantees monotonicity of the objective value. In the rest of this Section, we present the primal revised simplex algorithm.

Prior to the application of the simplex algorithm, a presolve and a scaling routine should be executed to reduce the problem's dimensions and improve the computational properties of the constraint matrix A (for more details, see Chapter 4). Then, an initial basis should be calculated in order to initialize the simplex algorithm. The simplest initial basis is the all-artificial basis [10, 21]. Artificial variables are added to all constraints of the original LP. The initial basis is matrix I. This method is very simple, but usually results in many iterations. Another simple initial basis is the slack-artificial basis [11]. Slack and surplus variables are added firstly to all inequality constraints. Then, artificial variables are added to equality constraints and inequality constraints of the type $\geq$. The initial basis is a matrix containing the slack variables added in inequality constraints of type $\leq$ and the artificial variables. The slack-artificial basis is better than the all-artificial basis but it also leads to more iterations than other methods. Many sophisticated methods have been proposed in the literature for the initialization of the simplex algorithm [17] [47] [11]. In our implementation, we use a Gauss-Jordan elimination with partial pivoting that calculates a row echelon form of the constraint matrix. This method is more time-consuming than using the all-artificial or the slack-artificial basis, but it yields more advanced bases and it also leads the revised simplex algorithm to perform less iterations than using the other two previously mentioned methods.

Having calculated a basic partition (B, N), if the solution $x_B \geq 0$ ($x_B = A_B^{-1}b$), then we apply the revised simplex algorithm to solve the original problem. Otherwise, we formulate an auxiliary problem to find an initial feasible basis. Three methods are mainly used to drive out the artificial variables: (i) the two-phase method, (ii) the big-M method, and (iii) the single artificial variable method.

The big-M method attempts to execute Phase I and Phase II in a single execution of the simplex algorithm. The big-M method adds an artificial variable to each constraint and modifies the objective function in order to penalize the artificial variables:

$$\begin{aligned} \min \quad & c^T x + M e^T y \\ s.t. \quad & Ax + I_m y = b \\ & x, y \geq 0 \end{aligned}$$

where M is a large positive constant, much larger than the largest coefficient in vector c. If the original LP is feasible and its optimal value is finite, all artificial variables will eventually be driven to 0.

The single artificial variable method selects a submatrix $A_B \in \mathbb{R}^{m \times m}$ of A to be the basis irrespective of whether the solution $x_B = A_B^{-1}b \geq 0$. If A has full row rank, then such a matrix exists; if not, we can apply a presolve method prior to the execution of the

simplex algorithm to make the constraint matrix A of full row rank. If $x_B = A_B^{-1}b \geq 0$, then an initial basic feasible solution was found and we can apply simplex algorithm to solve the original LP. If $x_b \ngeq 0$, then we add a single artificial variable and we can use either the two-phase or the big-M method.

In our implementation, we use the two-phase method along with the single artificial technique. The two-phase method adds an artificial variable to each constraint and solves an auxiliary LP in Phase I:

$$\begin{aligned} \min \quad & e^T y \\ s.t. \quad & Ax + I_m y = b \\ & x, y \geq 0 \end{aligned}$$

where $e \in \mathbb{R}^n$ is a vector of ones. The auxiliary LP is solved using the simplex algorithm.

The leaving variable k is the index with the minimum value of x_B. The artificial variable y enters the basis. Hence, we update the basic partition, $B = B \cup [n+1] \setminus [k]$ and $N = N \cup [k]$.

The last column of the constraint matrix A is calculated as:

$$d = -A_B e$$

Initially, we calculate the following matrices and vectors:

$$A_B^{-1},\ x_B = A_B^{-1}b,\ w = c_B^T A_B^{-1}, \text{ and } s_N = c_N^T - wA_N$$

Then, we perform the optimality test. We terminate the algorithm or proceed with Phase II in the following cases:

- If $s_N \geq 0$ and $y \neq 0$, then the LP is infeasible. Otherwise, we proceed with Phase II.
- If the artificial variable y left the basic list, we proceed with Phase II.

If the auxiliary LP is not optimal, then we continue with the remaining steps of Phase I. We use Dantzig's rule to choose the entering variable. Dantzig's pivoting rule selects the column A_l with the most negative $\overline{s_l}$. We also calculate the pivoting column: $h_l = A_B^{-1}A_l$.

Next, we perform the minimum ratio test in order to calculate the leaving variable k:

$$x_k = x_{B[r]} = \frac{x_{B[r]}}{h_{il}} = min\left\{\frac{x_{B[i]}}{h_{il}} : h_{il} > 0\right\}$$

Then, we update the basic partition, $B = B \cup [l] \setminus [k]$ and $N = N \cup [k] \setminus [l]$. Finally, we update the basis inverse using the Product Form of the Inverse (PFI) method. The PFI scheme updates the basis inverse using the following equation:

$$\left(A_{\overline{B}}\right)^{-1} = (A_B E)^{-1} = E^{-1}(A_B)^{-1} \tag{3.1}$$

where E^{-1} is the inverse of the eta-matrix and can be computed by the the following equation:

$$E^{-1} = I - \frac{1}{h_{rl}}(h_l - e_l)\, e_l^T = \begin{bmatrix} 1 & & -h_{1l} & & \\ & \ddots & \vdots & & \\ & & 1/h_{rl} & & \\ & & \vdots & \ddots & \\ & & -h_{ml}/h_{rl} & & 1 \end{bmatrix} \tag{3.2}$$

Note that the basis inverse is calculated from scratch every 80 iterations. Finally, we update vectors x_B, w and s_N:

$$x_B = A_B^{-1}b,\ w = c_B^T A_B^{-1}, \text{ and } s_N = c_N^T - wA_N$$

We start again with the optimality test and all other steps presented previously for Phase I until we find a feasible basis for the original problem or we find that the original problem is infeasible. In the first case (feasible basis), we continue with Phase II by applying the revised simplex algorithm to the original problem. We start by initializing the needed vectors and matrices:

$$A_B^{-1},\ x_B = A_B^{-1}b,\ w = c_B^T A_B^{-1}, \text{ and } s_N = c_N^T - wA_N$$

Next, we perform the optimality test of Phase II. If $s_N \geq 0$, then the linear programming problem is optimal. Otherwise, we continue with the remaining steps of Phase II. We use Dantzig's rule to choose the entering variable. Dantzig's pivoting rule selects the column A_l with the most negative $\overline{s_l}$, as already mentioned. We also calculate the pivoting column: $h_l = A_B^{-1}A_l$.

If $h_l \leq 0$, then the LP is unbounded. Otherwise, we perform the minimum ratio test to calculate the leaving variable k:

$$x_k = x_{B[r]} = \frac{x_{B[r]}}{h_{il}} = min\left\{\frac{x_i}{h_{il}} : i \in B, h_{il} > 0\right\}$$

In case of ties when selecting the entering and leaving variable, we break ties by selecting the variable with the smallest subscript. Then, we update the basic partition, $B = B \cup [l] \setminus [k]$ and $N = N \cup [k] \setminus [l]$. Finally, we update the basis inverse using the PFI method:

$$\left(A_{\overline{B}}\right)^{-1} = (A_B E)^{-1} = E^{-1}(A_B)^{-1} \tag{3.3}$$

Note again that the basis inverse is calculated from scratch every 80 iterations. Finally, we update vectors x_B, w and s_N:

$$x_B = A_B^{-1}b,\ w = c_B^T A_B^{-1}, \text{ and } s_N = c_N^T - wA_N$$

We start again with the optimality test and all other steps presented previously for Phase II until we find a feasible basis for the linear programming problem or we find that the problem is unbounded.

A formal description of the revised simplex algorithm is given in Table 3.1.

TABLE 3.1: Revised Simplex Algorithm

Step 0. *(Initialization).*
Presolve and scale the LP.
Select an initial basic partition (B, N).
If $x_B = A_B^{-1}b \geq 0$, proceed to Step 2.
Step 1. *(Phase I).*
Construct an auxiliary problem by adding an artificial variable y with a coefficient vector equal to $-A_B e$.
Apply the revised simplex algorithm in the auxiliary problem.
If the final basic solution (B, N) is feasible, then proceed to Step 2.
Else the problem is infeasible.
Step 2. *(Phase II).*
Step 2.0. *(Initialization).*
Compute $(A_B)^{-1}$ and vectors x_B, w and s_N.
Step 2.1. *(Test of Optimality).*
If $s_N \geq 0$ then the problem is optimal.
Else choose the index l of the entering variable using Dantzig's rule.
Step 2.2. *(Pivoting).*
Compute the pivot column $h_l = (A_B)^{-1}A_l$.
if $h_l \leq 0$ then the problem is unbounded.
else choose the leaving variable $x_{B[r]} = x_k$ using the following relation:

$$x_k = x_{B[r]} = \frac{x_{B[r]}}{h_{il}} = min\left\{\frac{x_i}{h_{il}} : i \in B, h_{il} > 0\right\}$$

Step 2.3. *(Update).*
Swap indices k and l. Update the new basis inverse $\left(A_{\overline{B}}\right)^{-1}$, using PFI.
Update vectors x_B, w and s_N.
Go to Step 2.1.

Next, we will demonstrate the revised simplex algorithm with an example. The LP that will be solved is the following:

$$\begin{array}{llrcrcrcl} \min & z = & 8x_1 & + & 4x_2 & - & 6x_3 & & \\ s.t. & & x_1 & + & x_2 & + & x_3 & \leq & 2 \\ & & 3x_1 & + & x_2 & - & x_3 & \geq & 3 \\ & & 3x_1 & + & 2x_2 & - & x_3 & \geq & 5 \\ x_j \geq 0, & (j = 1, 2, 3) & & & & & & & \end{array}$$

Let's convert the LP in its standard form:

$$\begin{array}{llrcrcrcrcrcl} \min & z = & 8x_1 & + & 4x_2 & - & 6x_3 & & & & & & \\ s.t. & & x_1 & + & x_2 & + & x_3 & + & x_4 & & & = & 2 \\ & & 3x_1 & + & x_2 & - & x_3 & & & - & x_5 & = & 3 \\ & & 3x_1 & + & 2x_2 & - & x_3 & & & & - \; x_6 & = & 5 \\ x_j \geq 0, & (j = 1, 2, 3, 4, 5, 6) & & & & & & & & & & & \end{array}$$

So, the matrices and vectors that will be given as input to simplex algorithm are the following:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 3 & 1 & -1 & 0 & -1 & 0 \\ 3 & 2 & -1 & 0 & 0 & -1 \end{bmatrix}, \; c = \begin{bmatrix} 8 \\ 4 \\ -6 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; b = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$$

We start with the basic partition $B = [4, 5, 6]$ and $N = [1, 2, 3]$. The initial basis is not feasible ($x_B \ngeq 0$):

$$A_B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = A_B^{-1}$$

$$x_B = A_B^{-1} b = \begin{bmatrix} 2 \\ -3 \\ -5 \end{bmatrix}$$

So, we should formulate the auxiliary problem and apply simplex algorithm (Phase I). We add the artificial variable y and calculate vector d:

$$d = -A_B e = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

The auxiliary LP that will be solved in Phase I is the following:

$$\begin{array}{lrcrcrcrcrcrcrcl}
\min & z = & & & & & & & & & & & & y & & \\
s.t. & & x_1 & + & x_2 & + & x_3 & + & x_4 & & & & - & y & = & 2 \\
& & 3x_1 & + & x_2 & - & x_3 & & & - & x_5 & & + & y & = & 3 \\
& & 3x_1 & + & 2x_2 & - & x_3 & & & & & - & x_6 & + & y & = & 5 \\
x_j, y \geq 0, & (j = 1,2,3,4,5,6) & & & & & & & & & & & & & &
\end{array}$$

The leaving variable is x_6:

$$min\left\{x_{B[4]}, x_{B[5]}, x_{B[6]}\right\} = min\left\{2, -3, -5\right\} = -5$$

We update the basic partition, $B = [4,5,7]$ and $N = [1,2,3,6]$. Finally, we update the needed matrices and vectors:

$$A_B^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x_B = A_B^{-1} b = \begin{bmatrix} 7 \\ 2 \\ 5 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$s_N = c_N^T - w A_N = \begin{bmatrix} -3 & -2 & 1 & 1 \end{bmatrix}$$

Next, we perform the optimality test. $s_N \ngeq 0$, so the current basic partition is not optimal. According to Dantzig's rule, the entering variable is x_1 since it has the most negative $\overline{s_l}$ (-3). Then, we calculate the pivoting column:

$$h_1 = A_B^{-1} A_{\cdot 1} = \begin{bmatrix} 4 \\ 0 \\ 3 \end{bmatrix}$$

Next, we perform the minimum ratio test:

$$min\left\{\frac{x_i}{h_{il}} : i \in B, h_{il} > 0\right\} = \frac{5}{3} = \frac{x_{B[7]}}{h_{31}}$$

The leaving variable is x_7. So, we update the basic partition, $B = [4, 5, 1]$ and $N = [7, 2, 3, 6]$. Next, we update the basis inverse using PFI:

$$E^{-1} = \begin{bmatrix} 1 & 0 & -4/3 \\ 0 & 1 & 0 \\ 0 & 0 & 1/3 \end{bmatrix}$$
$$\left(A_{\overline{B}}\right)^{-1} = E^{-1}A_B^{-1} = \begin{bmatrix} 1 & 0 & -1/3 \\ 0 & -1 & 1 \\ 0 & 0 & 1/3 \end{bmatrix}$$

Finally, we update the needed vectors:

$$x_B = A_B^{-1}b = \begin{bmatrix} 1/3 \\ 2 \\ 5/3 \end{bmatrix}$$
$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$
$$s_N = c_N^T - wA_N = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

We continue with the second iteration by performing the optimality test. $s_N \geq 0$, so the current basic partition is optimal for the auxiliary LP and feasible for the original problem. We delete the artificial variable from the nonbasic list, so the nonbasic list is $N = [2, 3, 6]$ and proceed to Phase II.

We start Phase II (third iteration) by calculating the needed vectors:

$$A_B^{-1} = \begin{bmatrix} 1 & 0 & -1/3 \\ 0 & -1 & 1 \\ 0 & 0 & 1/3 \end{bmatrix}$$
$$x_B = A_B^{-1}b = \begin{bmatrix} 1/3 \\ 2 \\ 5/3 \end{bmatrix}$$
$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 8/3 \end{bmatrix}$$
$$s_N = c_N^T - wA_N = \begin{bmatrix} -4/3 & -10/3 & 8/3 \end{bmatrix}$$

We start by performing the optimality test. $s_N \ngeq 0$, so the current basis is not optimal. According to Dantzig's rule, the entering variable is x_3 since it has the most negative $\overline{s_l}$ $(-10/3)$. The pivoting column is equal to:

$$h_3 = A_B^{-1} A_{.3} = \begin{bmatrix} 4/3 \\ 0 \\ -1/3 \end{bmatrix}$$

Next, we perform the minimum ratio test to find the leaving variable:

$$min\left\{ \frac{x_i}{h_{il}} : i \in B, h_{il} > 0 \right\} = \frac{1/3}{4/3} = \frac{x_{B[4]}}{h_{13}}$$

The leaving variable is x_4.

Next, we update the basic partition, $B = [3, 5, 1]$ and $N = [2, 4, 6]$, and the basis inverse using PFI:

$$E^{-1} = \begin{bmatrix} 3/4 & 0 & 0 \\ 0 & 1 & 0 \\ 1/4 & 0 & 1 \end{bmatrix}$$

$$\left(A_{\overline{B}}\right)^{-1} = E^{-1} A_B^{-1} = \begin{bmatrix} 3/4 & 0 & -1/4 \\ 0 & -1 & 1 \\ 1/4 & 0 & 1/4 \end{bmatrix}$$

Finally, we update the needed vectors:

$$x_B = A_B^{-1} b = \begin{bmatrix} 1/4 \\ 2 \\ 7/4 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} == \begin{bmatrix} 0 & 0 & 8/3 \end{bmatrix}$$

$$s_N = c_N^T - w A_N = \begin{bmatrix} -1/2 & 5/2 & 7/2 \end{bmatrix}$$

We start the fourth iteration by performing the optimality test. $s_N \ngeq 0$, so the current basis is not optimal. According to Dantzig's rule, the entering variable is x_2 since it has the most negative $\overline{s_l}$ $(-1/2)$. The pivoting column is equal to:

$$h_2 = A_B^{-1} A_{.2} = \begin{bmatrix} 1/4 \\ 1 \\ 3/4 \end{bmatrix}$$

Next, we perform the minimum ratio test to find the leaving variable:

$$min\left\{ \frac{x_i}{h_{il}} : i \in B, h_{il} > 0 \right\} = \frac{1/4}{1/4} = \frac{x_{B[3]}}{h_{12}}$$

The leaving variable is x_3.

Next, we update the basic partition, $B = [2, 5, 1]$ and $N = [3, 4, 6]$, and the basis inverse using PFI:

$$E^{-1} = \begin{bmatrix} 4 & 0 & 0 \\ -4 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

$$\left(A_{\overline{B}}\right)^{-1} = E^{-1}A_B^{-1} = \begin{bmatrix} 3 & 0 & -1 \\ -3 & -1 & 2 \\ -2 & 0 & 1 \end{bmatrix}$$

Finally, we update the needed vectors:

$$x_B = A_B^{-1}b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} == \begin{bmatrix} 0 & 0 & 9/8 \end{bmatrix}$$

$$s_N = c_N^T - wA_N = \begin{bmatrix} 2 & 4 & 4 \end{bmatrix}$$

Next, we start the fifth iteration by perform the optimality test. $s_N \geq 0$, so the current basic solution is optimal. As a result, we calculate the solution vector and the value of the objective function:

$$x = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$z = c_B^T x_B = 12$$

3.3 Exterior Point Simplex Algorithm

Since the development of the simplex method, various papers presented variants of the simplex method that relax feasibility requirements. A simplex-type algorithm generating solutions that are not feasible is called an exterior point simplex algorithm (EPSA) [92]. Dantzig's [27] parametric self-dual algorithm, Kuhn's [66] Hungarian method for the assignment problem, Iri's [57] successive shortest part method for minimum cost flow problems, Zionts' [122] and Terlaky's [110] criss-cross methods are some examples of exterior point methods. However, as all these methods are completely combinatorial,

they are not very efficient in practice for solving LPs. The first efficient implementation of an EPSA for solving LPs was proposed by Paparrizos [92].

EPSA was proposed by Paparrizos initially for the assignment problem [91] and then for the solution of LPs [92]. The geometry of the exterior point simplex algorithms shows that this algorithm is more efficient than the simplex algorithm [94]. The key idea of the exterior point simplex algorithm is based on making steps in directions that are linear combinations of attractive descent directions that can lead to fast convergence to the optimal solution. Two paths are created by the exterior point simplex algorithm; the first one consists of basic but not feasible solutions and the second path is feasible. A more effective approach proposed by Samaras [105]. Samaras transformed the exterior path into a dual feasible simplex path. This algorithm is called Primal-Dual Exterior Point Simplex Algorithm (PDEPSA). Its revised form was presented in [93]. A more effective approach is the Primal-Dual Interior Point Simplex Algorithm (PDIPSA) [41]. PDIPSA is able to deal with stalling and cycling more effectively than the other exterior point simplex algorithms. The key advantage of PDIPSA is that it uses an interior point in order to compute the leaving variable in contrast to primal-dual exterior point algorithms that use a boundary point. PDEPSA and PDIPSA are further discussed in Chapter 5.

Ploskas et al. [95] proposed a parallel implementation of an exterior point simplex algorithm showing promising results. Ploskas & Samaras [101] proposed a GPU-based implementation of PDEPSA achieving a maximum speedup of 181 on dense LPs and 20 on sparse LPs over MATLAB's interior point method. In addition, the GPU-based implementation was 2.3× faster than MATLAB's interior point method on a set of benchmark LPs.

In the rest of this Section, we present the exterior point simplex algorithm. Prior to the application of the simplex algorithm, a presolve and a scaling routine should be executed to reduce the problem's dimensions and improve the computational properties of the constraint matrix A (for more details, see Chapter 4). Then, an initial basis should be calculated in order to initialize the simplex algorithm. As already presented, we use a Gauss-Jordan elimination with partial pivoting that calculates a row echelon form of the constraint matrix.

Initially, we should determine if the direction d_B crosses the feasible region using the following relation:

$$\beta = max\left\{\frac{x_i}{-d_i} : i \in B, x_i < 0\right\} < \alpha = min\left\{\frac{x_i}{-d_i} : \ i \in B, d_i < 0\right\}$$

where $1 \leq i \leq m$, $d_B = -\sum_{j \in P} \lambda_j h_j$, $P = \{j \in N : s_j < 0\}$, $h_j = A_B^{-1} A_{.j}$, and λ is an arbitrary vector such that $\lambda = \left(\lambda_1, \lambda_2, \cdots, \lambda_{|P|}\right) > 0$. If $P \neq \emptyset$ and $\beta < \alpha$, then the direction d_B crosses the feasible region, so we apply the exterior point simplex algorithm to the original problem (Phase II). If $P = \emptyset$ or the direction d_B does not cross the feasible region, then we formulate an auxiliary problem to find an initial feasible basis (Phase I) using the revised simplex algorithm. Note, that it is possible to also use the exterior point simplex algorithm in Phase I.

As already presented in the previous Section, we use the two-phase method along with the single artificial technique to formulate the following auxiliary LP in Phase I:

$$\begin{aligned} \min \quad & e^T y \\ s.t. \quad & Ax + I_m y = b \\ & x, y \geq 0 \end{aligned}$$

where $e \in \mathbb{R}^n$ is a vector of ones. The auxiliary LP is solved using the simplex algorithm, as shown in Section 3.2.

While performing the optimality test for Phase I, we find either a feasible basis for the original problem or that the original problem is infeasible. In the first case (feasible basis), we continue with Phase II by applying the exterior point simplex algorithm to the original problem. We start by initializing the needed vectors and matrices:

$$A_B^{-1},\ x_B = A_B^{-1} b,\ w = c_B^T A_B^{-1}, \text{ and } s_N = c_N^T - w A_N$$

Next, we calculate sets P and Q:

$$P = \{j \in N : s_j < 0\},\ Q = \{j \in N : s_j \geq 0\}$$

Moreover, we define an arbitrary vector $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_{|P|}) > 0$ and compute s_0 as follows:

$$s_0 = \sum_{j \in P} \lambda_j s_j$$

and the direction $d_B = -\sum_{j \in P} \lambda_j h_j$, where $h_j = A_B^{-1} A_P$.

Next, we perform the optimality test of Phase II. The LP is optimal when one of the following conditions is true:

- Set P is empty

- $d_B \geq 0$ and $s_0 = 0$

In addition, if $d_B \geq 0$ and $s_0 < 0$, then the problem is unbounded. Otherwise, we continue with the remaining steps of Phase II. We select the leaving variable k using the following relation:

$$a = \frac{x_{B[r]}}{-d_{B[r]}} = min\left\{\frac{x_i}{-d_i} : i \in B, d_i < 0\right\}$$

If $a = \infty$, the problem is unbounded. Otherwise, the following vectors are calculated:

$$H_{rP} = \left(A_B^{-1}\right)_{r.} A_P \text{ and } H_{rQ} = \left(A_B^{-1}\right)_{r.} A_Q$$

Then, we compute ratios θ_1 and θ_2:

$$\begin{aligned} \theta_1 = \frac{-s_P}{H_{rP}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} > 0 \wedge j \in P\right\} \text{ and} \\ \theta_2 = \frac{-s_Q}{H_{rQ}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} < 0 \wedge j \in Q\right\} \end{aligned}$$

In addition, we determine indices t_1 and t_2 such that $P[t_1] = p$ and $Q[t_2] = q$. If $\theta_1 \leq \theta_2$ then $l = p$ and we update sets P and Q, $P = P \setminus [l]$ and $Q = Q \cup [k]$. Otherwise, $l = q$ and we update only set Q, $Q = Q \cup [k] \setminus [l]$. We also update the basic partition, $B = B \cup [l] \setminus [k]$ and $N = P \cup Q$.

Finally, we update the basis inverse using the PFI method:

$$\left(A_{\overline{B}}\right)^{-1} = (A_B E)^{-1} = E^{-1}(A_B)^{-1} \tag{3.4}$$

where E^{-1} is the inverse of the eta-matrix and can be computed by the the following equation:

$$E^{-1} = I - \frac{1}{h_{rl}}\left(h_l - e_l\right)e_l^T = \begin{bmatrix} 1 & & -h_{1l} & & \\ & \ddots & \vdots & & \\ & & 1/h_{rl} & & \\ & & \vdots & \ddots & \\ & & -h_{ml}/h_{rl} & & 1 \end{bmatrix} \tag{3.5}$$

Note again that the basis inverse is calculated from scratch every 80 iterations. Finally, we update vectors x_B, w, s_N and d_B:

$$x_B = A_B^{-1}b,\ w = c_B^T A_B^{-1},\ s_N = c_N^T - wA_N, \text{ and } d_b = E^{-1}d_B$$

If $\theta_1 \leq \theta_2$, then $\overline{d_{B(r)}} = d_{B(r)} + \lambda_l$.

We start again with the optimality test and all other steps presented previously for Phase II until we find an optimal basis for the LP or we find that the problem is unbounded.

A formal description of the exterior point simplex algorithm is given in Table 3.2.

Next, we will demonstrate the exterior point simplex algorithm with an example. The linear programming problem that will be solved is the following:

$$\begin{array}{lrrcrcrcl}
\min & z = & 8x_1 & + & 4x_2 & - & 6x_3 & & \\
s.t. & & x_1 & + & x_2 & + & x_3 & \leq & 2 \\
 & & 3x_1 & + & x_2 & - & x_3 & \geq & 3 \\
 & & 3x_1 & + & 2x_2 & - & x_3 & \geq & 5 \\
x_j \geq 0, & (j = 1, 2, 3) & & & & & & &
\end{array}$$

Let's convert the LP in its standard form:

$$\begin{array}{lrrcrcrcrcrcl}
\min & z = & 8x_1 & + & 4x_2 & - & 6x_3 & & & & & & \\
s.t. & & x_1 & + & x_2 & + & x_3 & + & x_4 & & & = & 2 \\
 & & 3x_1 & + & x_2 & - & x_3 & - & x_5 & & & = & 3 \\
 & & 3x_1 & + & 2x_2 & - & x_3 & & & - & x_6 & = & 5 \\
x_j \geq 0, & (j = 1, 2, 3, 4, 5, 6) & & & & & & & & & & &
\end{array}$$

So, the matrices and vectors that will be given as input to exterior point simplex algorithm are the following:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 3 & 1 & -1 & 0 & -1 & 0 \\ 3 & 2 & -1 & 0 & 0 & -1 \end{bmatrix}, \; c = \begin{bmatrix} 8 \\ 4 \\ -6 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; b = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$$

We start with the basic partition $B = [4, 5, 6]$ and $N = [1, 2, 3]$. The initial basis is not feasible ($P \neq \emptyset$ but $\beta > \alpha$):

$$A_B^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

TABLE 3.2: Exterior Point Simplex Algorithm

Step 0. *(Initialization).*
Presolve and scale the LP.
Select an initial basic partition (B, N).
Calculate set $P = \{j \in N : s_j < 0\}$.
If $P \neq \emptyset$ and direction d_B crosses the feasible region, proceed to Step 2.
Step 1. *(Phase I).*
Construct an auxiliary problem by adding an artificial variable y with a coefficient vector equal to $-A_B e$.
Apply revised simplex algorithm in the auxiliary problem.
If the final basic solution (B, N) is feasible, proceed to Step 2.
Else the problem is infeasible.
Step 2. *(Phase II).*
Step 2.0. *(Initialization).*
Compute $(A_B)^{-1}$ and vectors x_B, w and s_N.
Calculate sets $P = \{j \in N : s_j < 0\}$ and $Q = \{j \in N : s_j \geq 0\}$.
Define $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_{|P|}) > 0$ and compute s_0 and d_B:
$s_0 = \sum_{j \in P} \lambda_j s_j$ and the direction $d_B = -\sum_{j \in P} \lambda_j h_j$
where $h_j = A_B^{-1} A_{.j}$.
Step 2.1. *(Test of Optimality).*
If $P = \emptyset$ then the problem is optimal.
Else
  If $d_B \geq 0$ then
    If $s_0 = 0$ then the problem is optimal.
  Else choose the leaving variable $x_{B[r]} = x_k$ using the following relation:
  $a = \frac{x_{B[r]}}{-d_{B[r]}} = min\left\{\frac{x_i}{-d_i} : i \in B, d_i < 0\right\}$
  If $a = \infty$ then the problem is unbounded.
Step 2.2. *(Pivoting).*
Compute the vectors: $H_{rP} = \left(A_B^{-1}\right)_{r.} A_P$ and $H_{rQ} = \left(A_B^{-1}\right)_{r.} A_Q$.
Compute the ratios θ_1 and θ_2:
$\theta_1 = \frac{-s_P}{H_{rP}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} > 0 \wedge j \in P\right\}$ and
$\theta_2 = \frac{-s_Q}{H_{rQ}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} < 0 \wedge j \in Q\right\}$
Calculate the indices t_1 and t_2 such that $P[t_1] = p$ and $Q[t_2] = q$.
If $\theta_1 \leq \theta_2$ then set $l = p$
Else set $l = q$.
Step 2.3. *(Update).*
Swap indices k and l. Update the new basis inverse $\left(A_{\overline{B}}\right)^{-1}$, using PFI.
Update sets P and Q. Set $B(r) = l$.
If $\theta_1 \leq \theta_2$, set $P = P \setminus [l]$ and $Q = Q \cup [k]$.
Else set $Q(t_2) = k$.
Update vectors x_B, w s_N and d_B. using the relation $\overline{d_B} = E^{-1} d_B$.
If $l \in P$, set $\overline{d_{B(r)}} = d_{B(r)} + \lambda_l$.
Go to Step 2.1.

$$x_B = A_B^{-1}b = \begin{bmatrix} 2 \\ -3 \\ -5 \end{bmatrix}$$
$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$
$$s_N = c_N^T - wA_N = \begin{bmatrix} 8 & 4 & -6 \end{bmatrix}$$
$$P = \{j \in N : s_j < 0\} = [3]$$
$$h_j = A_B^{-1}A_P = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$
$$\lambda = (\lambda_1, \lambda_2, \lambda_3) = (1, 1, 1),\ d_B = -\sum_{j \in P} \lambda_j h_j = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$
$$\beta = max \left\{ \tfrac{x_i}{-d_i} : i \in B, x_i < 0 \right\} = -3$$
$$\alpha = min \left\{ \tfrac{x_i}{-d_i} : i \in B, d_i < 0 \right\} = -5$$

So, we should formulate the auxiliary problem and apply the revised simplex algorithm (Phase I). We add the artificial variable y and calculate vector d:

$$d = -A_B e = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

The auxiliary LP that will be solved in Phase I is the following:

$$\begin{array}{lrcrcrcrcrcrcrcl}
\min & z = & & & & & & & & & & & & y & & \\
s.t. & & x_1 & + & x_2 & + & x_3 & + & x_4 & & & & - & y & = & 2 \\
& & 3x_1 & + & x_2 & - & x_3 & & & - & x_5 & & + & y & = & 3 \\
& & 3x_1 & + & 2x_2 & - & x_3 & & & & & - & x_6 & + & y & = & 5 \\
x_j, y \geq 0, & (j = 1, 2, 3, 4, 5, 6) & & & & & & & & & & & & & &
\end{array}$$

The leaving variable is x_6:

$$min \left\{ x_{B[4]}, x_{B[5]}, x_{B[6]} \right\} = min \left\{ 2, -3, -5 \right\} = -5$$

We update the basic partition, $B = [4, 5, 7]$ and $N = [1, 2, 3, 6]$. Finally, we update the needed matrices and vectors:

$$A_B^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & -1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x_B = A_B^{-1} b = \begin{bmatrix} 7 \\ 2 \\ 5 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$s_N = c_N^T - w A_N = \begin{bmatrix} -3 & -2 & 1 & 1 \end{bmatrix}$$

Next, we perform the optimality test. $s_N \not\geq 0$, so the current basic partition is not optimal. According to Dantzig's rule, the entering variable is x_1 since it has the most negative $\overline{s_l}$ (-3). Then, we calculate the pivoting column:

$$h_1 = A_B^{-1} A_{\cdot 1} = \begin{bmatrix} 4 \\ 0 \\ 3 \end{bmatrix}$$

Next, we perform the minimum ratio test:

$$min \left\{ \frac{x_i}{h_{il}} : i \in B, h_{il} > 0 \right\} = \frac{5}{3} = \frac{x_{B[7]}}{h_{31}}$$

The leaving variable is x_7. So, we update the basic partition, $B = [4, 5, 1]$ and $N = [7, 2, 3, 6]$. Next, we update the basis inverse using PFI:

$$E^{-1} = \begin{bmatrix} 1 & 0 & -4/3 \\ 0 & 1 & 0 \\ 0 & 0 & 1/3 \end{bmatrix}$$

$$\left(A_{\overline{B}}\right)^{-1} = E^{-1} A_B^{-1} = \begin{bmatrix} 1 & 0 & -1/3 \\ 0 & -1 & 1 \\ 0 & 0 & 1/3 \end{bmatrix}$$

Finally, we update the needed vectors:

$$x_B = A_B^{-1} b = \begin{bmatrix} 1/3 \\ 2 \\ 5/3 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$s_N = c_N^T - w A_N = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

We continue with the second iteration by performing the optimality test. $s_N \geq 0$, so the current basic partition is optimal for the auxiliary LP and feasible for the original problem. We delete the artificial variable from the nonbasic list, so the nonbasic list is $N = [2, 3, 6]$ and proceed to Phase II.

We start Phase II (third iteration) by calculating the needed vectors:

$$A_B^{-1} = \begin{bmatrix} 1 & 0 & -1/3 \\ 0 & -1 & 1 \\ 0 & 0 & 1/3 \end{bmatrix}$$
$$x_B = A_B^{-1} b = \begin{bmatrix} 1/3 \\ 2 \\ 5/3 \end{bmatrix}$$
$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 8/3 \end{bmatrix}$$
$$s_N = c_N^T - w A_N = \begin{bmatrix} -4/3 & -10/3 & 8/3 \end{bmatrix}$$

Next, we calculate sets P and Q, $P = \{j \in N : s_j < 0\} = [2, 3]$ and $Q = N \setminus P = [6]$. Then, we calculate the direction d_B:

$$h_j = B^{-1} A_P = \begin{bmatrix} 1/3 & 4/3 \\ 1 & 0 \\ 2/3 & -1/3 \end{bmatrix}$$
$$d_B = -\sum_{j \in P} \lambda_j h_j = \begin{bmatrix} -5/3 \\ -1 \\ -1/3 \end{bmatrix}$$

$P \neq \emptyset$, so the current base is not optimal. $d_B \not\geqslant 0$, so we continue to find the leaving variable using the following relation:

$$a = min \left\{ \tfrac{x_i}{-d_i} : i \in B, d_i < 0 \right\} = \tfrac{1/3}{5/3}$$

The leaving variable is x_4. Then, we compute vectors H_{rP} and H_{rQ}:

$$H_{rP} = \left(A_B^{-1}\right)_{r.} A_P = \begin{bmatrix} 1/3 & 4/3 \end{bmatrix}$$
$$H_{rQ} = \left(A_B^{-1}\right)_{r.} A_Q = \begin{bmatrix} 1/3 \end{bmatrix}$$

Next, we compute the ratios θ_1 and θ_2:

$$\theta_1 = \frac{-s_P}{H_{rP}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} > 0 \wedge j \in P\right\} = \frac{10/3}{4/3}$$
$$\theta_2 = \frac{-s_Q}{H_{rQ}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} < 0 \wedge j \in Q\right\} = \infty$$

$\theta_1 \leq \theta_2$, so the leaving variable is x_3. We update sets P and Q, $P = [2]$, $Q = [6, 4]$, and the basic partition, $B = [3, 5, 1]$, $N = [2, 6, 4]$. We continue calculating the pivoting column:

$$h_3 = A_B^{-1}A_{.3} = \begin{bmatrix} 4/3 \\ 0 \\ -1/3 \end{bmatrix}, h_3(1) = -1, h_3 = \begin{bmatrix} -1 \\ 0 \\ -1/3 \end{bmatrix}$$

Then, we update the basis inverse using PFI:

$$E^{-1} = \begin{bmatrix} 3/4 & 0 & 0 \\ 0 & 1 & 0 \\ 1/4 & 0 & 1 \end{bmatrix}$$
$$\left(A_{\overline{B}}\right)^{-1} = E^{-1}A_B^{-1} = \begin{bmatrix} 3/4 & 0 & -1/4 \\ 0 & -1 & 1 \\ 1/4 & 0 & 1/4 \end{bmatrix}$$

Finally, we update the needed vectors:

$$x_B = A_B^{-1}b = \begin{bmatrix} 1/4 \\ 2 \\ 7/4 \end{bmatrix}$$
$$w = c_B^T A_B^{-1} = \begin{bmatrix} -5.2 & 0 & 7.2 \end{bmatrix}$$
$$s_N = c_N^T - wA_N = \begin{bmatrix} -4/2 & -10/3 & 8/3 \end{bmatrix}$$
$$d_B = E^{-1}d_B = \begin{bmatrix} -5/4 \\ -1 \\ -3/4 \end{bmatrix}, d_B(1) = d_B(1) + 1 = -5/4 + 1 = -1/4$$

We start the fourth iteration by performing the optimality test. $P \neq \emptyset$, so the current base is not optimal. $d_B \ngeq 0$, so we continue to find the leaving variable using the following relation:

$$a = min\left\{\frac{x_i}{-d_i} : i \in B, , d_i < 0\right\} = \frac{1/4}{1/4}$$

The leaving variable is x_3. Then, we compute vectors H_{rP} and H_{rQ}:

$$H_{rP} = \left(A_B^{-1}\right)_{r.} A_P = \begin{bmatrix} 1/4 \end{bmatrix}$$
$$H_{rQ} = \left(A_B^{-1}\right)_{r.} A_Q = \begin{bmatrix} 1/4 & 3/4 \end{bmatrix}$$

Next, we compute the ratios θ_1 and θ_2:

$$\theta_1 = \frac{-s_P}{H_{rP}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} > 0 \wedge j \in P\right\} = \frac{1/2}{1/4}$$
$$\theta_2 = \frac{-s_Q}{H_{rQ}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} < 0 \wedge j \in Q\right\} = \infty$$

$\theta_1 \leq \theta_2$, so the leaving variable is x_2. We update sets P and Q, $P = \emptyset$, $Q = [6, 4, 3]$, and the basic partition, $B = [2, 5, 1]$, $N = [6, 4, 3]$. We continue calculating the pivoting column:

$$h_2 = A_B^{-1} A_{.3} = \begin{bmatrix} 1/4 \\ 1 \\ 3/4 \end{bmatrix}, h_2(1) = -1, h_2 = \begin{bmatrix} -1 \\ 1 \\ 3/4 \end{bmatrix}$$

Then, we update the basis inverse using PFI:

$$E^{-1} = \begin{bmatrix} 4 & 0 & 0 \\ -4 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$
$$\left(A_{\overline{B}}\right)^{-1} = E^{-1} A_B^{-1} = \begin{bmatrix} 3 & 0 & -1 \\ -3 & -1 & 2 \\ -2 & 0 & 1 \end{bmatrix}$$

Finally, we update the needed vectors:

$$x_B = A_B^{-1} b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$
$$w = c_B^T A_B^{-1} = \begin{bmatrix} -4 & 0 & 4 \end{bmatrix}$$
$$s_N = c_N^T - w A_N = \begin{bmatrix} -4/3 & -10/3 & 8/3 \end{bmatrix}$$
$$d_B = E^{-1} d_B = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, d_B(1) = d_B(1) + 1 = -1 + 1 = 0$$

Next, we start the fifth iteration by performing the optimality test. $P = \emptyset$, so the current basic solution ($B = [2, 5, 1]$, $N = [6, 4, 3]$) is optimal. As a result, we calculate the solution vector and the value of the objective function:

$$x = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$
$$z = c_B^T x_B = 12$$

3.4 Interior Point Methods

Since Dantzig's initial contribution [25], many researchers have made efforts to enhance the performance of the simplex algorithm. Researchers proposed interior point methods (IPMs) that traverse across the interior of the feasible region [36] [53] [117]. However, the proposed algorithms are not efficient in practice due to the very expensive time per iteration and numerical difficulties. Later, Khachiyan [63] proposed the ellipsoid method, the first polynomial algorithm for LP. However, this method was not efficient due to very expensive time per iteration. In 1984, a totally new method arose and changed everything in LP; Karmarkar's method [60] was the first efficient IPM. Since then, many IPMs have been proposed [43] [44] [79] [73] (for a literature review, see [46] [112] [120]).

As its name indicates, an IPM moves through the interior of the feasible region towards the optimal solution; this is the big difference with the simplex algorithm, which follows a sequence of adjacent boundary points to the optimal solution. It has been observed that IPMs can deal much better than the simplex algorithm in large-scale sparse LPs [46]; these problems are very common in transportation and scheduling applications that have network models at their core. IPMs are also of interest from a theoretical point of view, because they have polynomial complexity. There are three main categories of IPMs: (i) affine-scaling methods, (ii) potential reduction methods, and (iii) central trajectory methods. The affine-scaling algorithm is an attractive choice due to its simplicity and its relative good performance in practice. However, its performance is sensitive to the starting point. Potential reduction methods do not have the simplicity of affine-scaling methods, but they are more attractive than affine-scaling methods. IPMs based on the central trajectory are the most useful in theory and the most used in practice.

The main advantages of IPMs in comparison to the simplex algorithm are: (i) the number of iterations is not related with the number of vertices, and (ii) IPMs are not influenced by degeneracies. On the other hand, IPMs have some significant weaknesses: (i) it has been observed that IPMs are not very effective to detect infeasibility or unboundedness in some cases, and (ii) although IPMs have a fast convergence to optimal solution in the first iterations, they present a very slow rate of convergence in the later iterations.

The primal–dual path following algorithm is an example of an IPM that operates simultaneously on the primal and dual LPs. Moreover, the primal–dual algorithms that incorporate predictor and corrector steps are the most efficient IPMs. In the rest of this Section, we give a brief overview of the basic concepts of primal-dual IPMs and then we describe Mehrotra's Predictor-Correct method that we use in our implementations when an IPM is needed.

As already presented in Chapter 2, solving the primal LP is equivalent of solving the dual LP. The primal and dual LPs are referred as the primal-dual pair. According to the duality theory, if x and (w, s) are feasible for the primal and dual LPs, respectively, then $b^T w \leq c^T x$. The duality gap ($|c^T x - b^T w|$) is the difference between the objective function of the primal LP and the dual LP.

Let $u = (x, w, s)$ be an interior point with $x > 0$ and $s > 0$. If the interior point is not optimal, we can apply Newton's method with a fixed barrier parameter $\mu > 0$ to obtain a new interior point $\bar{u}$.

The dual residual (r_d), the primal residual (p_d) and the complementarity residual (r_c) at u are calculate as:

$$\begin{aligned} r_d &= A^T w + s - c \\ r_p &= Ax - b \\ r_c &= Xs - \mu e \end{aligned} \tag{3.6}$$

Using also Karush-Kuhn-Tucker (KKT) conditions for the primal-dual pair, we can formulate the following linear system:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta s \end{bmatrix} = \begin{bmatrix} r_d \\ r_p \\ r_c \end{bmatrix} \tag{3.7}$$

The solution of the above system gives as the solution:

$$\begin{aligned} \Delta w &= (AXS^{-1}A^T)^{-1}(r_p - AS^{-1}(r_c - Xr_d)) \\ \Delta s &= r_d - A^T \Delta w \\ \Delta x &= S^{-1}(r_c - X\Delta s) \end{aligned} \tag{3.8}$$

Solving the above system, we can calculate the next interior point. Then, we continue applying the Newton method until a termination criterion is satisfied. Next, we describe Mehrotra's Predictor-Corrector method.

Prior to the application of the simplex algorithm, a presolve and a scaling routine should be executed to reduce the problem's dimensions and improve the computational properties of the constraint matrix A (for more details, see Chapter 4). After that, we need to calculate an initial interior point. Most primal-dual IPMs need a strictly feasible interior point as a starting point. However, Mehrotra's Predictor-Corrector method is an infeasible primal-dual IPM and it just requires an interior point $(x^0, s^0) > 0$ for the starting point. We calculate a new interior point (x, w, s) in each iteration. This point may be infeasible with $(x, s) > 0$.

Mehrotra proposed a heuristic to obtain a starting interior point:

$$\begin{gathered} \bar{x} = A^T(AA^T)^{-1}b, \bar{w} = (AA^T)^{-1}Ac, \bar{s} = c - A^T\bar{w} \\ \delta x = max(-1.5min(\bar{x}), 0), \delta s = max(-1.5min(\bar{s}), 0) \\ \bar{\delta}_x = \delta_x + 0.5\frac{(\bar{x}+\delta_x e)^T(\bar{s}+\delta s e)}{\sum_{i=1}^{n}\bar{s}_i+\delta_s}, \bar{\delta}_s = \delta_s + 0.5\frac{(\bar{x}+\delta_x e)^T(\bar{s}+\delta s e)}{\sum_{i=1}^{n}\bar{x}_i+\delta_x} \\ x^0 = \bar{x} + \bar{\delta}_x e, w^0 = \bar{w}, s^0 = \bar{s} + \bar{\delta}_s e \end{gathered} \tag{3.9}$$

After finding the starting point, we continue performing the steps of Mehrotra's Predictor-Method, described below, until a termination criterion is satisfied. The termination criterion is the following:

$$max(\mu, ||Ax - b||, ||A^T w + s - c||) \leq tol \tag{3.10}$$

where tol is the tolerance.

As the name reveals, Mehrotra's Predictor-Corrector method uses two direction, the predictor and the corrector. In the predictor step, the following system of equations is solved:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^p \\ \Delta w^p \\ \Delta s^p \end{bmatrix} = \begin{bmatrix} A^T w + s - c \\ Ax - b \\ Xs \end{bmatrix} = \begin{bmatrix} r_d \\ r_p \\ r_c \end{bmatrix} \tag{3.11}$$

where $(\Delta x^p, \Delta w^p, \Delta s^p)$ is the Newton direction. Then, we calculate the largest step lengths $\alpha_p^p, \alpha_d^p \in (0, 1]$:

$$x(\alpha_p^p) = x - \alpha_p^p \Delta x^p, s(\alpha_d^p) = s - \alpha_d^p \Delta s^p \geq 0 \tag{3.12}$$

where:

$$\alpha_p^p = min\left\{1, min_{\Delta x_i^p > 0} \frac{x_i}{\Delta x_i^p}\right\}, \alpha_d^p = min\left\{1, min_{\Delta s_i^p > 0} \frac{s_i}{\Delta s_i^p}\right\} \tag{3.13}$$

Next, we compute the centering parameter σ:

$$\sigma = \left(\frac{\left(x - \alpha_p^p \Delta x^p\right)^T \left(s - \alpha_d^p \Delta s^p\right)}{n\mu}\right)^3 \tag{3.14}$$

where $\mu = x^T s/n$.

In the corrector step, the following system of equations is solved:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta s \end{bmatrix} = \begin{bmatrix} A^T w + s - c \\ Ax - b \\ Xs - \sigma\mu e \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Delta X^p \Delta s^p \end{bmatrix} \tag{3.15}$$

As the coefficient matrix is the same in the corrector and predictor step, we can use only one factorization at its iteration.

Next, we calculate the primal and dual steps lengths:

$$\alpha_p = min\left\{1, \eta min_{\Delta x_i > 0} \frac{x_i}{\Delta x_i}\right\}, \alpha_d = min\left\{1, \eta min_{\Delta s_i > 0} \frac{s_i}{\Delta s_i}\right\} \tag{3.16}$$

where $\eta \in (0, 1)$ is a value close to 1.

We also update the interior point:

$$\begin{aligned} \bar{x} &= x - \alpha_p \Delta x \\ \bar{w} &= w - \alpha_d \Delta w \\ \bar{s} &= s - \alpha_d \Delta s \end{aligned} \tag{3.17}$$

Finally, we perform the termination test to find out if the solution satisfies the termination criterion or we should continue performing the next iteration of the method.

A formal description of the MPC method is given in Table 3.3.

Next, we will demonstrate Mehrotra's Predictor-Corrector method with an example. Since this method includes a factorization and the solution of two systems of equations, it is out of scope of this thesis to present the exact steps of the algorithm. We will only focus on how the interior point is updated and we will demonstrate the convergence of the algorithm to the optimal solution.

TABLE 3.3: Mehrotra's Predictor-Corrector Method

Step 0. *(Initialization).*
Presolve and scale the LP.
Find an initial interior point (x^0, w^0, s^0) using Mehrotra's heuristic (3.9).
Step 1. *(Termination Criterion).*
Calculate the primal (r_p), dual (r_d) and complementarity (r_c) residuals.
Calculate the duality measure (μ).
If $max(\mu, ||r_p||, ||r_d||) \leq tol$ then the problem is optimal.
Step 2. *(Predictor Step).*
Solve the system (3.11).
Step 3. *(Centering Parameter Step).*
Compute the centering parameter σ.
Step 4. *(Corrector Step).*
Solve the system (3.15).
Step 5. *(Update Step).*
Update the solution (x, w, s).
Go to Step 1.

The LP that will be solved is the following:

$$\begin{array}{llrcrcrcl} \min & z = & 8x_1 & + & 4x_2 & - & 6x_3 & & \\ s.t. & & x_1 & + & x_2 & + & x_3 & \leq & 2 \\ & & 3x_1 & + & x_2 & - & x_3 & \geq & 3 \\ & & 3x_1 & + & 2x_2 & - & x_3 & \geq & 5 \\ x_j \geq 0, & (j = 1, 2, 3) & & & & & & & \end{array}$$

Let's convert the LP in its standard form:

$$\begin{array}{llrcrcrcrcrcl} \min & z = & 8x_1 & + & 4x_2 & - & 6x_3 & & & & & & \\ s.t. & & x_1 & + & x_2 & + & x_3 & + & x_4 & & & = & 2 \\ & & 3x_1 & + & x_2 & - & x_3 & & & - & x_5 & = & 3 \\ & & 3x_1 & + & 2x_2 & - & x_3 & & & & - \; x_6 & = & 5 \\ x_j \geq 0, & (j = 1, 2, 3, 4, 5, 6) & & & & & & & & & & & \end{array}$$

So, the matrices and vectors that will be given as input to IPM are the following:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 3 & 1 & -1 & 0 & -1 & 0 \\ 3 & 2 & -1 & 0 & 0 & -1 \end{bmatrix}, \; c = \begin{bmatrix} 8 \\ 4 \\ -6 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; b = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$$

Using Mehrotra's heuristic (3.9), we can calculate an initial interior point:

$$x = \begin{bmatrix} 2.2719 \\ 2.4719 \\ 1.4719 \\ 1.6719 \\ 1.8719 \\ 0.8719 \end{bmatrix}, w = \begin{bmatrix} -1.3143 \\ 1.3905 \\ 1.7714 \end{bmatrix}, s = \begin{bmatrix} 3.4548 \\ 4.0071 \\ 2.1024 \\ 4.9405 \\ 5.0167 \\ 5.3976 \end{bmatrix}$$

In the first iteration, we perform all the steps described previously and we finally calculate a new interior point:

$$x = \begin{bmatrix} 1.1555 \\ 0.9558 \\ 0.0074 \\ 0.1138 \\ 1.2986 \\ 0.1964 \end{bmatrix}, w = \begin{bmatrix} -4.5975 \\ 0.8098 \\ 3.0787 \end{bmatrix}, s = \begin{bmatrix} 0.9319 \\ 1.6302 \\ 2.4860 \\ 4.5975 \\ 0.8098 \\ 3.0787 \end{bmatrix}$$

The objective value using this solution is 13.0229. Hence, we observe that Merhotra's Predictor-Corrector method with only one iteration reached very close to the optimal solution, which is 12.

In the second iteration, we perform all the steps described previously and we finally calculate a new interior point:

$$x = \begin{bmatrix} 1.0143 \\ 0.9826 \\ 0.0019 \\ 0.0081 \\ 1.0201 \\ 0.0010 \end{bmatrix}, w = \begin{bmatrix} -3.5950 \\ 0.0897 \\ 3.7486 \end{bmatrix}, s = \begin{bmatrix} 0.0802 \\ 0.0082 \\ 1.4333 \\ 3.5950 \\ 0.0897 \\ 3.7486 \end{bmatrix}$$

The objective value using this solution is 12.0332, even closer to the optimal solution, which is 12. Hence, we observe that Mehrotra's Predictor-Corrector method (and IPMs in general) has a very fast convergence to the optimal solution at the first iterations. However, the convergence is very slow at the last iterations. For example, Mehrotra's Predictor-Corrector method needs six iterations to find the solution to this linear programming problem.

Chapter 6 presents a hybrid algorithm that takes advantage of Mehrotra's Predictor-Corrector method fast convergence in the first iterations and combines it with an exterior point simplex algorithm, which is utilized in the last iterations.

3.5 Conclusions

This Chapter gave an overview of the linear programming algorithms used in this dissertation. The primal revised simplex algorithm is one of the most important LP algorithms and is used on many applications. We also presented the most well known simplex exterior point simplex algorithm. Moreover, we will build on this algorithm in the next Chapters and present other variants of this algorithm. We also presented Mehrotra's Predictor-Corrector method, an algorithm that we combine with the exterior point simplex algorithm in Chapter 6 in order to build a very efficient hybrid algorithm.

Chapter 4

Presolve Techniques and A New Method

4.1 Introduction

It is well known that preprocessing is of great value for linear programming and its significance has been proved in practice. The main reason for making the presence of preprocessing necessary is the numerical difficulties that software face while solving a large scale LP. In addition, some LPs are automatically created from modelling software. The use of modelling software includes both advantages and disadvantages. Modelling software packages are extremely useful and they simplified the use of linear programming techniques in a vast number of sectors for people who are not experts in linear programming. On the other hand, the simple ways of constructing a LP led to problems with larger dimensions and their form became more complicated for linear programming solvers.

Preprocessing can contribute significantly to reduce this gap between the problem form, which is produced from modelling software, and the form that a linear programming solver demands for an accurate solution within an acceptable time interval. The main goal of preprocessing is to construct a new equivalent LP with significant reduced dimensions. As it is obvious, presolve methods are all these techniques that are responsible for reforming a LP to an equivalent problem with less dimensions in order to enhance the solvers' computational performance.

As it is mentioned previously, preprocessing can improve the computational performance of a LP solver; the main goals of presolve methods are [82]:

- Reduction of the dimensions of LPs.

- Enhancement of some arithmetic and computational characteristics of LPs.
- Detection if a LP is infeasible or unbounded.
- Detection of specific characteristics and forms of LPs that are detected only when presolve methods are used.

The significant role of preprocessing was recognized from the first years of linear programming. In general, preprocessing includes techniques that can reduce the size of LPs and the number of non-zero elements. This can be achieved by eliminating redundant constraints and variables.

Consequently, for all these reasons, presolve analysis became a fundamental component of all well-known commercial and non-commercial linear programming packages which include many presolve techniques. The number of constraints and variables in the linear optimization problems are mainly those that are responsible for the increase or decrease of the computational complexity. Consequently, it is possible to achieve significant reductions to the size of the problem if presolve methods are applied prior to the execution of a linear programming algorithm. This reduction in the size of the LPs can result in the decrease of execution time that a linear programming solver demands in order to solve it.

Solving LPs efficiently has always been a challenge for Operations Research community. The wide range of industrial and scientific applications that requires fast and efficient linear programming solvers is the reason for the existence of preprocessing methods. These presolve techniques can be combined with simplex methods, interior point methods and exterior point algorithms.

Preprocessing can be applied in many different ways; other techniques focus on the bounds of variables and how they can be strengthened, while other procedures are implemented in order to eliminate redundant constraints of the initial LP. Consequently, a new equivalent LP is produced after a presolve method is applied. In this new LP, the same presolve techniques are applied consecutively until no changes are happened to LP. Furthermore, presolve procedures can be used by any linear programming solver independently of its algorithm. In contrast to the benefits, they are some crucial points, for example, in many cases the detection of redundant constraints and variables are extremely computationally demanding.

The structure of this Chapter is as follows. In Section 4.2, we present some useful information referring to the evolution of presolve techniques and their main categories. In Section 4.3 to 4.10, a complete presolve routine is presented, which consists of three different scaling techniques and eight presolve methods that aim to reduce the size of the

LP. Furthermore, in Section 4.11, a new presolve technique is introduced with significant computational results from the analysis of primal feasibility conditions. Computational results with a set of optimal benchmark problems from the NETLIB set are also presented in Section 4.12. The computational study includes statistics of benchmarks before the application of the presolve session, statistics after the application of the presolve session and the impact of the proposed new presolve technique. Finally, simplex algorithm has been used in order to solve benchmarks before and after the new presolve technique has been performed and its execution time and number of iterations to solve the LPs are reported. Finally, conclusions are presented in Section 4.13.

4.2 Background

Presolve procedures have arisen in the literature many decades ago and nowadays, they are one of the most fundamental parts in all linear programming solvers. Furthermore, previous researchers have discussed and studied thoroughly the use of presolve techniques in linear programming. The first approaches presented the idea of using general linear transformations on the constraints in order to reduce the number of non-zeros in the LPs.

Furthermore, presolve techniques are able to spot possible infeasibility and unboundedness of the problem. For all these reasons presolve analysis became a fundamental component of all the well-known commercial and non-commercial linear programming solvers. The number of constraints and variables in the linear optimization problems is mainly responsible for the increase or decrease of computational complexity [8].

In 1970, the preconditioning techniques were widely introduced through a paper of Reid [103] referring to large sparse matrices. When these matrices are reasonably well-conditioned, an iterative procedure was applied in order to reduce the steps performed by the algorithm to compute the linear solution. This paper inspired many researchers to be involved in this issue. Moreover, in the next years, researchers focused their work on row elimination and bound tightening in mathematical programming systems [74] [121].

There are many presolve techniques already in use in many linear programming solvers. Furthermore, previous researchers [34] [72] [82] [109] [119] have discussed and studied thoroughly the use of presolve techniques in linear programming. The first approaches presented the idea of using general linear transformations on the constraints in order to reduce the number of non-zeros in the LPs [2] [19] [77]. A recently new set of presolve techniques for LPs with box constraints is presented by Ioslovich [56].

Finally, a detailed survey of presolve techniques can be found in [5]. These presolve techniques can be combined with simplex methods, interior point methods and exterior point algorithms [45] [50] [105] [116] [118]. Moreover, there are many papers referring to the effective implementation of presolve procedures in linear programming algorithms [59] [18] [9], [39].

The presolve techniques can be divided into four categories. These categories are:

- Techniques derived from analysis of the primal feasibility conditions.
- Techniques derived from the dual feasibility conditions.
- Techniques derived from the complementarity conditions.
- Linear transformations performed on a set of equalities which aim to reduce the density of the constraint matrix A.

4.3 Scaling Techniques

Presolve procedures are well known as preconditioning techniques and one of the most well-known preconditioning techniques is scaling. The numerical accuracy is the main concern of LP solver at the same time of reducing the computational time. The problem of accuracy is most common on large scale problems and it stems from the class difference that the numerical values seem to include. Matrices and vectors with no such problems are known as well-scaled because they can reduce the number of iterations and at the same time they do not affect the correctness of the solution.

In this Section, three different scaling techniques are used, the equilibration method, the geometric mean and the hybrid technique which combines the first two methods. The equilibration and geometric mean methods are the two most widely-used scaling techniques. Apart from these three methods, there are many other scaling techniques [23] [37] [68] [30], however the equilibration method remains the winner according to measurements like solution time, scaling time, solution iterations and condition number [98] [99].

Consequently, it is obvious that scaling is implemented as a part of the presolve techniques and its main goals are [114]:

- To enhance the condition number of the constraint matrix A and the numerical properties of the LP.

- To enforce the algorithm achieve the proper tolerances which are essential for the numerical accuracy.
- To create more clear bounds of variables.
- To lead to significant reductions of computational time through the reduction of algorithm iterations.

4.3.1 Equilibration

In the equilibration technique, the main goal is to transform the constraint matrix A in a new matrix where all of its elements will have values between -1 and 1. The desirable result will be achieved if the constraint matrix A is scanned twice, first a row scan is performed and then a column scan. Each row is multiplied by the inverse of its largest element in absolute value.

Furthermore, the largest element in absolute value is spotted in each column and if this value is greater than 1, then the specific column is multiplied by the inverse of this element.

4.3.2 Geometric Mean

In geometric mean, the main goal is the variance reduction of the nonzero elements of the constraint matrix A. Similarly to the equilibration technique, the constraint matrix A is scanned twice. However, in the geometric mean technique, a column scan is performed initially. The geometric mean is calculated for each column and its inverse is multiplied by the elements of the specific column.

$$s_j = (\bar{X}_j \underline{X}_j)^{-1/2} : j \in N \tag{4.1}$$

Respectively, each row is multiplied by its geometric mean.

$$r_i = (\bar{X}_i \underline{X}_i)^{-1/2} : i \in M \tag{4.2}$$

4.3.3 Hybrid

Finally, in the hybrid technique, both the equilibration and the geometric mean are taking place. Firstly, the equilibration technique is performed and then the geometric mean technique.

4.4 Eliminate Empty Rows and Columns

A row of the constraint matrix A is an empty row if all the coefficients in that row are zeros.

$$\underline{b}_i \leq a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq \bar{b}_i \tag{4.3}$$

with $i = 1, 2, \ldots, m$ and $a_{ij} = 0, j = 1, 2, \ldots n$. A constraint of this type can be redundant or it can state that the LP is infeasible.

All possible cases are shown below:

1. $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq \bar{b}_i$ and $\bar{b}_i \geq 0$. The constraint is redundant.
2. $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq \bar{b}_i$ and $\bar{b}_i < 0$. The linear problem is infeasible.
3. $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq \underline{b}_i$ and $\underline{b}_i \leq 0$. The constraint is redundant.
4. $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq \bar{b}_i$ and $\bar{b}_i > 0$. The linear problem is infeasible.
5. $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq \underline{b}_i = \bar{b}_i = b_i$ and $b_i = 0$. The constraint is redundant.
6. $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq \underline{b}_i = \bar{b}_i = b_i$ and $b_i \neq 0$. The linear problem is infeasible.

A column of the constraint matrix A is an empty column if all of its elements are zeros. A variable of this type can be redundant or it can state that the LP is unbounded.

All possible cases are shown below:

1. $c_j \geq 0$. The variable (column) is redundant and it can be deleted.
2. $c_j < 0$. The linear problem is unbounded.

4.5 Eliminate Singleton Rows

A row of the constraint matrix A is a singleton row if the constraint is an equality and only one element in that row is non-zero. Using mathematical notation, the singleton row can be stated as follows:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = \underline{b}_i = \bar{b}_i \tag{4.4}$$

with only one $a_{ik} \neq 0 \wedge a_{ij} = 0, j \neq k$.

According to the above statement, the value of x_k corresponding to the column k is fixed at b_i/a_{ik}. Moreover, there are two cases in this occasion:

1. $x_k \geq 0$. In this case, the row i and the column k are redundant and they can be deleted.
2. $x_k < 0$. In this case, the current value is out of the bounds for the variable and the LP is infeasible.

The result of this presolve technique is the calculation of the value of one variable. After the replacement of this variable to all constraints, it is very often to arise a new singleton row; as a result, the current presolve technique should continue until no singleton row is found.

4.6 Eliminate Singleton Inequality Rows

A constraint with only one non-zero coefficient:

$$\underline{b}_i \leq a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq \bar{b}_i \tag{4.5}$$

with $i = 1, 2, \ldots, m$ and $k, j \in [1, 2, \ldots, n]$ is called a singleton inequality row. A constraint of this type can be redundant or it can state that the LP is infeasible. All possible cases are shown below.

If the constraint is of type $a_{ik}x_k \leq \bar{b}_i$ then there are the cases:

1. $a_{ik} > 0 \wedge \bar{b}_i < 0$. The LP is infeasible.
2. $a_{ik} < 0 \wedge \bar{b}_i > 0$. The constraint i is redundant and it can be eliminated.
3. $a_{ik} > 0 \wedge \bar{b}_i = 0$. The constraint i and the variable k are redundant and they can be eliminated.
4. $a_{ik} < 0 \wedge \bar{b}_i = 0$. The constraint i is redundant and it can be eliminated.

If the constraint is of type $a_{ik}x_k \geq \bar{b}_i$ then there are the cases:

1. $a_{ik} > 0 \wedge \bar{b}_i < 0$. The constraint i is redundant and it can be eliminated.

2. $a_{ik} < 0 \wedge \bar{b}_i > 0$. The LP is infeasible.
3. $a_{ik} > 0 \wedge \bar{b}_i = 0$. The constraint i is redundant and it can be eliminated.
4. $a_{ik} < 0 \wedge \bar{b}_i = 0$. The constraint i and the variable k are redundant and they can be eliminated.

4.7 Eliminate Dual Singleton Inequality Row

This technique is very similar with the previous. Initially, the dual LP is created. Next, the algorithm searches for singleton inequality rows. In other words, the previous technique is applied on the dual LP of the initial problem.

4.8 Eliminate Free Singleton Column

A constraint

$$a_{ij}x + a_{is}x_s = \bar{b}_i = \underline{b}_i = b \tag{4.6}$$

with $i = 1, 2, \ldots, m$ and $s \in [1, 2, \ldots, n]$in which there is a singleton column (only $a_{is} \neq 0$) with:

$$a_{is} > 0 \wedge a_{ij} \leq 0, j \neq s \tag{4.7}$$

or

$$a_{is} < 0 \wedge a_{ij} \geq 0, j \neq s \tag{4.8}$$

then the variable x_s can be deleted from the LP. In addition, the i^{th} constraint can be removed too.

4.9 Eliminate Redundant Bounds from Constraints

A constraint of the type:

$$\underline{b}_i \leq a_i x \leq \bar{b}_i \tag{4.9}$$

with the variables x, $\underline{x} \leq x \leq \bar{x}$ and $\underline{x} = 0, \bar{x} = +\infty$, calculates new bounds for the constraints. These new bounds can easily be computed by:

$$\underline{b_i}' = \underset{\underline{x} \leq x \leq \bar{x}}{inferior} \langle a_i x \rangle = \sum_{a_{ij} \geq 0} a_{ij} \underline{\mathrm{x}}_j + \sum_{a_{ij} \leq 0} a_{ij} \bar{x_j} \tag{4.10}$$

$$\bar{b_i}' = \underset{\underline{x} \leq x \leq \bar{x}}{superior} \langle a_i x \rangle = \sum_{a_{ij} \geq 0} a_{ij} \bar{x_j} + \sum_{a_{ij} \leq 0} a_{ij} \underline{\mathrm{x}}_j \tag{4.11}$$

The function inferior calculates the greatest from the inferior bounds and the function superior calculates the lowest from the superior bounds.

If $[\underline{b_i}', \bar{b_i}'] \cap [\underline{b_i}, \bar{b_i}] = \oslash$, then the LP is infeasible. On the other hand, if $[\underline{b_i}', \bar{b_i}'] \subseteq [\underline{b_i}, \bar{b_i}]$, then the i^{th} constraint is redundant and it can be eliminated.

4.10 Eliminate Linearly Dependent Rows

Two constraints a_i and a_k with $i \neq k$ are called linearly dependent if

$$a_i = \lambda a_k \tag{4.12}$$

with $\lambda \in \mathbf{R}$, $i, k \in \{1, 2, \ldots, m\}$ with $i \neq k$.

This kind of constraints are redundant and one of the constraints can be eliminated from the LP. The Gaussian elimination method is needed for spotting of all linearly dependent constraints. Furthermore, linearly dependent constraints can be also spotted during the process of calculating the rank of matrix A.

The first step is to construct the augmented matrix $[A|b]$, since operations on equations also affect their right hand side. The main goal of this process is to produce an equivalent matrix that will have an upper triangular form. Consequently, constraints of the following form will be produced

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = 0, i = 1, 2, \ldots, m \tag{4.13}$$

where $a_{ij} = 0, j = 1, 2, \ldots, n$.

In case that during operations a constraint is produced of the form

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i, i = 1, 2, \ldots, m \tag{4.14}$$

where $a_{ij} = 0$ and $b \neq 0, j = 1, 2, \ldots, n$, then the LP is infeasible.

4.11 A New Presolve Technique - Eliminate Redundant Columns

The proposed new presolve technique is derived from the analysis of the primal feasibility conditions. It performs well in the process of reducing the dimensions of LPs.

Case 1. A linear constraint of the form:

$$a_1x_1 + a_2x_2 + \ldots + a_kx_k = 0 \tag{4.15}$$

with $a_i > 0$, $i = 1, 2, \ldots, k$ implies that $x_j = 0 \; \forall \; j = 1, 2, \ldots, k$.

Proof. It is very easy to check the correctness of the above statement. Assume that there is a feasible solution to the LP. As a consequence $x_j = 0$, $j = 1, 2, \ldots, n$. Then, according to the constraint (1) and with all $a_i > 0$, $i = 1, 2, \ldots, k$, we can conclude that either there is $x_j < 0$ with $j \neq i$ or $x_j = 0 \; \forall \; j = 1, 2, \ldots, k$. The first case $x_j < 0$ with $j \neq i$ is not possible because it is against the constraint $x_j \geq 0$. Consequently, $x_j = 0 \; \forall \; j = 1, 2, \ldots, k$.

Case 2. Likewise, a constraint of the form:

$$a_1x_1 + a_2x_2 + \ldots + a_kx_k = 0 \tag{4.16}$$

with $a_i < 0$, $i = 1, 2, \ldots, k$ implies that $x_j = 0 \; \forall \; j = 1, 2, \ldots, k$.

Proof. Similarly, there are two possible cases; the first claims that all $x_j = 0 \; \forall \; j = 1, 2, \ldots, k$ and the second that there is $x_i < 0$ with $j \neq i$, which is not possible because this case is excluded by the constraint $x_j \geq 0$ that stands for all variables in a LP.

Hence, all variables $x_j = 0$, $j = 1, 2, \ldots, k$ are redundant in both cases and they can be deleted from the LP. Consequently, the constraints are linearly dependent and they can be deleted from the LP.

In the next section, we present some small examples of our new presolve technique in order to demonstrate it in practice.

Illustrative Examples

We illustrate our presolve technique by applying it to the following LP, which is optimal with an optimal value of the objective function $z* = 2.6667$.

$$
\begin{array}{lrcrcrcrcr}
\min & -2\ x_1 & + & 3x_2 & - & x_3 & + & 5x_4 & & \\
\text{s.t.} & 3\ x_1 & + & 5x_2 & + & 7\ x_3 & + & 2x_4 & = & 4 \\
& -\ x_1 & + & x_2 & - & 2\ x_3 & - & x_4 & \geq & -\ 14 \\
& 9\ x_1 & + & 15x_2 & + & 21x_3 & + & 11x_4 & \leq & 12 \\
& 2\ x_1 & - & 3x_2 & - & 5\ x_3 & + & 3x_4 & \leq & 16
\end{array}
$$

$$x_j \geqslant 0,\ j = 1, 2, 3, 4$$

First of all, the slack variables x_5, x_6 and x_7 are introduced. In matrix notation the above problem is written as follows:

$$
\begin{array}{ll}
min & c^T x \\
s.t. & Ax = b \\
 & x \geq 0
\end{array}
\qquad (LP.1)
$$

where

$$C = \begin{bmatrix} -2 & 3 & -1 & 5 & 0 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 3 & 5 & 7 & 2 & 0 & 0 & 0 \\ -1 & 1 & -2 & -1 & -1 & 0 & 0 \\ 9 & 15 & 21 & 11 & 0 & 1 & 0 \\ 2 & -3 & -5 & 3 & 0 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 4 \\ -14 \\ 12 \\ 16 \end{bmatrix}$$

After the application of appropriate elementary row operations, like the multiplication of the first row of the constraint matrix A by -3 and its addition to the third row of

the constraint matrix A, the updated constraint matrix A and vector b are:

$$A = \begin{bmatrix} 3 & 5 & 7 & 2 & 0 & 0 & 0 \\ -1 & 1 & -2 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 1 & 0 \\ 2 & -3 & -5 & 3 & 0 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 4 \\ -14 \\ 0 \\ 16 \end{bmatrix}$$

Now, the third row of the above constraint matrix A is the following:

$$5x_4 + x_6 = 0 \tag{4.17}$$

In this case variables x_4, x_6 and the third constraint are redundant and they can be eliminated. Consequently, the new matrices of the specific LP are:

$$C = \begin{bmatrix} -2 & 3 & -1 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 3 & 5 & 7 & 0 & 0 \\ -1 & 1 & -2 & -1 & 0 \\ 2 & -3 & -5 & 0 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} 4 \\ -14 \\ 16 \end{bmatrix}$$

As we can observe, the number of variables has been reduced after the application of the new presolve technique. Moreover, the reduced problem is still an optimal problem with optimal objective value $z* = 2.6667$.

Apart from that, it is possible to detect a constraint of the form:

$$a_1x_1 + a_2x_2 + \ldots + a_kx_k = 0 \tag{4.18}$$

with $a_i < 0$, $i = 1, 2, \ldots, k$, which means that $x_j = 0 \ \forall \ j = 1, 2, \ldots, k$. The next example shows clearly this case. Consider the following LP after the addition of slack variables x_5, x_6, x_7 and x_8.

$$
\begin{array}{lrcrcrcrcrcrcrcrcr}
\min & -4\,x_1 & + & x_2 & - & 2\,x_3 & + & x_4 & & & & & & & & \\
\text{s.t.} & x_1 & + & 2x_2 & + & 3\,x_3 & + & 4x_4 & + & x_5 & & & & & = & 4 \\
& -\,x_1 & - & x_2 & - & 2\,x_3 & - & x_4 & & & + & x_6 & & & \leq & -\,14 \\
& 2\,x_1 & + & x_2 & + & 4\,x_3 & + & 5x_4 & & & & & + & x_7 & \leq & 12 \\
& 2\,x_1 & + & x_2 & + & 4\,x_3 & + & 2x_4 & & & & & & & + \; x_8 \; \leq & 16
\end{array}
$$

$$x_j \geqslant 0,\ j = 1, 2, 3, 4$$

The above LP is an optimal one with optimal value of the objective function $z^* = 12$. In matrix notation, the specific LP is written as follows:

$$C = \begin{bmatrix} -4 & 1 & -2 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 & 0 & 0 & 0 \\ -1 & -1 & -2 & -1 & 0 & -1 & 0 & 0 \\ 2 & 1 & 4 & 5 & 0 & 0 & 1 & 0 \\ 2 & 1 & 4 & 2 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$b = \begin{bmatrix} 8 \\ -3 \\ 11 \\ 6 \end{bmatrix}$$

If we multiply the second row of the above constraint matrix A by 2 and add it to the last row of the constraint matrix A and vector b, the new constraint matrix A and vector b are:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 & 0 & 0 & 0 \\ -1 & -1 & -2 & -1 & 0 & -1 & 0 & 0 \\ 2 & 1 & 4 & 5 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & -2 & 0 & -1 \end{bmatrix}$$

$$b = \begin{bmatrix} 8 \\ -3 \\ 11 \\ 0 \end{bmatrix}$$

Note that the last row of the constraint matrix A has the following form:

$$-x_2 - 2x_6 - x_8 = 0 \tag{4.19}$$

In this case, variables x_2, x_6, and x_8, and the last constraint are redundant and therefore can be deleted. The new matrices and vectors of the LP are:

$$C = \begin{bmatrix} -4 & -2 & 1 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 3 & 4 & 1 & 0 \\ -1 & -2 & -1 & 0 & 0 \\ 2 & 4 & 5 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 8 \\ -3 \\ 11 \end{bmatrix}$$

The reduced LP is still optimal with an optimal value of the objective function $z* = 12$.

Another significant issue of the proposed new presolve technique is that it can be combined with other presolve techniques in order to achieve greater dimensions reductions in LPs. For example, it is very possible to appear linearly dependent constraints after the deletion of one variable. Two constraints a_i and a_k with $i \neq k$ are called linearly dependent if $a_i = \lambda a_k$ with $\lambda \in \mathbf{R}$. The Gaussian elimination method can be used for spotting of all the linearly dependent constraints. In the next example, it is clear how these two methods can be combined and how significant results can be produced. Next, there is a LP after the addition of the slack variable x_6:

$$C = \begin{bmatrix} 1 & 1 & -2 & 1 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 4 & 1 & 5 & 1 & 0 \\ 7 & 8 & 4 & 6 & 0 & 0 \\ 4 & 4 & 2 & 3 & 0 & 0 \\ 5 & 12 & 4 & 15 & 3 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 12 \\ 8 \\ 4 \\ 36 \end{bmatrix}$$

The above LP is optimal and the optimal value of the objective function is $z^* = 1$. If we multiply the third row of constraint matrix A by -2 and we add it to the second row, the updated constraint matrix A and vector b are:

$$A = \begin{bmatrix} 1 & 4 & 1 & 5 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 2 & 3 & 0 & 0 \\ 5 & 12 & 4 & 15 & 3 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 12 \\ 0 \\ 4 \\ 36 \end{bmatrix}$$

Now, the second row of the constraint matrix A is the following:

$$-x_1 = 0 \tag{4.20}$$

In this case, variable x_1 is redundant and it can be deleted. The new matrices and vectors of the LP are:

$$C = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 4 & 1 & 5 & 1 & 0 \\ 8 & 4 & 6 & 0 & 0 \\ 4 & 2 & 3 & 0 & 0 \\ 12 & 4 & 15 & 3 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 12 \\ 8 \\ 4 \\ 36 \end{bmatrix}$$

It is clear that constraints 2 and 3 are linearly depended and as a result one of them can be eliminated from the problem. We delete the second constraint and the new problem is:

$$C = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 4 & 1 & 5 & 1 & 0 \\ 4 & 2 & 3 & 0 & 0 \\ 12 & 4 & 15 & 3 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 12 \\ 4 \\ 36 \end{bmatrix}$$

We can also continue with the application of our new presolve technique and multiply the first constraint by -3 and then add it to the last constraint. The new constraint matrix A and vector b are:

$$A = \begin{bmatrix} 4 & 1 & 5 & 1 & 0 \\ 4 & 2 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 12 \\ 4 \\ 0 \end{bmatrix}$$

The last constraint of the constraint matrix A is:

$$x_3 + x_6 = 0 \tag{4.21}$$

Hence, variables x_3, x_6 and the last constraint are redundant and they can be deleted. The new matrices of the linear problem are:

$$C = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 4 & 5 & 1 \\ 4 & 3 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} 12 \\ 4 \end{bmatrix}$$

The reduced problem is still optimal with optimal value of the objective function $z^* = 1$. It is obvious, that the advantages of the new presolve technique in combination with other presolve techniques are of great value and they can lead to significant dimensions reductions of LPs. In the latter example, the initial problem had 4 constraints and 6

TABLE 4.1: Description of the Computing Environment

CPU	Intel(R) Core™, i7 3.00 GHz (2 processors)
RAM size	16384 MB
L3 Cache size	8 MB
L2 Cache size	4x256 KB
L1 Cache size	4x32 KB
Operating System	Microsoft Windows 7 Professional SP1
MATLAB version	7.0.1.24704 R14 SP1

variables and the final problem has 2 constraints and 3 variables. We achieve a 50% reduction in constraints and 50% reduction in variables.

4.12 Computational Study

The computational comparison was implemented in the MATLAB version R14. The main reasons for this choice were the inherent capability of MATLAB for matrix operations and the support for sparse matrices. Our tests ran in the environment described in Table 4.1.

We implemented the following presolve techniques:

- Eliminate empty rows
- Eliminate empty columns
- Eliminate singleton equality rows
- Eliminate singleton inequality rows
- Eliminate dual singleton inequality rows
- Eliminate implied free singleton columns
- Eliminate implied bounds from constraints
- Make coefficient matrix A structurally full rank
- Eliminate linearly dependent rows

The presolve techniques were implemented as a function. A LP is passed as input to this function along with some others parameters. In this computational study, 53 optimal benchmarks are used, from the NETLIB collection, that do not have bounds and ranges in their MPS files.

4.12.1 Statistics Before the Presolve Analysis

Below there is some useful information about the optimal benchmarks, which are going to be used in the computational study. The first column of the table includes the name of the benchmark, the second the number of constraints, the third the number of variables, the fourth the number of non-zeros elements of matrix A, the fifth column the sparsity of the constraint matrix A, the sixth the maximum value of the constraint matrix A, the seventh the minimum value of the constraint matrix A, the eighth the condition number of the constraint matrix A and the ninth the condition number of matrix M. Using mathematical notation, matrix M is:

$$M = \begin{bmatrix} A & b \\ c & 0 \end{bmatrix}$$

TABLE 4.2: Optimal Benchmarks: Statistics Before the Presolve Analysis

Problem	Constraints	Variables	Non-Zeros A	Sparsity A	Max(A)	Min(A)	C.N. A	C.N. M
25FV47	820	1,571	10,400	0.81%	238.95	-207	3,89E+19	5,62E+18
ADLITTLE	56	97	383	7.05%	55	-64.3	9,37E+02	9,71E+04
AFIRO	27	32	83	9.61%	2,429	-1.06	1.61E+16	1.99E+05
AGG	488	163	2,410	3.03%	424	-24.391	1.09E+19	4.53E+23
AGG2	516	302	4,284	2.75%	424	-24.391	8.98E+33	1.28E+27
AGG3	516	302	4,300	2.76%	424	-24.391	1.51E+34	3.03E+28
BANDM	305	472	2,494	1.73%	200	-104	3.79E+03	3.78E+03
BEACONFD	173	262	3,375	7.45%	500	-100	1.46E+04	1.03E+06
BLEND	74	83	491	7.99%	66	-14	1.69E+17	1.59E+17
BNL1	643	1,175	5,121	0.68%	5.4	-78	7.47E+34	3.43E+36
BNL2	2,324	3,489	13,999	0.17%	30	-78	1.81E+33	3.47E+33
BRANDY	220	249	2,148	3.92%	203.7	-60	Inf	Inf
D2Q06C	2,171	5,167	32,417	0.29%	2.32E+03	-982	1.48E+20	2.05E+19
DEGEN2	444	534	3,978	1.68%	1	-1	4.21E+17	1.57E+18
DEGEN3	1,503	1,818	24,646	0.90%	1	-1	5.61E+16	1.09E+17
E226	223	282	2,578	4.10%	771	-1.49E+03	1.58E+34	1.21E+34
FFFFF800	524	854	6,227	1.39%	1.09E+05	-120	3.85E+22	7.51E+21
ISRAEL	174	142	2,269	9.18%	1,600	-1,600	3.21E+16	7.26E+18
LOTFI	153	308	1,078	2.29%	1,000	-475	4.15E+07	6.68E+08
MAROS-R7	3,136	9,408	144,848	0.49%	1	-1	2.32E+06	2.83E+05
QAP8	912	1,632	7,296	0.49%	1	-1	1.86E+17	2.40E+19
QAP12	3,192	8,856	38,304	0.14%	1	-1	2.31E+17	5.16E+19
QAP15	6,330	22,275	94,950	0.07%	1	-1	1.54E+17	1.42E+20
SC50A	50	8	130	5.42%	2	-1	4.72E+06	6.63E+03
SC50B	50	48	118	4.92%	3	-1	7.43E+06	1.21E+04
SC105	105	103	280	2.59%	2	-1	1.11E+02	2.86E+04

SC205	205	203	551	1.32%	2	-1	6.39E+02	2.06E+05
SCAGR7	129	140	420	2.33%	1.5	-9.32	1.03E+04	6.75E+05
SCAGR25	471	500	1,554	0.66%	1.5	-9.32	2.70E+09	5.04E+06
SCFXM1	330	457	2,589	1.72%	99	-130	6.38E+17	3.17E+18
SCFXM2	660	914	5,183	0.86%	99	-130	1.26E+18	5.36E+18
SCFXM3	990	1,371	7,777	0.57%	99	-130	1.02E+18	5.24E+18
SCORPION	388	358	1,426	1.03%	1	-1	2.75E+03	6.06E+05
SCRS8	490	1,169	3,182	0.56%	389	-2	5.66E+16	2.84E+17
SCSD1	77	760	2,388	4.08%	1	-1	2.12E+06	2.27E+02
SCSD6	147	1,350	4,316	2.17%	1	-1	8.85E+06	1.29E+03
SCSD8	397	2,750	8,584	0.79%	1	-1	9.93E+02	2.24E+04
SCTAP1	300	480	1,692	1.18%	80	-33	1.11E+17	1.07E+17
SCTAP2	1,090	1,880	6,714	0.33%	80	-33	6.67E+16	6.45E+17
SCTAP3	1,480	2,480	8,874	0.24%	80	-33	2.16E+17	2.05E+18
SHARE1B	117	225	1,151	4.37%	1.32E+03	-480	1.38E+05	3.54E+05
SHARE2B	96	79	694	9.15%	101	-103	1.84E+18	8.25E+16
SHIP04L	402	2,118	6,332	0.74%	4.706	-1	Inf	Inf
SHIP04S	402	1,458	4,352	0.74%	4.706	-1	Inf	Inf
SHIP08L	778	4,283	12,802	0.38%	5	-1	Inf	Inf
SHIP08S	778	2,387	7,114	0.38%	5	-1	Inf	Inf
SHIP12L	1,151	5,427	16,170	0.26%	1.6	-1	Inf	Inf
SHIP12S	1,151	2,763	8,178	0.26%	1.6	-1	Inf	Inf
STOCFOR1	117	111	447	3.44%	336.6	-1	6.36E+05	4.69E+04
STOCFOR2	2,157	2,031	8,343	0.19%	336.6	-1	1.07E+06	1.29E+05
TRUSS	1,000	8,806	27,836	0.32%	1	-1	4.77E+02	2.53E+04
WOOD1P	244	2,594	70,215	11.09%	244	-1,000	1.22E+14	1.37E+14
WOODW	1,098	8,405	37,474	0.41%	1,000	-1,000	4.76E+04	4.77E+04

4.12.2 Statistics After the Presolve Analysis

In Table 4.3, there are the same statistics as above, the number of constraints, variables, non-zeros elements of the constraint matrix A, sparsity of the constraint matrix A, condition number of the constraint matrix A and condition number of matrix M for the same optimal benchmarks. In contrast, all these values refer to the benchmarks after the presolve techniques have been performed (only the proposed new presolve technique was not used).

As it is obvious from Tables 4.2 and 4.3, all presolve techniques are important and they can lead to significant reductions. Apart from that, it is clear why presolve techniques became a fundamental component of all the commercial linear programming packages. According to these tables, the percentages of reduction for constraints, variables and the non-zeros of the constraitn matrix A are of great value. In many benchmarks, the range of reductions varies from 15% to 45% and in some benchmarks, the percentage of reduction is more than 50%. Furthermore, it is significant to comment the values of the condition number of matrices A and M after the presolve techniques are applied to the benchmarks. It is obvious that these values have also been significantly reduced. Some illustrative examples are the reductions in benchmarks like AGG2, BEACONFD, BNL2, BRANDY, MAROS-R7, SHIP04L, SHIP04S, SHIP08L, SHIP08S, SHIP012L and SHIP012S. On the other hand, there is no reduction of the data size in some benchmarks (only 10 from 53).

Table 4.3: Optimal Benchmarks: Statistics After the Presolve Analysis

Problem	Constraints	Variables	Non-Zeros A	Sparsity A	Max(A)	Min(A)	C.N. A	C.N. M
25FV47	788	1,541	10,236	0.84%	238.95	-206.9	3.49E+18	3.68E+18
ADLITTLE	55	95	375	7.18%	55	-64.3	9.38E+06	98,875.21
AFIRO	27	32	83	9.61%	2.43	-1.06	1.61E+16	1.99E+05
AGG	468	163	2,348	3.08%	424	-2.44	7.42E+33	1.42E+38
AGG2	390	112	1,723	3.94%	424	-2.44	1.05E+19	8.07E+23
AGG3	514	301	4,274	2.76%	424	-2.44	9.27E+32	4.11E+27
BANDM	255	422	2,032	1.89%	200	-104	2.93E+18	3.89E+18
BEACONFD	104	193	1,826	9.10%	1	-1	8.41E+16	8.67E+20
BLEND	71	80	446	7.85%	66	-14	7.19E+17	3.44E+17
BNL1	618	1,169	5,095	0.71%	5.4	-78	4.04E+40	1.35E+35
BNL2	1,853	3,007	12,674	0.23%	30	-78	1.25E+65	2.61E+35
BRANDY	134	207	1,901	6.85%	203.7	-60	3.56E+16	1.18E+16
D2Q06C	2,131	5,124	31,472	0.29%	2,322.7	-981.7	7.75E+18	9.63E+18
DEGEN2	442	534	3,944	1.67%	1	-1	9.41E+16	1.37E+18
DEGEN3	1,501	1,818	24,639	0.90%	1	-1	5.55E+16	1.11E+17
E226	200	271	2,470	4.56%	771	-1.49	8.13E+48	5.27E+33
FFFFF800	322	663	5,001	2.34%	221.29	-120	1.84E+34	1.11E+37
ISRAEL	174	142	2,269	9.18%	1,600	-1,600	3.21E+16	7.26E+18
LOTFI	133	288	809	2.11%	1000	-475	14.31E+06	3.52E+08
MAROS-R7	2,152	7,440	100,486	0.63%	1	-1	2.27E+04	2.82E+09
QAP8	912	1,632	7,296	0.49%	1	-1	1.86E+17	2.40E+19
QAP12	3,192	8,856	38,304	0.14%	1	-1	2.31E+17	5.16E+19
QAP15	6,330	22,275	94,950	0.07%	1	-1	1.54E+17	1.42E+20
SC50A	49	48	130	5.53%	2	-1	4.72E+06	6.63E+06
SC50B	48	48	118	5.12%	3	-1	7.43E+06	12050.53
SC105	104	103	280	2.61%	2	-1	1.11E+06	1.11E+06

SC205	203	202	550	1.34%	2	-1	6.39E+06	20,5726.8
SCAGR7	127	138	410	2.34%	1.5	-9.32	10,230.31	57,9541.9
SCAGR25	469	498	1,544	0.66%	1.5	-9.32	26.9E+09	4.83E+06
SCFXM1	310	440	2,352	1.72%	99	-130	7.74E+32	2.02E+33
SCFXM2	620	880	4,709	0.86%	99	-130	1.52E+33	3.87E+33
SCFXM3	930	1,320	7,066	0.58%	99	-130	7.16E+32	6.47E+33
SCORPION	320	328	1,178	1.12%	1	-1	7.47E+17	1.78E+20
SCRS8	428	1,112	2,966	0.62%	388.77	-2	1.66E+21	4.36E+21
SCSD1	77	760	2,388	4.08%	1	-1	2.12E+06	2.27E+02
SCSD6	147	1,350	4,316	2.17%	1	-1	8.85E+06	1.29E+03
SCSD8	397	2,750	8,584	0.79%	1	-1	9.93E+02	2.24E+04
SCTAP1	284	480	1,638	1.20%	80	-33	4.60E+06	1.16E+06
SCTAP2	1,033	1,880	6,489	0.33%	80	-33	4.38E+06	2.10E+06
SCTAP3	1,408	2,480	8,595	0.25%	80	-33	7.21E+06	3.70E+06
SHARE1B	112	220	1,120	4.55%	1,322.2	-479.56	1.38E+05	4.55E+06
SHARE2B	96	79	694	9.15%	101	-103	1.84E+18	8.25E+16
SHIP04L	323	2,089	4,719	0.70%	4.71	-1	5.43E+30	4.54E+33
SHIP04S	235	1,341	3,029	0.96%	4.71	-1	2.48E+31	1.46E+34
SHIP08L	630	4,209	9,504	0.36%	5	-1	3.56E+30	3.38E+32
SHIP08S	358	2,041	4,612	0.63%	5	-1	8.12E+30	4.64E+32
SHIP12L	756	5,146	11,467	0.29%	1.6	-1	2.99E+31	6.20E+31
SHIP12S	384	2,110	4,694	0.58%	1.6	-1	3.57E+31	9.60E+32
STOCFOR1	102	96	367	3.75%	336.6	-1	53,8109.1	52,683.89
STOCFOR2	1,996	1,870	7,152	0.19%	336.6	-1	1,06E+06	12,9874.6
TRUSS	1,000	8,806	27,836	0.32%	1	-1	4.77E+02	2.53E+04
WOOD1P	243	2,593	67,710	10.75%	942.76	-1000	3.57E+31	9.60E+32
WOODW	1,094	8,401	31,566	0.34%	1000	-1000	3.57E+31	9.60E+32

4.12.3 Statistics For New Presolve Technique

In Table 4.4, there is some useful information that refers to the reduction in the size of LPs which has been made from the application of the new presolve technique. The new presolve technique has effect in 20 out of 53 benchmarks. The first column of the table includes the name of the benchmark, the second the number of constraints, the third the number of variables, the fourth the non-zeros elements of the constraint matrix A, the fifth the sparsity of the constraint matrix A, the sixth the condition number of the constraint matrix A and the seventh the condition number of matrix M.

TABLE 4.4: Optimal Benchmarks: Statistics After the New Presolve Technique

Problem	Constraints	Variables	Non-Zeros A	Sparsity A	C.N. A	C.N. M
AGG	390	112	1,723	3.94%	1.05E+19	8.07E+23
BANDM	243	398	1,925	1.99%	2.89E+06	3.82E+06
BEACONFD	82	143	1,255	10.70%	7.05E+06	1.01E+06
BNL1	618	1,169	5,095	0.71%	1.99E+35	1.87E+20
BNL2	1,853	3,007	12,674	0.23%	6.12E+36	2.79E+21
E226	199	266	2,388	4.51%	4.91E+34	1.31E+19
FFFFF800	322	663	5,001	2.34%	1.13E+19	1.19E+21
SCFXM1	305	431	2,320	1.76%	1.01E+18	3.03E+18
SCFXM2	610	862	4,645	0.88%	5.38E+05	3.00E+18
SCFXM3	915	1,293	6,970	0.59%	7.53E+17	4.67E+18
SCORPION	317	324	1,159	1.13%	7.47E+06	177,533.7
SCRS8	425	1,109	1,109	0.63%	1.66E+06	4,36E+06
SHIP04L	292	1,890	4,275	0.77%	2.46E+15	4.59E+18
SHIP04S	216	1,266	2,860	1.05%	1.45E+17	1.99E+19
SHIP08L	470	3,099	7,100	0.49%	3.99E+15	1.15E+18
SHIP08S	276	1,582	3,622	0.83%	3.82E+17	2.11E+17
SHIP12L	610	4,147	9,230	0.36%	4.02E+16	5.28E+16
SHIP12S	340	1,919	4,273	0.65%	3.29E+16	2.69E+18
WOOD1P	171	1,801	46,830	15.21%	4.65E+14	6.61E+14
WOODW	708	5,351	19,796	0.52%	28,769.9	29,540.73

Table 4.5 includes the reduction of the constraints, the variables and the non-zeros of the constraint matrix A. The first column refers to the constraints reduction that the new presolve technique is responsible for and the second has the total variables reduction which stems from all the presolve techniques including the proposed new technique. The next four columns refer to the variables reduction and the non-zeros of the constraint matrix A reduction. Moreover, the percentages for these reductions appear in the last three columns. On average, the new presolve technique has achieved 26.64% reduction in the number of constraints, 48.03% reduction in the number of variables and 30.73% reduction in the number of non-zeros.

TABLE 4.5: Optimal Benchmarks: New Presolve Technique Reduction and Rest Presolve Technique Reduction

Problem	New Technique Constraints Reduction	Total Techniques Constraints Reduction	New Technique Variables Reduction	Total Techniques Variables Reduction	New Technique Non-Zeros A Reduction	Total Techniques Non-Zeros A Reduction	%New Technique Constraints Reduction	%New Technique Variables Reduction	%New Technique Non-Zeros A Reduction
AGG	78	98	51	51	625	687	79.59%	100.00%	90.98%
BANDM	12	62	20	74	107	569	19.35%	27.03%	18.80%
BEACONFD	22	91	52	119	571	2,120	24.18%	43.70%	26.93%
BNL1	0	25	2	6	0	26	0.00%	33.33%	0.00%
BNL2	0	471	5	482	0	1,325	0.00%	1.04%	0.00%
E226	1	24	5	16	82	190	4.17%	31.25%	43.16%
FFFFF800	0	202	2	191	0	1,226	0.00%	1.05%	0.00%
SCFXM1	5	25	9	26	32	269	20.00%	34.62%	11.90%
SCFXM2	10	50	18	52	64	538	20.00%	34.62%	11.90%
SCFXM3	15	75	27	78	96	807	20.00%	34.62%	11.90%
SCORPION	3	71	4	34	19	267	4.23%	11.76%	7.12%
SCRS8	3	65	18	60	1,857	2,073	4.62%	30.00%	89.58%
SHIP04L	31	110	199	228	444	2,057	28.18%	87.28%	21.58%
SHIP04S	19	186	75	192	169	1,492	10.22%	39.06%	11.33%
SHIP08L	160	308	1,110	1,184	2,404	5,702	51.95%	93.75%	42.16%
SHIP08S	82	502	459	805	990	3,492	16.33%	57.02%	28.35%
SHIP12L	146	541	999	1,280	2,237	6,940	26.99%	78.05%	32.23%
SHIP12S	44	811	191	844	421	3,905	5.43%	22.63%	10.78%
WOOD1P	72	73	792	793	20,880	23,385	98.63%	99.87%	89.29%
WOODW	386	390	3,050	3,054	11,770	17,678	98.97%	99.87%	66.58%
AVERAGE							26.64%	48.03%	30.73%

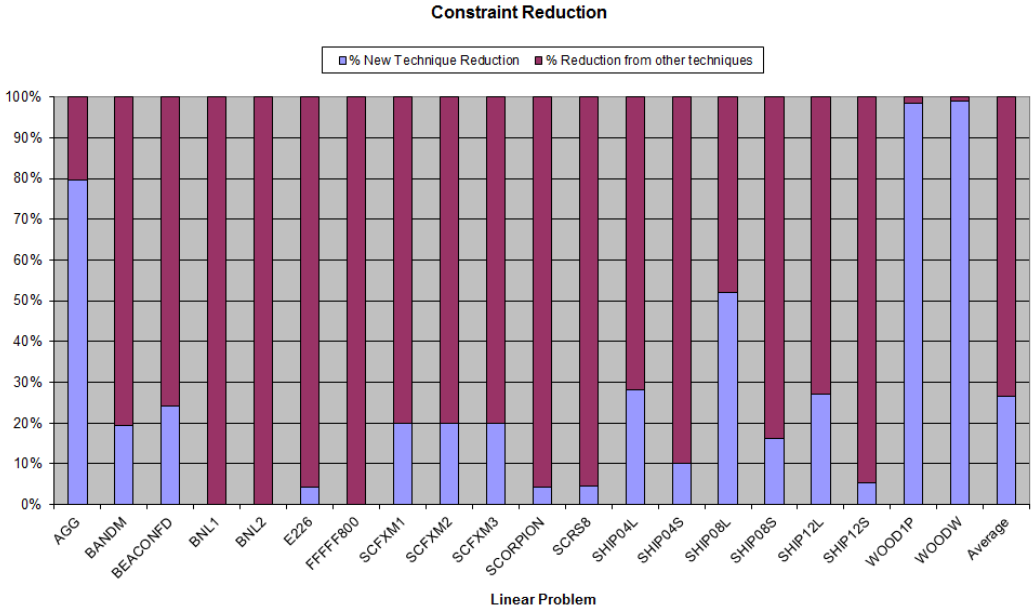

FIGURE 4.1: Constraint Reduction: % Reduction of the New Presolve Technique

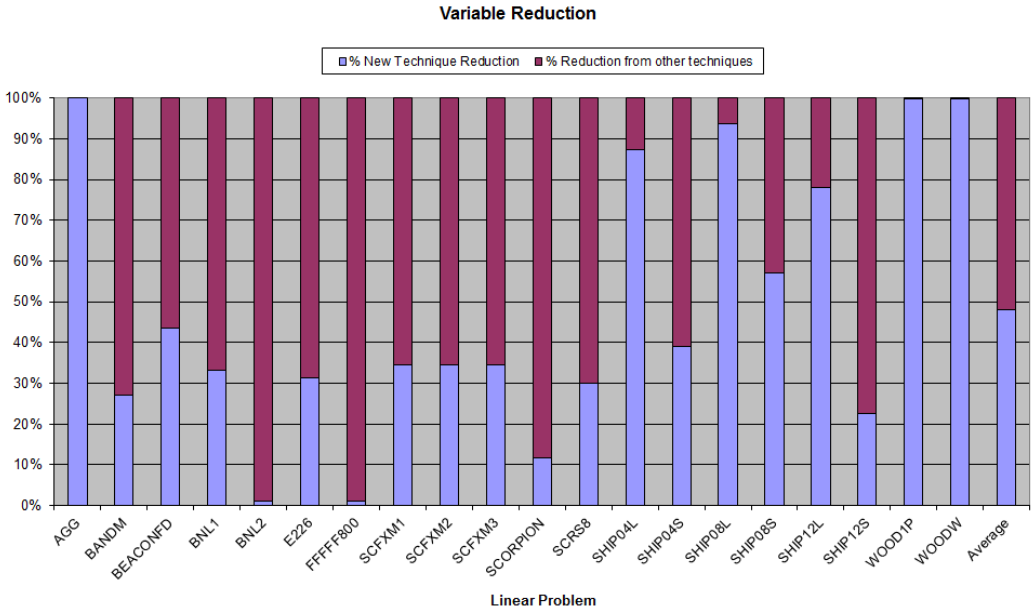

FIGURE 4.2: Variable Reduction: % Reduction of New the Presolve Technique

It is clear from the computational results that the new presolve technique can lead to significant reductions in problem's dimensions. The reduction of problem's dimension can be crucial for the execution time of a linear programming solver. According to the results, the new presolve technique has remarkable results in reduction of constraints, variables and non-zeros elements of a LP.

Figure 4.1, which refers to the constraints reduction, shows that the new presolve technique is almost totally responsible for the whole reduction in some benchmarks. These

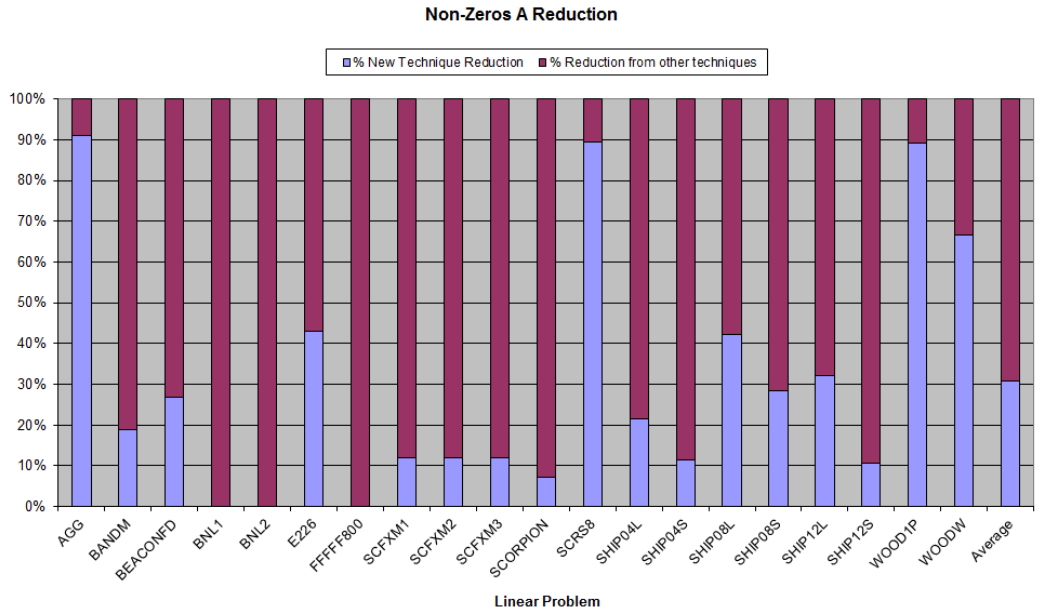

FIGURE 4.3: Non-Zeros A Reduction: % Reduction of New presolve Technique

benchmarks are the WOOD1P and WOODW and the reduction percentage that stems from the new presolve technique is above 95%. Moreover, AGG and SHIP08L present significant reductions that came from the new presolve technique. Finally, the new presolve technique is responsible for almost 20% reduction of the total reduction in the majority of benchmarks. In Figure 4.2, the variables reduction is presented and it is very interesting since the new presolve technique eliminates directly variables from a LP in contrast to previous results which show the constraints reduction after the combination of the new presolve with other presolve techniques, like the elimination of linearly dependent rows. Likewise, benchmarks AGG, WOOD1P and WOODW present the most promising results for the new presolve technique. Furthermore, the average percentage of variables reduction derived from the new presolve technique is 48.03%. In Figure 4.3, there is the percentage of reduction in the number of non-zeros. Benchmarks like AGG, SCRS8, WOOD1P and WOODW have significant reduction of non-zeros due to the new presolve technique, something which can lead to significant reduction of execution time in the process of solving these LPs. On average, the new presolve technique has achieved 26.64% reduction in the number of constraints, 48.03% reduction in the number of variables and 30.73% reduction in the number of non-zeros.

In order to gain a clearer and deeper insight about the practical performance of the proposed presolve technique, we carry out an experimental study. We solved the benchmarks of Table 4.4 before and after the new presolve technique is performed. For this purpose, we used the Optimization Toolbox of Matlab (linprog function). The Optimization Toolbox version 4.3 implements three different algorithms; a small-scale simplex algorithm,

a medium-scale active set projection method and a large-scale primal-dual interior point method. In our experiment, we used the simplex algorithm. All the reported CPU times were measured in seconds with the Matlab built-in function cputime. Table 4.6 reports results on the performance of the new presolve technique, before and after it is performed. $A*$ denotes that this problem is scaled using the equilibration scaling technique. The columns entitled *Niter* stand for the number of iterations needed by the simplex algorithm to solve a problem. The columns entitled *cpu* refer to the CPU time needed by the simplex algorithm to solve a problem. The last two columns show the percentages for the *Niter* and *cpu* reductions, respectively. On average, the new presolve technique has achieved 5.47% reduction in the number of iterations and 5.99% reduction in the execution time.

TABLE 4.6: Efficiency of the New Presolve Technique

	Before		**After**		**Δ %Niter**	**Δ %Cpu**
Problem	**Niter**	**Cpu**	**Niter**	**Cpu**	**Reduction**	**Reduction**
AGG	30	0.5460	22	0.4212	33.33%	22.86%
BANDM *	347	0.8892	217	0.6396	37.46%	28.07%
BEACONFD	13	0.0936	13	0.0936	0.00%	0.00%
BNL1	354	8.7205	354	8.7205	0.00%	0.00%
BNL2 *	2,612	111.2777	2,612	111.2777	0.00%	0.00%
E226	333	0.4992	324	0.4712	2.70%	5.61%
FFFFF800	173	0.7488	173	0.7488	0.00%	0.00%
SCFXM1	171	0.6240	158	0.5943	7.60%	4.76%
SCFXM2	318	1.6848	301	1.6611	5.35%	1.41%
SCFXM3	472	3.7596	447	3.5924	5.30%	4.45%
SCORPION	36	0.4368	36	0.4368	0.00%	0.00%
SCRS8	368	0.7176	336	0.6396	8.70%	10.87%
SHIP04L	246	0.4836	246	0.4836	0.00%	0.00%
SHIP04S	183	0.3276	183	0.3276	0.00%	0.00%
SHIP08L	379	4.3992	374	3.5880	1.32%	18.44%
SHIP08S	236	0.4524	213	0.3981	9.75%	12.00%
SHIP12L	634	6.4116	634	6.4116	0.00%	0.00%
SHIP12S	313	0.6240	330	0.6869	-5.43%	-10.08%
WOOD1P	217	1.7316	217	1.7316	0.00%	0.00%
WOODW	824	12.0815	796	9.5005	3.40%	21.36%
AVERAGE					**5.47%**	**5.99%**

4.13 Conclusions

From the computational results presented in previous section, it can be concluded that the presolve techniques are highly successful in reducing the size of a LP. Furthermore, the time taken by the presolve techniques is negligible compared to the time taken by the linear programming solver. In addition, the time taken by the linear programming

solver in order to solve the presolved problem is significantly less than the time taken by it to solve the same unpresolved LP. It should be noted that presolve techniques can be very useful if they are combined either with interior-point algorithms or simplex-type methods. Another possible use of the presolve procedure is as a model analyzing tool, because this procedure might reveal bad formulations such as primal and dual infeasibility. The possible detection of infeasible problems is especially advantageous for interior-point algorithms or simplex-type methods because this kind of problems is extremely time-consuming.

The computational results for the new presolve technique presented in this paper are quite satisfactory. This technique is a new approach for detecting and eliminating redundancies from a LP. The application of the proposed presolve technique reduces significantly the problem's dimensions, at the same time of decreasing the total execution time. According to computational results, the new presolve technique has yielded significant reductions in many optimal benchmarks.

Chapter 5

Primal-Dual Interior Point Simplex Algorithm

5.1 Introduction

As it was already mentioned in previous Chapters, there was a huge gap between the theoretical worst-case complexity and the practical performance of simplex-type algorithms. Apart from the simplex algorithm, another approach to solve LPs is not just to move from one basic vertex to another but to visit points in the exterior of the feasible region. The algorithm that includes these kinds of features is called Exterior Point Simplex Algorithm (EPSA). Furthermore, a significant improvement of EPSA is the primal-dual versions of the algorithm which also revealed its superiority contrary to the simplex algorithm and they are known as Primal-Dual Exterior Point Simplex Algorithms (PDEPSAs).

The aim of this chapter is to present an experimental investigation of a Primal-Dual Interior Point Simplex Algorithm (PDIPSA) for solving LPs [42]. As its name reveals PDIPSA is a variation of PDEPSAs and its main difference stems from the fact that it traverses across the interior of the feasible region in an attempt to avoid combinatorial complexities of vertex following algorithms. In order to gain an insight into the practical behavior of the proposed algorithm, we have performed some computational experiments. Our computational results demonstrate that PDIPSA is faster than the simplex algorithm and the primal exterior point simplex algorithm. A clear advantage is obtained by PDIPSA both in number of iterations and CPU time.

The structure of this Chapter is as follows. In Section 5.2, the basic concepts of primal-dual exterior point simplex algorithms are presented. In Section 5.3, we show geometrically the differences between PDIPSA and PDEPSA, while Section 5.4 presents PDIPSA. Section 5.5 includes the proof of correctness of the algorithm and Section 5.6 discusses the revised form of this algorithm and how it is able to solve general LPs. A computational study is presented in Section 5.7 to highlight the efficiency of the algorithm. Finally, conclusions are presented in Section 5.8.

5.2 PDEPSAs

In this section, we are going to describe a PDEPSA. For the initialization of the algorithm a dual feasible basis B is needed. We denote by $x = (x_B, x_N)$ and $(w, s) = (s_B, s_N)$ the solutions of the primal LP and the the dual LP corresponding to the basic partition (B, N), respectively. In addition, an initial point y is necessary for the initialization of the algorithm. This point can be a basic, a boundary or an interior point. As it is obvious, a sequence of dual feasible basis $B^n, n = 1, 2, ...$ is constructed by the algorithm and we denote by x^n the primal solution and (w^n, s^n) the dual solution. Moreover, x^n is not necessary feasible to the primal LP.

A direction $d' = x' - y'$ is calculated, where y' is the initial primal feasible solution of the primal LP; the y' can be a non-basic solution. Direction d' is an ascent direction for the objective function c^T because y' is feasible to the primal LP and x' refers to a feasible basis of the dual LP. The direction d' enters the feasible region of the primal LP through the boundary point y'' and this point lies on a hyperplane. This hyperplane corresponds to a basic variable, which we denote by $x_k < 0$. Consequently, a dual pivot operation takes place in which the basic variable x_k exits the basis. In addition, a new basis B'' is calculated and the corresponding solutions x'', w'', s''. After n iterations, the algorithm concludes on the dual feasible basis B^n and the points x^n, w^n, s^n, y^n, and the algorithms finishes when x^n becomes feasible to the primal LP.

In Figure 5.1, an illustrative example is presented. The algorithm moves from point x' to x'' and x''' and it also computes the points y' to y'' and y'''. The initial point y' is an interior point and the next points y'' and y''' are boundary points. Moreover, at each iteration the direction $d^i = y^i - x^i$ is computed and the points x^i and y^i, $i > 1$ lie in the same hyperplane $x_k = 0$. The algorithm stops at point $y''' = 0$, which is the optimal solution.

The algorithm is formally described in Table 5.1. From the geometrical representation, we observe that point $y = x + \lambda d$ lies on hyperplane $x_k = 0$ and y is feasible to the primal

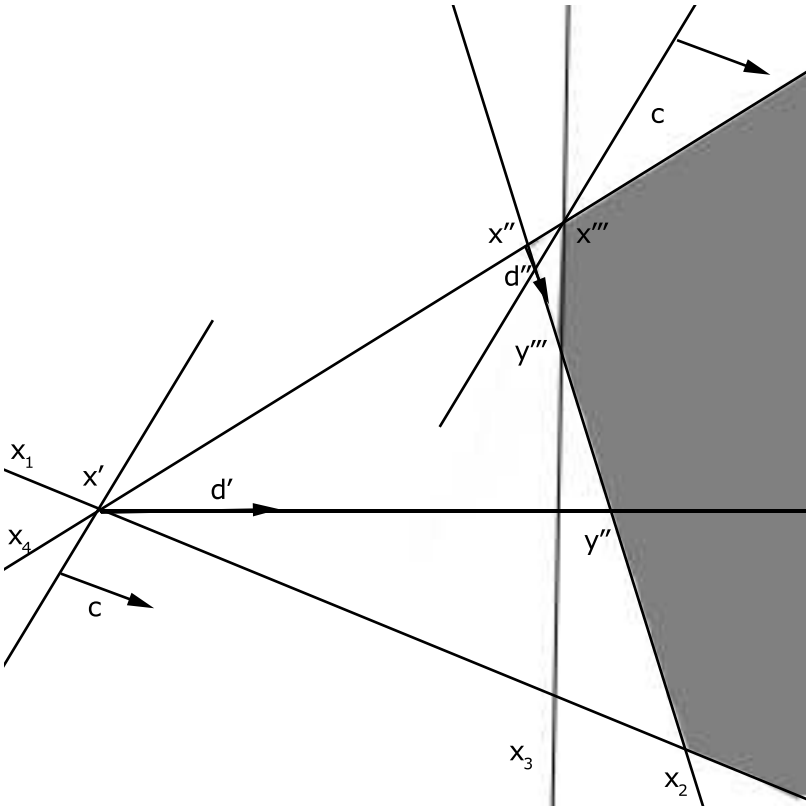

FIGURE 5.1: The geometrical representation.

LP. Furthermore, the correctness of the algorithm is easily proved due to the similarity to the dual simplex method. It is obvious that a dual feasible solution is computed in each iteration and the algorithm stops when the basic solution is also primal feasible. Consequently, the algorithm solves correctly any LP.

A good implementation of this type of PDEPSA is described by Paparrizos et al. [93]. The main idea of the Revised Primal Dual Simplex Algorithm (RPDSA) is based on the process of moving from any interior point to an optimal basic solution. Although this algorithm is better from the exterior point simplex algorithm, it also faces some weaknesses. RPDSA begins with an interior point and at each iteration a boundary point is used in order to compute the leaving variable. It has been observed that the problem of stalling and cycling can arise very often at this stage. This weakness can be overcome if the boundary point is replaced by an interior point. The transfer to the interior of the feasible region disappears the problem of stalling and cycling.

TABLE 5.1: Primal–Dual Exterior Point Simplex Algorithm

Step 0. *(Initialization).*
Start with a dual feasible basic partition (B, N) and a primal feasible solution y of the primal LP.
Set:

$$x_B = (A_B)^{-1} b,\ w^T = (c_B)^T (A_B)^{-1},\ s_N = (c_N)^T - w^T A_N, d = y - x$$

Step 1. *(General Step).*
while $(\exists i \in B : x_i < 0)$

$\lambda = \frac{x_{B[r]}}{-d_{B[r]}} = \max\left\{\frac{x_i}{-d_i} : i \in B, x_i < 0\right\}$

$y = x + \lambda d$

$H_{rN} = (B)^{-1}_{r.} A_N$

$\mu = \frac{s_{N(t)}}{H_{rN(t)}} = \min\left\{\frac{s_{N(j)}}{-H_{rN(j)}} : H_{rN(j)} < 0\right\}$

$k = B(r), p = N(t), B(r) = p, N(t) = k$

$x_B = (A_B)^{-1} b, w^T = (c_B)^T (A_B)^{-1}, (s_N)^T = (c_N)^T - w^T A_N$

$d = y - x$

end

5.3 Geometrical Representation

In this Chapter, we propose an alternative efficient approach of PDEPSA that it clearly outperforms the simplex algorithm and EPSA. This approach is much more efficient because it can overcome two significant drawbacks of PDEPSA, stalling and cycling. Furthermore, this algorithm is primal–dual, meaning that it simultaneously solves both the primal and the dual LP. In contrast to PDEPSA, in our algorithm, we use an interior point at each iteration to compute the leaving variable and this is the key factor that leads the algorithm to avoid the problem of stalling and cycling.

A geometrical representation is necessary to clarify the reasons that our algorithm can deal quite satisfactory with these two drawbacks. In Figure 5.2, we present a LP where the problem of stalling arises. We assume that our algorithm is at vertex A at the current iteration. According to PDEPSA, the direction d' computes the boundary point y' from which it enters the feasible region. This point is used to choose the leaving variable (constraint); at this point, there are three possible options, constraints (1), (2) or (3). In other words, there are bonds in the specific LP. If the next leaving variable is constraint (1), then our algorithm will move to vertex B. In this case, the new direction enters again the feasible region from point y'; keeping the point y' boundary, the algorithm will move to vertices C and D consecutively until it reaches vertex E, which is the optimal solution. Consequently, it will compute the optimal value after four iterations; this phenomenon of pivoting between degenerated vertices of the feasible region is called stalling. The problem of stalling can be overcome if the boundary point y' is replaced from an interior point y''. Now, using this interior point the direction d''

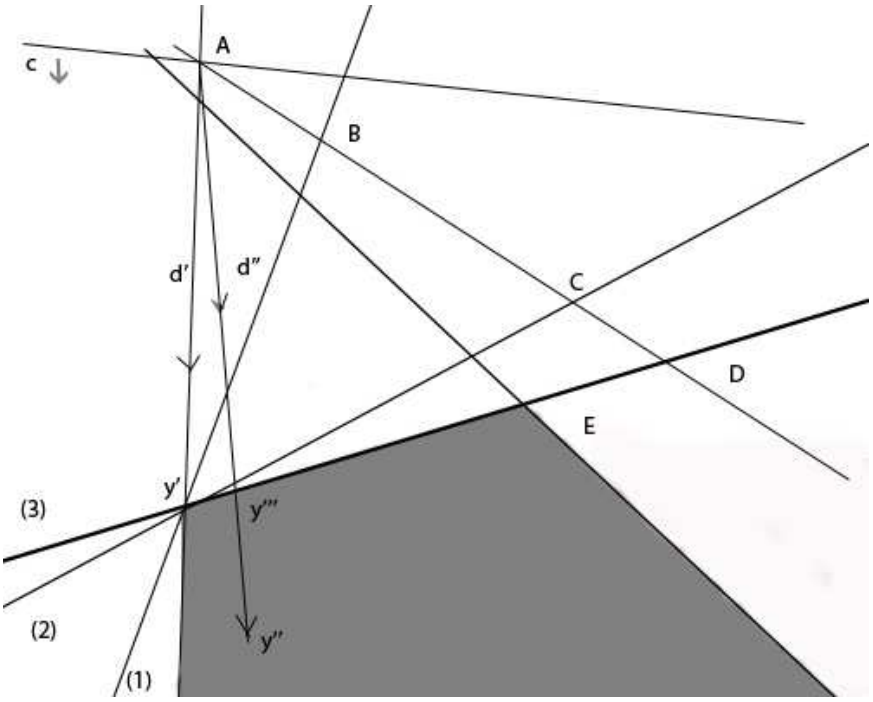

FIGURE 5.2: Stalling

enters the feasible region from the boundary point point y'''. Only constraint (3) comes through this point and PDIPSA will move to vertex D without visiting vertices B and C. Consequently, the optimal solution will be calculated in two iterations in contrast to PDEPSA that needs four iterations.

Apart from the problem of stalling, PDEPSA has another significant drawback, it is vulnerable to cycling. In order to clarify this specific situation, we assume in the above LP that the objective function is parallel to line $\varepsilon 1$ (see Figure 5.4). According to PDEPSA, the algorithm's pivot from vertex A to vertex D cannot lead to any change of the objective function's value. Furthermore, in such cases, the algorithm may continue cycling from one vertex to another. This weakness can be also overcome if we replace the boundary point with an interior point as it was described previously.

5.4 Description of PDIPSA

In this section, we briefly describe PDIPSA (for a full description of PDIPSA see [105] and for solving general LPs with PDIPSA see [93]).

PDIPSA is initialized with a dual feasible basis. The procedure of a dual feasible basis construction is based on a big–M problem [105]; so, it essentially attempts to execute phase I and phase II of a simplex-type algorithm in a single execution. The solutions of

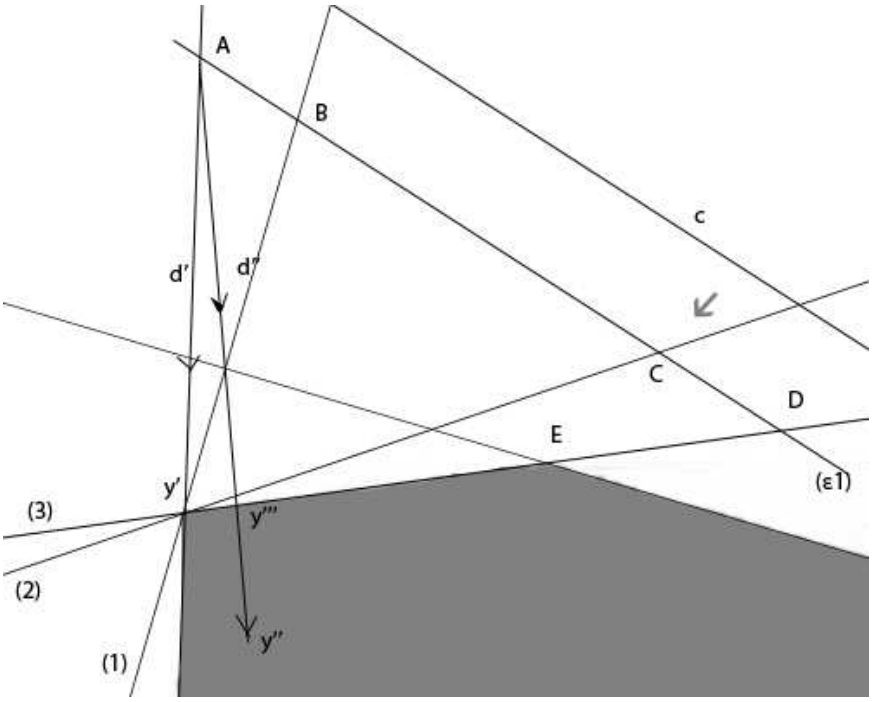

FIGURE 5.3: Cycling

the primal LP and the dual LP corresponding to the basic partition (B, N) are denoted by $x = (x_B, x_N)$ and $(w, s) = (w, s_N)$, respectively. We assume that an interior point y to problem (LP.1) is available. We explain how we choose the initial point y later in this Section. The main idea of our algorithm is that the point y must be an interior point. The algorithm generates a sequence of dual feasible bases $B^{(k)}$, $k = 1, 2, \cdots$. The primal solution $x^{(k)}$ and the dual solution $(w^{(k)}, s^{(k)})$ correspond to the basis $B^{(k)}$. Recall that $x^{(k)}$ is not (in general) feasible to the primal LP.

Initially a direction $d_B = y_B - x_B$, where y_B is an initial interior point of the primal LP, is computed. As x_B corresponds to a feasible basis of the dual LP and y_B is feasible to primal LP, the direction d_B is an ascent direction for the objective function $c^T x$.

At the next step, the leaving variable $x_k = x_{B[r]}$ is calculated from the maximum ratio test:

$$a = \frac{x_{B[r]}}{-d_{B[r]}} = max \left\{ \frac{x_i}{-d_i} : i \in B, d_i > 0 \wedge x_i < 0 \right\}$$

i.e., r is the index of the basic list B where the fraction $\frac{x_i}{-d_i}$, $i \in B$, $d_i > 0 \wedge x_i < 0$, is maximized. In case of ties, the rightmost index is selected.

The ray $R = \{x + a'd : a' \geq 0\}$ enters the feasible region of the primal LP through the boundary point $x + ad$. Then, the algorithm moves to the point y_B, which is inside the feasible region. This computation is achieved by the relation $y_B = x_B + a'd_B$, where $a' = \frac{a+1}{2}$.

Let H be a hyperplane on which the point y_B lies on. This hyperplane corresponds to a basic variable, which we denote by x_k. Obviously, $x_k \geq 0$. At this point, a dual pivot operation on which the basic variable x_k exits the basis is performed. This operation computes a new dual feasible basis A_B and the corresponding solutions x, w, and s. The algorithm terminates when x becomes feasible to (LP.1).

PDIPSA can be described formally as shown in Table 5.2.

TABLE 5.2: Primal-Dual Interior Point Simplex Algorithm

Step 0. *(Initialization).*
A) Start with a dual feasible basic partition (B, N) and an interior point $y > 0$ of (LP.1).
Set:

$$P = N, Q = \varnothing$$

and compute

$$x_B = (A_B)^{-1} b,\ w^T = (c_B)^T (A_B)^{-1},\ s_N = (c_N)^T - w^T A_N$$

B) Compute the direction d_B from the relation: $d_B = y_B - x_B$
Step 1. *(Test of optimality and choice of the leaving variable).*
if $x \geq 0$ then STOP. (LP.1) is optimal.
else
Choose the leaving variable $x_k = x_{B[r]}$ from the relation:

$$a = \frac{x_{B[r]}}{-d_{B[r]}} = max\left\{\frac{x_i}{-d_i} : i \in B, d_i > 0 \wedge x_i < 0\right\}$$

Step 2. *(Computation of the next interior point).*
Set:

$$a' = \frac{a+1}{2}$$

Compute the interior point from the relation: $y_B = x_B + a'd_B$
Step 3. *(Choice of the entering variable).*
Set: $H_{rN} = (A_B)_r^{-1} A_N$
Choose the entering variable x_l from the relation:

$$\frac{-s_l}{H_{rN}} = \min\left\{\frac{-s_l}{H_{rN}} : H_{rj} \wedge j \in N\right\}$$

Compute the pivoting column: $h_l = (A_B)^{-1} A_{.l}$
if $l \in P$ then
$P \leftarrow P \setminus \{l\}$
else
$Q \leftarrow Q \setminus \{l\}$
Step 4. *(Pivoting).*
Set:

$$B[r] = l \text{ and } Q \leftarrow Q \cup \{k\}$$

Using the new partition (B, N) where $N = (P, Q)$, compute/update the new basis inverse A_B^{-1} and the variables x_B, w, and s_N.
Go to step $0B$.

By the choice of a in the maximum ratio test, we conclude that the point $y = x + a'd$, where $a' = \frac{a+1}{2}$, lies on hyperplane $x_k = 0$. Also, y is feasible to (LP.1). Furthermore, the geometrical representation, which was presented previously, and the similarity of the algorithm to the dual simplex algorithm (in fact, it can be seen as a modification of the dual simplex algorithm) reveals immediately the correctness of the algorithm. The algorithm constructs dual feasible solutions. When it stops, the basic solution is also primal feasible. Hence, the algorithm correctly solves any LP. Furthermore, PDIPSA can deal very effectively with the problem of stalling and cycling.

A revised form of the algorithm is implemented in this dissertation. Any known technique for updating the basic inverse matrix $(A_B)^{-1}$ and the vectors x_B, w, and s_N can be combined with the algorithm. Here, our effort is focused on how the direction d can be computed more efficiently. Our results lead to a different calculation of d_B; the component d_B is updated in a way very similar to that of x_B. Moreover, our results lead to the construction of a big–M problem (for solving general LPs). The revised form of the algorithm and the method it uses to solve general LPs are discussed in Section 5.6.

5.5 Proof of Correctness

The geometrical representation that was presented in Section 5.3 and the similarity of the proposed algorithm to the dual simplex method reveal immediately the correctness of the algorithm. The algorithm finds a basic solution that is also primal feasible when it terminates. To complete the proof of correctness of the algorithm, it suffices to show that every basic partition, which is constructed by PDIPSA, is dual feasible and the computation of the maximum ratio test is well defined.

Theorem 1: If the initial basic partition of PDIPSA is dual feasible, then every consecutive partition is dual feasible.

Proof: The proof is by induction on the number of iterations. Denote by k the number of iterations. It is obvious from Step $0A$ of PDIPSA that for $k = 1$ the relations $S_j^{(1)} \geq 0\,, j \in N^{(1)}$, and $S_j^{(1)} = 0, j \in B^{(1)}$, hold. Suppose now that the relation $S_j^{(k)} \geq 0$, $j \in N^{(k)}$, holds. Let $(B^{(k+1)}, N^{(k+1)})$ be the new basic partition and $S_j^{(k+1)}$, $j \in N^{(k+1)}$, the corresponding dual slack variables. The dual slack variables can be computed by the relation

$$S_j^{(k+1)} = S_j^{(k)} - \frac{S_l^{(k)}}{H_{rl}} H_{rj}, j \in N^{(k+1)} \tag{5.5.1}$$

where H_{rj}, $j \in N^{(k+1)}$, is the pivot row. From the choice of the entering variable x_l

$$\frac{-S_l^{(k)}}{H_{rl}} = min\{\frac{-S_l^{(k)}}{H_{rl}} : H_{rj} < 0 \wedge j \in N^{(k)}\} \tag{5.5.2}$$

we conclude that $\frac{-S_l^{(k)}}{H_{rl}} \geq 0$. If $H_{rj} \leq 0, j \in N^{(k+1)}$, then $S_j^{(k+1)} \geq 0$ holds as the summation of two vectors with positive entries. If $H_{rj} > 0, j \in N^{(k+1)}$, then the relation (3.1) is equivalent to

$$\frac{S_j^{(k)}}{H_{rj}} \geq \frac{S_l^{(k)}}{H_{rl}}$$

which is true according to relation (5.5.2) and consequently $S_j^{(k+1)} \geq 0$. Hence, if the initial basic partition is dual feasible, then PDIPSA constructs dual feasible partitions at every iteration.

Lemma 1: At every iteration of PDIPSA, the maximum ratio test yields $a \in (0, 1)$.

Proof: The condition $x_{B[i]} < 0$ combined with the relation $d_B = y_B - x_B$ and the facts that x_B is dual feasible and y_B is primal feasible, implies the relation $x_i < 0 \Rightarrow d_i > 0, i \in B$.

From the maximum ratio test we have

$$\begin{aligned} a = max\{\frac{-x_i}{d_i} : i \in B, d_i > 0 \wedge x_i < 0\} = \\ max\{\frac{|x_i|}{y_i - x_i} : i \in B, d_i > 0 \wedge x_i < 0\} = \\ max\{\frac{|x_i|}{|y_i| + |x_i|} : i \in B, d_i > 0 \wedge x_i < 0\} \end{aligned}$$

It is obvious from the above relation that $0 < a < 1$.

5.6 Revised Form and Solution of General LPs

As in most implementations, a revised form is used in this thesis and it is described in this Section. As it is well known, the computational time is one of the most significant issues for every mathematical software. Likewise, in linear algorithms, which are based on simplex algorithms, the inversion of the basis is the most time consuming operation.

The most common procedures lie on the idea to update the current basis through specific steps. Consequently, researchers have taken for granted that any computational efficient version of simplex algorithm use updating methods.

Our algorithm is not restricted in a specific technique but it can corporate with any known method for updating the basic inverse matrix $(A_B)^{-1}$ and the vectors x_B, w, s_N. In our implementation, we update the basis inverse with the help of a set of eta vectors, the procedure is a version of the product form of inverse. According to this technique, the new basis $(\overline{A_B})^{-1}$ can be calculated by updating the current basis inverse $(A_B)^{-1}$ based on information from the entering variable l. The new basis inverse can be computed by the equation:

$$(\overline{A_B})^{-1} = (A_B E)^{-1} = E^{-1}(A_B)^{-1} \tag{5.1}$$

where E^{-1} is the inverse of the eta matrix and it is calculated as follows:

$$E^{-1} = I - \frac{1}{h_{rl}}(h_l - e_l)(e_l)^T = \begin{bmatrix} 1 & \cdots & -h_{1l}/h_{rl} & \cdots & \cdots \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \vdots & \cdots & 1/h_{rl} & \cdots & \vdots \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \vdots & \cdots & -h_{ml}/h_{rl} & \cdots & 1 \end{bmatrix}$$

An illustrative example is necessary in order to clarify the above method. We will apply it to the following LP:

$$\begin{array}{lrcrcrcrr} \min & x_1 & + & x_2 & - & 4x_3 & & & \\ \text{s.t.} & 2\,x_1 & + & 2x_2 & + & 2x_3 & \leq & & 18 \\ & -\,x_1 & + & 1x_2 & + & x_3 & \leq & - & 4 \\ & 3\,x_1 & + & 3x_2 & - & 3x_3 & \leq & & 6 \\ & & & x_j \geqslant 0,\ j = 1,2,3 & & & & & \end{array}$$

First of all, the slack variables x_4, x_5 and x_6 are introduced. In matrix notation the above problem is written as follows:

$$\begin{aligned} min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned} \qquad (LP.1)$$

where:

$$C = \begin{bmatrix} 1 & 1 & -4 & 0 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 2 & 2 & 2 & 1 & 0 & 0 \\ -1 & 1 & 1 & 0 & 1 & 0 \\ 3 & 3 & -3 & 0 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 18 \\ 4 \\ 6 \end{bmatrix}$$

The first feasible partition is $B = [4 \quad 5 \quad 6], N = [1 \quad 2 \quad 3]$. Consequently:

$(A_B)^{-1} = A_B = [A_{.4}, A_{.5}, A_{.6}]$ or

$$(A_B)^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

According to our algorithm, x_6 is the leaving variable and x_1 the entering variable. Moreover, the leaving variable is the 3^{rd} column in B, so $h_l = h_3 = (A_B)^{-1}A_{.3} = [2 \quad 1 \quad 3]^T$. In the next step, the E matrix is calculated:

$$E^{-1} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 3 \end{bmatrix}$$

And finally, the new basis inverse:

$$(\overline{A_B})^{-1} = (A_B E)^{-1} = E^{-1}(A_B)^{-1}$$

$$(\overline{A_B})^{-1} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 3 \end{bmatrix}$$

Another significant aspect with crucial behaviour for the computational performance of our algorithm is the computation of the direction d. The computation of d is based on a very similar way to that of A_B. A different description of the above algorithm is created in a simpler approach that leads to a big-M problem.

Assume that x is a basic nonfeasible solution and y a feasible (not basic) solution of a LP. Consequently, the direction $d = x - y$ can be written as

$$Ad = A(y - x) = Ay - Ax = b - b = 0$$

If we combine this relation with $x_N = 0$, then we can produce

$$A_B y_B + A_N y_N - A_B x_B = 0$$

which is equal to

$$A_B d_B = -A_N y_N$$

and

$$d_B = -(A_B)^{-1} A_N y_N \tag{5.2}$$

As it was described previously in the geometrical representation of the algorithm, y' is the boundary point from which the ray $R = x + td : t \geq 0$ enters the feasible region of the LP, so:

$$y' = x + \lambda d \tag{5.3}$$

where λ is computed by the maximum ratio test of Step 2 of the algorithm. Also, $\bar{B} = B - \{k\} \cup \{p\}$, $\bar{N} = N - \{p\} \cup \{k\}$ is the next basic partition for the next iteration of the algorithm. Moreover, $N(t) = k$ and $B(r) = p$.

Combining $\bar{y}_k = 0$, $x_N = 0$ and 5.2:

$$\bar{y}_{\bar{N}} = \lambda(y_N - y_p e_t) \tag{5.4}$$

where e_t is the t^{th} unit vector of dimension $n - m$.

Likewise, $\bar{x}$ is the basic solution at the next iteration and E the eta matrix when the r^{th} column is replaced by the vector $h_p = (A_B)^{-1} a_p$. Moreover:

$$(A_{\bar{B}})^{-1} = E^{-1}(A_B)^{-1} \tag{5.5}$$

$$E_{-1}h_p = e_r \tag{5.6}$$

where e_r is the r^{th} vector with m elements.

As it was mentioned previously, the next direction is $\bar{d} = \bar{x} - \bar{y}$. And if we use relation 5.2, we have:

$$\bar{d}_{\bar{B}} = -(A_{\bar{B}})_{-1}A_{\bar{N}}y_{\bar{N}} \tag{5.7}$$

So:

$$A_{\bar{N}} = A_N + D \tag{5.8}$$

where D is a $m(n-z)$ zero matrix and its t^{th} column is replaced by the vector $a_p - a_k$.

Now, the combination of relations (5.2), (5.3), (5.4), (5.5), (5.6), (5.7) and (5.8) leads to

$$\begin{aligned}
\frac{1}{\lambda}\bar{d}_{\bar{B}} &= -\frac{1}{\lambda}(A_{\bar{B}})_{-1}A_{\bar{N}}y_{\bar{N}} \\
&= -\frac{1}{\lambda}E^{-1}(A_B)^{-1}(A_N + D)(y_N - y_p e_t)\lambda \\
&= -E^{-1}(A_B)^{-1}A_N y_N + -E^{-1}(A_B)^{-1}D y_N + E^{-1}(A_B)^{-1}A_N y_p e_t) + E^{-1}(A_B)^{-1}D y_p e_t) \\
&= E^{-1}d_B - E^{-1}(h_p - h_k)y_p + E^{-1}h_p y_p + E^{-1}(h_p - h_k)y_p \\
&= E^{-1}d_B + e_r y_p
\end{aligned} \tag{5.9}$$

Finally, $y_p = 0$ when the nonbasic index p becomes basic and $y_p > 0$, after the addition of y_p to the r^{th} component of the updated d_B. This is something that stems from the fact that $\bar{y}_k = 0$ for the leaving index $k = B(r)$.

A revised version of the algorithm has been presented. This revised version starts with a dual feasible partition (B, N) and a vector $y_N \geq 0$ where the direction d intersects the feasible region and we have $d_B = -A_N y_N$ and $d_N = y_N$. Moreover, the direction d is updated according to the relation 5.9 and we set $y_p = 0$ if $y_p \geq 0$.

A good idea for solving general LPs is to use the big-M method when the initial basis B is not dual feasible. In addition, the new problem is formulated as:

$$\begin{aligned} min \quad & (c_B)^T x_B + (c_N)^T x_N \\ s.t. \quad & A_B x_B + A_N x_N = b, \\ & e^T x_N + x_{N+1} = M, \\ & x, x_{n+1} \geq 0 \end{aligned} \tag{LP.1}$$

where M is a sufficiently big number, x_{n+1} a slack variable and $e \in \mathbb{R}^{n-m}$ is a vector of ones and x_{n+1} is a slack variable.

Moreover, $B \cup \{p\}$ is dual feasible to (LP.1) where p stems from the relation: $s_p = min\{s_j : j \in N\}$. Since $s_N = c_N - A^T w$ and $w^T = (c_B)^T (A_B)^{-1}$, we add an artificial variable x_{n+2} at the next step. Consequently, the new problem is:

$$\begin{aligned} min \quad & (c_B)^T x_B + (c_N)^T x_N + M' x_{n+2} \\ s.t. \quad & A_B x_B + A_N x_N + f x_{n+2} = b, \\ & e^T x_N + x_N + g x_{n+2} = M, \\ & x, x_{n+1}, x_{n+2} \geq 0 \end{aligned} \tag{LP.2}$$

where $F \in \mathbb{R}^m$, g is scalar and M' is a sufficiently big number and we can set $M = M'$. For the vector f and the scalar g, we have:

$$-e = Ae + F \tag{5.10}$$

$$-1 = n - m + g \tag{5.11}$$

5.7 Computational Study

In order to test the performance and the practical effectiveness of our algorithm, a computational study is presented. The computational study compares three algorithms; the revised simplex algorithm, the Exterior Point Simplex Algorithm (EPSA) and the Primal Dual Interior Point Simplex Algorithm (PDIPSA). In all instances the compared algorithms converge to the same solution.

TABLE 5.3: Results for randomly generated sparse LPs with dimension $n \times n$ and density 5%

	PDIPSA		EPSA		Simplex	
Density 5%	**CPU_time (sec)**	**Niter**	**CPU_time (sec)**	**Niter**	**CPU_time (sec)**	**Niter**
500×500	0.8502	406	3.7924	1,536	3.9617	1,660
600×600	1.4492	520	6.8079	2,042	8.5560	2,487
700×700	2.3447	684	11.4099	2,627	17.9035	3,706
800×800	3.5225	831	19.4112	3,238	33.4693	5,187
900×900	5.2479	1,010	27.5498	3,591	48.6587	6,034
$1,000 \times 1,000$	7.7650	1,218	39.6211	4,148	82.6693	7,962
$1,100 \times 1,100$	11.3865	1,131	59.9605	4,991	138.3463	10,331
$1,200 \times 1,200$	15.6400	1,552	75.3610	5,406	197.5266	12,372
$1,300 \times 1,300$	21.9790	1,749	103.0355	6,151	300.8046	15,533
$1,400 \times 1,400$	27.4468	1,916	125.4295	6,676	416.6580	18,237
$1,500 \times 1,500$	34.0847	2,071	151.6782	7,120	574.1572	21,802
$1,600 \times 1,600$	46.9520	2,428	186.7300	7,712	748.6919	24,400
$1,700 \times 1,700$	54.4100	2,576	225.6000	8,388	1,012.0000	28,642
$1,800 \times 1,800$	64.1663	2,696	270.6056	9,015	1,366.8000	34,026
$1,900 \times 1,900$	80.0582	2,988	328.8189	9,713	1,643.0000	36,365
$2,000 \times 2,000$	91.4950	3,175	375.5810	10,164	2,057.0000	41,011
Total	468.7981	27,139	2,011.3925	92,518	8,650.2031	269,755
Mean	29.2999	1,696	125.7120	5,782	540.6377	16,860

All implemented algorithms are running in MATLAB Professional R2010b. The computing environment includes an Intel(R) Core^{TM} i7 3.00 GHz (2 processors) and 16 GB RAM. The operating system was Microsoft Windows 7 Professional SP1 edition. All times in the following tables are measured in seconds with the "cputime" built-in function of MATLAB, including the time spent on scaling. All runs were made as a batch job. The constraint matrices for most practical LPs are sparse. Only nonzeros in the coefficient matrix are stored and operated on. Matlab stores sparse matrices on a single processor in Compressed Sparse Column (CSC) data structure. Two techniques are used in order to improve the performance of memory-bound code in MATLAB. These techniques are: (i) Pre-allocate arrays, and (ii) Store and access data in columns. In order to guarantee the accuracy, periodically we compute from scratch the inverse of the basis.

Three different density classes of LPs are used in the computational study: 5%, 10% and 20%. In each dimension, 10 different instances are tested. These LPs include only randomly generated matrices. The initial basis consists of only slack variables.

Tables 5.3 - 5.5 present the average execution time in seconds (CPU time) and the average number of iterations (niter) that each of the algorithms spent to solve the LPs. The last row of each table shows the mean value at each column.

TABLE 5.4: Results for randomly generated sparse LPs with dimension $n \times n$ and density 10%

	PDIPSA		EPSA		Simplex	
Density 10%	**CPU_time (sec)**	**Niter**	**CPU_time (sec)**	**Niter**	**CPU_time (sec)**	**Niter**
500×500	0.9641	420	3.7627	1,529	5.9177	2,281
600×600	1.7690	569	7.0434	1,996	13.9170	3,557
700×700	2.5460	653	11.8650	2,378	24.2287	4,658
800×800	4.2027	828	19.6858	2,817	41.2261	6,001
900×900	6.7361	1,049	26.6060	3,186	71.3937	7,869
$1,000 \times 1,000$	9.0184	1,109	38.6524	3,646	112.3835	9,647
$1,100 \times 1,100$	13.8825	1,290	55.6768	4,195	174.3109	11,878
$1,200 \times 1,200$	20.6296	1,539	67.4408	4,485	245.1684	13,907
$1,300 \times 1,300$	25.7230	1,644	97.9452	5,265	387.0143	17,688
$1,400 \times 1,400$	31.8000	1,762	111.0493	5,484	514.3469	19,897
$1,500 \times 1,500$	42.4307	2,048	139.9000	5,889	646.8230	22,105
$1,600 \times 1,600$	49.3322	2,077	166.4187	6,305	878.5583	25,665
$1,700 \times 1,700$	63.1305	2,351	208.5203	6,852	1,137.4827	28,744
$1,800 \times 1,800$	77.4171	2,567	251.1772	7,338	1,441.0763	32,301
$1,900 \times 1,900$	90.2918	2,680	296.2397	3,505	1,773.6488	35,054
$2,000 \times 2,000$	104.0885	2,803	332.9217	8,147	2,300.1632	40,319
Total	543.9622	25,389	1,834.9050	76,706	9,767.6594	281,571
Mean	33.9976	1,587	114.6816	4,794	610.4787	17,598

As it is obvious from Table 5.3, PDIPSA is clearly superior to the other two algorithms. The execution time of PDIPSA is much lower than EPSA and simplex algorithm. For example, PDIPSA needs almost 8 seconds to solve a $1,000 \times 1,000$ LP when EPSA needs almost 40 seconds and even worse the simplex algorithm needs 83 seconds to complete the computations. Likewise, the number of iterations is much greater for simplex algorithm and much less for PDIPSA.

In addition, in a $2,000 \times 2,000$ LP, the simplex algorithm requires on average 41,000 iterations, while PDIPSA needs 3,200 iterations and EPSA 10,200 iterations. In the same problem, PDIPSA needs almost 90 seconds, while the simplex algorithm needs 2,100 seconds. One may infer a growth in the relative speed of PDIPSA with respect to simplex and EPSA as problem sizes increase. It is clear that PDIPSA can lead to significant reductions of the execution time and the number of iterations.

In less sparse problems, PDIPSA continues to present its good performance. Its superiority is clear and with no doubt it is an efficient and promising algorithm. In all dimensions, PDIPSA has the shortest execution time and the smallest number of iterations.

In addition, in the last group of problems that includes denser instances, PDIPSA also has the best performance. In a $1,500 \times 1,500$ LP, PDIPSA requires on average 52 seconds,

TABLE 5.5: Results for randomly generated sparse LPs with dimension $n \times n$ and density 20%

	PDIPSA		EPSA		Simplex	
Density 20%	**CPU_time (sec)**	**Niter**	**CPU_time (sec)**	**Niter**	**CPU_time (sec)**	**Niter**
500×500	1.1747	403	3.4476	1,313	6.5975	2,270
600×600	2.2386	527	5.4132	1,548	13.6264	3,255
700×700	3.4632	634	11.9684	1,964	26.4489	4,469
800×800	5.2323	762	17.7794	2,302	43.8315	5,597
900×900	8.6003	905	25.5483	2,598	74.2187	7,137
$1,000 \times 1,000$	12.6954	1,043	35.2547	2,937	125.6610	8,928
$1,100 \times 1,100$	16.7529	1,088	50.4632	3,304	184.9891	10,597
$1,200 \times 1,200$	22.9415	1,246	63.3988	3,555	231.0437	11,292
$1,300 \times 1,300$	31.5106	1,415	87.3496	4,099	349.3742	13,595
$1,400 \times 1,400$	42.2669	1,1604	104.9980	4,250	441.9010	15,130
$1,500 \times 1,500$	51.6083	1,1733	129.9145	4,673	600.8061	17,246
$1,600 \times 1,600$	63.8855	1,850	171.6245	5,232	840.5429	20,385
$1,700 \times 1,700$	76.7353	1,963	210.9352	5,653	1,097.0000	23,049
$1,800 \times 1,800$	90.8534	2,068	251.1585	6,049	1,364.2000	24,985
$1,900 \times 1,900$	112.5890	2,291	297.1148	6,407	1,720.4000	27,967
$2,000 \times 2,000$	135.8129	2,462	362.7834	6,951	2,183.1451	30,733
Total	678.3608	21,994	1,829.1521	62,835	9,303.7861	226,635
Mean	42.3976	1,375	114.3220	39,27	581.4866	14,165

while EPSA needs 130 seconds and the simplex algorithm 600 seconds. Likewise, the number of iterations is much less in PDIPSA, which is able to find the optimal solution only in 1,730 iterations, while EPSA needs 4,670 iterations and the simplex algorithm almost 17,200 iterations.

Furthermore, Figures 5.4 - 5.6 show more clearly the superiority of PDIPSA over EPSA and the simplex algorithm. Figures 5.4 - 5.6 present the ratios (execution time of EPSA)/(execution time of PDIPSA), (iterations of EPSA)/(iterations of PDIPSA), (execution time of the simplex algorithm)/(execution time of PDIPSA) and (iterations of the simplex algorithm)/(iterations of PDIPSA) for the corresponding densities and dimensions. The above ratios indicate how many times PDIPSA is better than EPSA and the simplex algorithm.

Figure 5.4 shows that as the problem dimensions increase, the superiority of PDIPSA over EPSA does not vary a lot and we can claim that PDIPSA is five times faster than EPSA and it requires five times less iterations to complete its computations. On the other hand, the superiority of PDIPSA over the simplex algorithm increases proportionally with the dimension of the problems. In small LPs, PDIPSA is almost five times faster and requires five times less iterations than the simplex algorithm. On the other

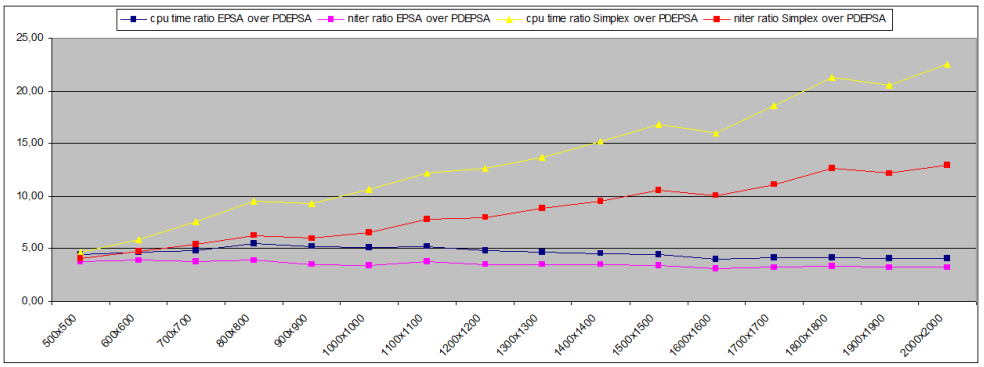

FIGURE 5.4: Speed-up ratios for $n \times n$ LPs and 5% density.

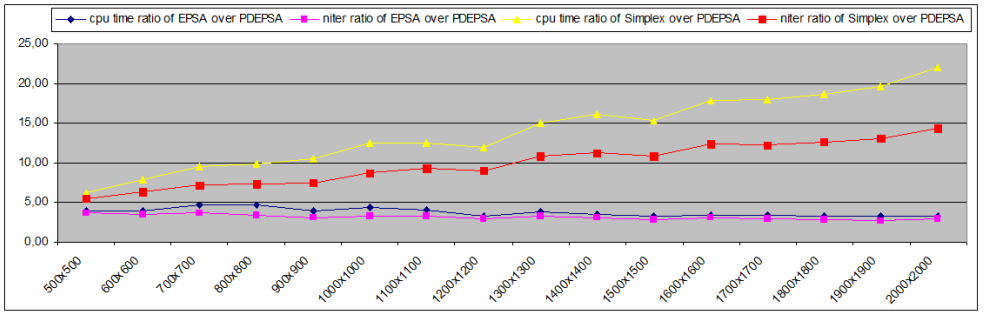

FIGURE 5.5: Speed-up ratios for $n \times n$ LPs and 10% density.

hand, in $2,000 \times 2,000$ LPs, PDIPSA is almost 23 times faster and it needs 13 times less iterations.

As the density of problems increases, PDIPSA continues its superiority and the results are very satisfactory. Comparatively to EPSA, PDIPSA is 4 times faster and it demands 4 times less iterations in small size LPs. However, in medium scale LPs the difference decreases to 3 times for the execution time and the number of iterations. Comparing to the simplex algorithm, the results are very similar with the previous group of problems (density 5%); the superiority of PDIPSA over simplex algorithm increases analogically with the dimension of problems.

In the last group of problems with density at the level of 20%, the results do not differ a lot from the previous. PDIPSA is almost 3 times better than EPSA both in the execution time and the number of iterations. On the other hand, PDIPSA is almost 6 times faster and it requires 6 times less iterations in small size LPs than the simplex algorithm. In medium scale LPs, the difference increases and PDIPSA is up to 16 times faster referring to the execution time and performs 10 times less iterations than the simplex algorithm.

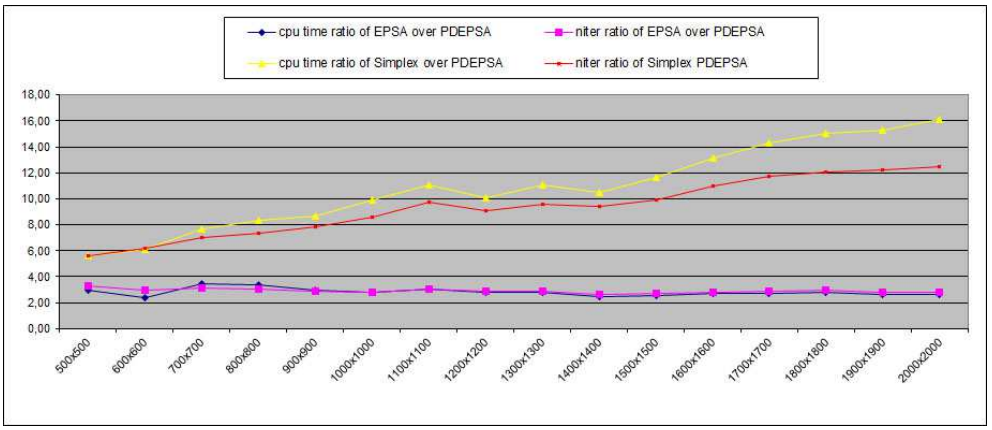

FIGURE 5.6: Speed-up ratios for $n \times n$ LPs and 20% density.

The results of the computational study clearly proved that PDIPSA is faster and requires fewer iterations than the simplex algorithm. Nevertheless, it is significant to mention the fact that as the density increases the superiority of PDIPSA decreases. More specific, in $2,000 \times 2,000$ dimension and 5% density, PDIPSA is 23 times faster than the simplex algorithm, while in 20% density, PDIPSA is almost 16 times faster. Despite the fact that this looks like a drawback, in fact it is not. In real world applications, the vast majority of LPs are extremely sparse problems, and their density does not reach the level of 5%. Consequently, this computational performance of PDIPSA is a remarkable advantage and clearly shows its worth.

5.8 Conclusions

The current Section investigated the practical behavior of PDIPSA. PDIPSA attempts to avoid the problem of stalling and cycling. The elimination of these disadvantages can lead to a more effective algorithm with better computational performance.

The computational results shows that PDIPSA is superior to EPSA and the simplex algorithm in all densities and dimensions. PDIPSA shows the same behavior comparing to EPSA in all dimensions of LPs. Their difference is not affected by the different dimensions of LPs. On the other hand, the performance of PDIPSA is getting better comparing to the simplex algorithm while the size of the LPs increases. Moreover, the computational performance of PDIPSA is much better in very sparse problems. The difference between simplex algorithm and PDIPSA is greater in sparser problems. This is a strong and significant advantage of PDIPSA because in real problems the level of density is very low.

Chapter 6

Hybrid Algorithm

6.1 Introduction

Interior point methods and simplex-type algorithms are the most widely-used algorithms for solving LPs. During the last decades, researchers proposed more efficient implementations of linear programming algorithms. Other efforts focused on the parallelization of linear programming algorithms, on CPUs [48] [49] [95] [96] [113] and on GPUs [67] [80] [100] [101], and the combination of different linear programming algorithms [4] [12] [13] [41] [65] [115]. The proposed algorithm belongs to the latter category. The idea to combine two types of LP algorithms is not new. Kortanek & Zhu [65] proposed a pivoting procedure from an interior point to a boundary point without worsing the objective value. This procedure can be performed in finite steps but may not be polynomial. Bixby et al. [12] and Bixby & Saltzman [13] proposed a combination of an IPM with the simplex algorithm. The hybrid procedure starts running an IPM first and later switches to the simplex algorithm. Andersen & Ye [4] proposed a combination of an IPM with a pivoting algorithm using a totally different idea from [12] [13]; they construct an artificial LP, which approximates the original problem, in any iteration of an IPM. Finally, they apply Megiddo's procedure [78] to compute an optimal basis of the approximate problem in n pivot steps. Al-Najjar and Malakotti [3] proposed hybrid-LP, a method for solving LPs using both interior and boundary paths. Their method uses an interior direction to pass to an improved basic feasible solution. Then, the simplex algorithm can be applied in order to reach an optimal solution. The computational results of the hybrid-LP method are very promising. Pan [90] proposed a pivoting algorithm using the affine-scaling technique. This method produces a sequence of interior points as well as a sequence of vertices, until reaching an optimal vertex. Triantafyllidis [115] proposed a non-monotonic variant of the exterior point algorithmic family by combining EPSA

with IPMs. Glavelis et al. [41] proposed also a combination of an IPM and an EPSA algorithm.

This Chapter presents a hybrid algorithm that combines the strengths of interior point methods and exterior point simplex algorithms [40]. The idea of combining these two different types of methods stemmed from the observation that IPMs are able to spot very fast feasible solutions with good objective values, but they need a relatively long time to converge to the optimal solution. In order to take full advantage of EPSA, we used a variation of a Primal-Dual Exterior Point Simplex Algorithm (PDEPSA). Primal-dual algorithms can deal more effectively with the problems of stalling and cycling and as a result, they are able to improve the performance of EPSA. This variation, which was presented in Chapter 5, is called Primal-Dual Interior Point Simplex Algorithm (PDIPSA) since the algorithm computes a direction to the feasible region according to the interior point that was found by an IPM. The IPM, which we use in our hybrid algorithm, is Mehrotra's Predictor-Corrector method, an infeasible primal-dual IPM.

The main advantage of this hybrid algorithm is that it exploits the strengths of both IPM and PDIPSA. In the first iterations, IPM moves inside the feasible area and computes interior points. At this point, the proposed algorithm uses PDIPSA to find the optimal solution in less expensive iterations. The goal of the proposed implementation is twofold: (i) improve the performance of EPSA, and (ii) find an optimal basic solution starting from an interior point (purification process). The latter goal is preferable for a couple of reasons [4]. First of all, a basic solution has generally fewer nonzero elements than a solution in the interior of the optimal face, which is desirable when LP relaxations of integer programming problems are solved. Secondly, an optimal basic solution can be used to warm-start simplex-type algorithms to solve closely related LPs.

This Chapter is organized as follows. Section 6.2 includes the description of the general framework of the proposed algorithm. In Section 6.3, we give the proof of correctness. In order to gain an insight into the practical behavior of the proposed algorithm, we have performed a computational study on a set of benchmark problems (Netlib, Kennington, Mészáros). These results are presented in Section 6.4. Finally, the conclusions of this Chapter are outlined in Section 6.5.

6.2 Description of the Algorithm

A hybrid algorithm is proposed in this Section. This algorithm combines an IPM with an EPSA and more specifically Mehrotra's Predictor-Corrector method with PDIPSA; the most efficient IPM and EPSA, respectively. The main goal of this combination is to

adopt the strengths of each algorithm and to eliminate their disadvantages. According to this thought, the hybrid algorithm executes Mehrotra's Predictor-Corrector method for a few iterations in order to start PDIPSA from a "good" interior point. Then, PDIPSA completes the calculations and solves the LP.

PDIPSA demands a starting interior point; this point is computed by Mehrotra's Predictor-Corrector method. Moreover, the interior point is necessary for the calculation of the direction d, which reveals the leaving variable. Another significant issue is that a "good" initial interior point can lead to significant less iterations of PDIPSA. Consequently, if the initial point is closer to optimal vertex, then the optimal solution will be sooner spotted by PDIPSA. This is the main reason for using an IPM at the first stage. IPM is able to move to an interior point close to the optimal vertex at the first iterations. Furthermore, we avoid a significant disadvantage of IPMs, their late convergence at the last iterations. In this step, PDIPSA receives the interior point and continues finding the solution. Taking under consideration IPMs' relative large computational cost per iteration and late convergence at the last iterations, our hybrid approach takes full advantage of IPMs at the same time of giving a "good" interior point to PDIPSA.

The hybrid algorithm is described formally as follows:

TABLE 6.1: Hybrid Approach Combining Mehrotra's Predictor-Corrector Method and PDIPSA

Step 1. *(IPM).* Perform a few iterations with Mehrotra's Predictor-Corrector method in order to compute a "good" interior point y. **Step 2.** *(PDIPSA).* A) Initialize PDIPSA with a dual feasible basic partition (B, N) and the interior point y taken from Step 1. B) Iteratively, PDIPSA continues until it computes the optimal solution as it was described in Chapter 5.

Next, we will demonstrate the hybrid algorithm with an example. The LP that will be solved is the following:

$$
\begin{array}{llrcrcrcl}
\min & z = & 8x_1 & + & 4x_2 & - & 6x_3 & & \\
s.t. & & x_1 & + & x_2 & + & x_3 & \leq & 2 \\
& & 3x_1 & + & x_2 & - & x_3 & \geq & 3 \\
& & 3x_1 & + & 2x_2 & - & x_3 & \geq & 5 \\
x_j \geq 0, & (j = 1, 2, 3) & & & & & & &
\end{array}
$$

Let's convert the LP in its standard form:

$$\begin{array}{llrcrcrcrcrcrl} \min & z = & 8x_1 & + & 4x_2 & - & 6x_3 & & & & & & & \\ s.t. & & x_1 & + & x_2 & + & x_3 & + & x_4 & & & & = & 2 \\ & & 3x_1 & + & x_2 & - & x_3 & & & - & x_5 & & = & 3 \\ & & 3x_1 & + & 2x_2 & - & x_3 & & & & & - & x_6 = & 5 \\ x_j \geq 0, & (j = 1,2,3,4,5,6) & & & & & & & & & & & & \end{array}$$

So, the matrices and vectors that will be given as input to the hybrid algorithm are the following:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 3 & 1 & -1 & 0 & -1 & 0 \\ 3 & 2 & -1 & 0 & 0 & -1 \end{bmatrix}, c = \begin{bmatrix} 8 \\ 4 \\ -6 \\ 0 \\ 0 \\ 0 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$$

We start with the basic partition $B = [4, 5, 6]$ and $N = [1, 2, 3]$. The initial basis is not dual feasible ($x_B \ngeq 0$):

$$A_B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = A_B^{-1}$$

$$x_B = A_B^{-1} b = \begin{bmatrix} 2 \\ -3 \\ -5 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

$$s_N = c_N^T - w A_N = \begin{bmatrix} 8 & 4 & -6 \end{bmatrix}$$

Using the big-M method, the auxiliary LP is formulated:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 3 & 1 & -1 & 0 & -1 & 0 & 0 \\ 3 & 2 & -1 & 0 & 0 & -1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, c = \begin{bmatrix} 8 \\ 4 \\ -6 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 3 \\ 10000 \end{bmatrix}$$

According to Dantzig's rule, the entering variable is x_3 since it has the most negative $\overline{s_l}$ (-6). We update the basic partition, $B = [4, 5, 6, 3]$ and $N = [1, 2, 7]$. Finally, we update the needed matrices and vectors:

$$A_B^{-1} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x_B = A_B^{-1} b = \begin{bmatrix} -9,998 \\ -10,003 \\ -10,005 \\ 10,000 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 0 & -6 \end{bmatrix}$$

$$s_N = c_N^T - w A_N = \begin{bmatrix} 14 & 10 & 6 \end{bmatrix}$$

$$d_B = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

At the next step, the leaving variable $x_k = x_{B[r]}$ is calculated from the maximum ratio test:

$$a = \frac{x_{B[r]}}{-d_{B[r]}} = max \left\{ \frac{x_i}{-d_i} : i \in B, d_i > 0 \wedge x_i < 0 \right\} = \frac{-10,005}{-1}$$

Hence, the leaving variable is x_6.

Using Mehrotra's heuristic (3.9), we can calculate an initial interior point:

$$x = \begin{bmatrix} 9,792 \\ 10,137 \\ 11,860 \\ 5,309 \\ 9,102 \\ 9,963 \\ 15,307 \end{bmatrix}, \; w = \begin{bmatrix} -0.7931 \\ 1.3678 \\ 1.8621 \\ -0.7931 \end{bmatrix}, \; s = \begin{bmatrix} 2.8086 \\ 3.4063 \\ 1.7282 \\ 3.7052 \\ 4.2799 \\ 4.7741 \\ 3.7052 \end{bmatrix}$$

In the first iteration, we perform all the steps of Mehrotra's Predictor-Corrector steps described in Chapter 3 and we finally calculate a new interior point:

$$x = \begin{bmatrix} 2,077 \\ 1,357 \\ 1,038 \\ 27 \\ 4,300 \\ 4,531 \\ 10,025 \end{bmatrix}, w = \begin{bmatrix} -3.9893 \\ 1.6228 \\ 1.8532 \\ -0.6674 \end{bmatrix}, s = \begin{bmatrix} 2.2286 \\ 3.3275 \\ 2.1328 \\ 3.9893 \\ 1.6228 \\ 1.8532 \\ 0.6674 \end{bmatrix}$$

Hence, the hybrid algorithm will start from the feasible basic list $B = [4, 5, 6, 3]$ and the interior point:

$$y_B = \begin{bmatrix} 26.5 \\ 4,299.5 \\ 4,530.7 \\ 1,038 \end{bmatrix}$$

Next, we initialize sets P and Q, $P = N = [1, 2, 7]$ and $Q = \emptyset$. Then, we compute vector H_{rN}:

$$H_{rN} = (A_B)_r^{-1} A_N = \begin{bmatrix} -4 & -3 & -1 \end{bmatrix}$$

Next, we find the entering variable:

$$\frac{-s_l}{H_{rN}} = \min\left\{ \frac{-s_l}{H_{rN}} : H_{rj} \wedge j \in N \right\} = \frac{-10}{-3}$$

The entering variable is x_2.

So, we update sets P and Q, $P = [1, 7]$ and $Q = [6]$, and the basic partition, $B = [4, 5, 2, 3]$ and $N = [1, 6, 7]$. Finally, we update all needed matrices and vectors:

$$E^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -2/3 & 0 \\ 0 & 0 & -1/3 & 0 \\ 0 & 0 & 1/3 & 1 \end{bmatrix}$$

$$\left(A_{\overline{B}}\right)^{-1} = E^{-1} A_B^{-1} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & -1 & 2/3 & -1/3 \\ 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & -1/3 & 2/3 \end{bmatrix}$$

$$x_B = A_B^{-1} b = \begin{bmatrix} -9,998 & -3,3333 & 3356,665 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} = \begin{bmatrix} 0 & 0 & 10/3 & -8/3 \end{bmatrix}$$
$$s_N = c_N^T - wA_N = \begin{bmatrix} 2/3 & 10/3 & 8/3 \end{bmatrix}$$
$$d_B = y_B - x_B = \begin{bmatrix} 10,025 \\ 7,633 \\ 1,196 \\ -5,627 \end{bmatrix}$$

In the first iteration, we perform the test of optimality. The current basic partition is not dual feasible ($x_B \ngeq 0$). So, we select the leaving variable:

$$a = \frac{x_{B[r]}}{-d_{B[r]}} = max\left\{\frac{x_i}{-d_i} : i \in B, d_i > 0 \wedge x_i < 0\right\} = \frac{-9,998}{-10,025}$$

The leaving variable is x_4.

Next, we compute the next interior point:

$$a' = \frac{a+1}{2} = 0.9987$$
$$y_B = x_B + a'd_B = \begin{bmatrix} 13.3 \\ 4,289.4 \\ 4,529.1 \\ 1,045.4 \end{bmatrix}$$

Then, we compute vector H_{rN}:

$$H_{rN} = (A_B)_r^{-1} A_N = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}$$

Next, we find the entering variable:

$$\frac{-s_l}{H_{rN}} = \min\left\{\frac{-s_l}{H_{rN}} : H_{rj} \wedge j \in N\right\} = \frac{-8/3}{-1}$$

The entering variable is x_7.

So, we update sets P and Q, $P = [1]$ and $Q = [6, 4]$, and the basic partition, $B = [7, 5, 2, 3]$ and $N = [1, 6, 4]$. Finally, we update all needed matrices and vectors:

$$E^{-1} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ -1/3 & 1 & 0 & 0 \\ 1/3 & 0 & 1 & 0 \\ 2/3 & 0 & 0 & 1 \end{bmatrix}$$

$$\left(A_{\overline{B}}\right)^{-1} = E^{-1}A_B^{-1} = \begin{bmatrix} -1 & 0 & 0 & 1 \\ -1/3 & -1 & 2/3 & 0 \\ 1/3 & 0 & 1/3 & 0 \\ 2/3 & 0 & -1/3 & 0 \end{bmatrix} \quad x_B = A_B^{-1}b = \begin{bmatrix} 9,998 \\ -0.3 \\ 2.3 \\ -0.3 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} = \begin{bmatrix} -8/3 & 0 & 10/3 & -8/3 \end{bmatrix}$$

$$s_N = c_N^T - wA_N = \begin{bmatrix} 2/3 & 10/3 & 8/3 \end{bmatrix}$$

$$d_B = y_B - x_B = \begin{bmatrix} -9,984.7 \\ 4,289.7 \\ 4,526.8 \\ 1,045.8 \end{bmatrix}$$

In the second iteration, we perform the test of optimality. The current basic partition is not dual feasible ($x_B \ngeq 0$). So, we select the leaving variable:

$$a = \frac{x_{B[r]}}{-d_{B[r]}} = max\left\{\frac{x_i}{-d_i} : i \in B, d_i > 0 \wedge x_i < 0\right\} = \frac{-0.3}{-1,045.8}$$

The leaving variable is x_3.

Next, we compute the next interior point:

$$a' = \frac{a+1}{2} = 0.5002$$

$$y_B = x_B + a'd_B = \begin{bmatrix} 5,004 \\ 2,145.2 \\ 2,266.4 \\ 522.7 \end{bmatrix}$$

Then, we compute vector H_{rN}:

$$H_{rN} = (A_B)_r^{-1}A_N = \begin{bmatrix} -1/3 & 1/3 & 2/3 \end{bmatrix}$$

Next, we find the entering variable:

$$\frac{-s_l}{H_{rN}} = \min\left\{\frac{-s_l}{H_{rN}} : H_{rj} \wedge j \in N\right\} = \frac{-2/3}{-1/3}$$

The entering variable is x_1.

So, we update sets P and Q, $P = \emptyset$ and $Q = [6, 4, 3]$, and the basic partition, $B = [7, 5, 2, 1]$ and $N = [3, 6, 4]$. Finally, we update all needed matrices and vectors:

$$E^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -4 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & -3 \end{bmatrix}$$

$$\left(A_{\overline{B}}\right)^{-1} = E^{-1}A_B^{-1} = \begin{bmatrix} -1 & 0 & 0 & 1 \\ -3 & -1 & 2 & 0 \\ 3 & 0 & -1 & 0 \\ -2 & 0 & 1 & 0 \end{bmatrix} \quad x_B = A_B^{-1}b = \begin{bmatrix} 9,998 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$w = c_B^T A_B^{-1} = \begin{bmatrix} -8/3 & 0 & 10/3 & -8/3 \end{bmatrix}$$

$$s_N = c_N^T - wA_N = \begin{bmatrix} 2 & 4 & 4 \end{bmatrix}$$

$$d_B = y_B - x_B = \begin{bmatrix} -4,994.0 \\ 2,144.2 \\ 2,265.4 \\ 521.7 \end{bmatrix}$$

In the third iteration, we perform the test of optimality. The current basic partition is dual feasible ($x_B \geq 0$). As a result, we calculate the solution vector and the value of the objective function:

$$x = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$z = c_B^T x_B = 12$$

6.3 Proof of Correctness

To prove the correctness of the algorithm, it suffices to show that every basic partition, which is constructed by PDIPSA, is dual feasible and the computation of the maximum ratio test is well defined.

Theorem 1: If the initial basic partition of PDIPSA is dual feasible, then every consecutive partition is dual feasible.

Proof: The proof is by induction on the number of iterations. Denote by k the number of iterations. It is obvious from Step $0A$ of PDIPSA that for $k = 1$ the relations $S_j^{(1)} \geq 0\,, j \in N^{(1)}$, and $S_j^{(1)} = 0, j \in B^{(1)}$, hold. Suppose now that the relation $S_j^{(k)} \geq 0$, $j \in N^{(k)}$, holds. Let $(B^{(k+1)}, N^{(k+1)})$ be the new basic partition and $S_j^{(k+1)}$, $j \in N^{(k+1)}$,

the corresponding dual slack variables. The dual slack variables can be computed by the relation

$$S_j^{(k+1)} = S_j^{(k)} - \frac{S_l^{(k)}}{H_{rl}} H_{rj}, j \in N^{(k+1)} \tag{3.1}$$

where H_{rj}, $j \in N^{(k+1)}$, is the pivot row. From the choice of the entering variable x_l

$$\frac{-S_l^{(k)}}{H_{rl}} = min\{\frac{-S_l^{(k)}}{H_{rl}} : H_{rj} < 0 \wedge j \in N^{(k)}\} \tag{3.2}$$

we conclude that $\frac{-S_l^{(k)}}{H_{rl}} \geq 0$. If $H_{rj} \leq 0, j \in N^{(k+1)}$, then $S_j^{(k+1)} \geq 0$ holds as the summation of two vectors with positive entries. If $H_{rj} > 0, j \in N^{(k+1)}$, then relation (3.1) is equivalent to

$$\frac{S_j^{(k)}}{H_{rj}} \geq \frac{S_l^{(k)}}{H_{rl}}$$

which is true according to relation (3.2) and consequently $S_j^{(k+1)} \geq 0$. Hence, if the initial basic partition is dual feasible, then PDIPSA constructs dual feasible partitions at every iteration.

Lemma 1: At every iteration of PDIPSA, the maximum ratio test yields $a \in (0, 1)$.

Proof: The condition $x_{B[i]} < 0$ combined with the relation $d_B = y_B - x_B$ and the facts that x_B is dual feasible and y_B is primal feasible, implies the relation $x_i < 0 \Rightarrow d_i > 0, i \in B$.

From the maximum ratio test we have

$$\begin{aligned} a = max\{\frac{-x_i}{d_i} : i \in B, d_i > 0 \wedge x_i < 0\} = \\ max\{\frac{|x_i|}{y_i - x_i} : i \in B, d_i > 0 \wedge x_i < 0\} = \\ max\{\frac{|x_i|}{|y_i| + |x_i|} : i \in B, d_i > 0 \wedge x_i < 0\} \end{aligned}$$

It is obvious from the above relation that $0 < a < 1$.

6.4 Computational Study

In this Section, we present the results from a computational study that we conducted to demonstrate the efficiency of the proposed hybrid algorithm. The computational comparison has been performed on a quad-processor Intel Core i7 3.4 GHz with 32 Gbyte of main memory and 8 cores, a clock of 3.7 GHz, an L1 code cache of 32 KB per core, an L1 data cache of 32 KB per core, an L2 cache of 256 KB per core, an L3 cache of 8 MB and a memory bandwidth of 21 GB/s, running under Microsoft Windows 8 64-bit. All algorithms have been implemented using MATLAB Professional R2015b. Some linear algebra built-in functions were also used to code the algorithms (e.g., inverse of an array, multiplication of two arrays, multiplication of array and vector, and the mldivide operator for solving systems of linear equations). Execution times have been measured in seconds using *tic* and *toc* MATLAB's built-in functions. For each instance, we averaged times over 10 runs. All runs were executed as a batch job.

Totally, 83 LPs were considered from the Netlib set (optimal and Kennington LPs) [16] [38] and the problematic, misc and stochlp sections of Mészáros collection [81]. The Netlib library is a well known suite containing many real world LPs. Ordóñez and Freund [86] have shown that 71% of the Netlib LPs are ill-conditioned. Hence, numerical difficulties may occur. We implemented a MPS reader to read MPS files and convert data into MATLAB mat files. All runs terminated with correct optimal objective values. Table 6.2 presents some useful information about the test bed, which was used in the computational study. The first column includes the name of the problem, the second the number of constraints, the third the number of variables, the fourth the nonzero elements of matrix A and the fifth the optimal objective value.

TABLE 6.2: Statistics of the Netlib (optimal and Kennington LPs) and Mészáros LPs

Name	**Constraints**	**Variables**	**Nonzeros A**	**Optimal objective value**
aa4	426	7,195	52,121	2.59E+04
aa5	801	8,308	65,953	5.37E+04
aa6	646	7,292	51,728	2.70E+04
adlittle	56	97	383	2.25E+05
afiro	27	32	83	-4.65E+02
agg	488	163	2,410	-3.60E+07
agg2	516	302	4,284	-2.02E+07
agg3	516	302	4,300	1.03E+07
aircraft	3,754	7,517	20,267	1.57E+03
beaconfd	173	262	3,375	3.36E+04

blend	74	83	491	-3.08E+01
bnl2	2,324	3,489	13,999	1.81E+03
car4	16,384	33,052	63,724	3.55E+01
cari	400	1,200	152,800	5.82E+02
cr42	905	1,513	6,614	2.80E+01
cre-a	3,516	4,067	14,987	2.36E+07
d6cube	415	6,184	37,704	3.15E+02
fffff800	524	854	6,227	5.56E+05
fit1d	24	1,026	13,404	-9.15E+03
forplan	161	421	4,563	-6.64E+02
fxm2-6	3,900	5,602	32,239	1.84E+04
fxm3-6	6,200	9,492	54,589	1.86E+04
gen	769	2,560	63,085	0.00E+00
gen1	769	2,560	63,085	0.00E+00
gfrd-pnc	616	1,092	2,377	6.90E+06
iiasa	669	2,970	6,648	2.63E+08
israel	174	142	2,269	-8.97E+05
jendrec1	2,109	4,228	89,608	7.03E+03
lotfi	153	308	1,078	-2.53E+01
maros-r7	3,136	9,408	144,848	1.50E+06
nsic1	451	463	2,853	-9.17E+06
nsic2	465	463	3,015	-8.20E+06
nsir1	4,407	5,717	138,955	-2.89E+07
nsir2	4,453	5,717	150,599	-2.72E+07
osa-07	1,118	23,949	143,694	5.36E+05
osa-14	2,337	52,460	314,760	1.11E+06
osa-30	4,350	100,024	600,138	2.14E+06
p05	5,090	9,500	58,955	3.15E+02
p010	10,090	19,000	117,910	1.12E+06
pgp2	4,034	9,220	18,440	4.47E+02
primagaz	1,554	10,836	21,665	1.07E+09
r05	5,190	9,500	103,955	5.58E+05
rail507	507	63,009	409,349	1.72E+02
rail516	516	47,311	314,896	1.82E+02
rail582	582	55,515	401,708	2.10E+02
rat1	3,136	9,408	88,267	2.00E+06
rat5	3,136	9,408	137,413	3.08E+06
rat7a	3,136	9,408	268,908	2.07E+06
recipe	91	180	663	-2.67E+02

rosen2	1,032	2,048	46,504	-5.44E+04
rosen7	264	512	7,770	-2.03E+04
rosen8	520	1,024	15,538	-4.21E+04
rosen10	2,056	4,096	62,136	-1.74E+05
sc105	105	103	280	-5.22E+01
sc205	205	203	551	-5.22E+01
sc205-2r-400	8,813	8,814	24,030	-1.01E+01
sc205-2r-800	17,613	17,614	48,030	-1.01E+01
sc205-2r-1600	35,213	35,214	96,030	0.00E+00
sc50a	50	48	130	-6.46E+01
sc50b	50	48	118	-7.00E+01
scagr25	471	500	1,554	-1.48E+07
scagr7	129	140	420	-2.33E+06
scagr7-2b-64	9,743	10,260	32,298	-8.33E+05
scagr7-2r-216	8,223	8,660	27,042	-8.34E+05
scagr7-2r-432	16,431	17,300	54,042	-8.34E+05
scfxm1	330	457	2,589	1.84E+04
scfxm1-2b-64	19,036	28,914	106,919	2.88E+03
scfxm3	990	1,371	7,777	5.49E+04
scrs8	490	1,169	3,182	9.04E+02
sctap1	300	480	1,692	1.41E+03
sctap2	1,090	1,880	6,714	1.72E+03
sctap3	1,480	2,480	8,874	1.42E+03
share1b	117	225	1,151	-7.66E+04
share2b	96	79	694	-4.16E+02
ship12l	1,151	5,427	16,170	1.47E+06
ship12s	1,151	2,763	8,178	1.49E+06
slptsk	2,861	3,347	72,465	2.99E+01
standata	359	1,075	3,031	1.26E+03
stocfor1	117	111	447	-4.11E+04
stocfor2	2,157	2,031	8,343	-3.90E+04
stocfor3	16,675	15,695	64,875	-4.00E+04
testbig	17,613	31,223	6,639	-6.04E+01
zed	116	43	567	-1.51E+04

Since our primary aim is to improve the computational performance of PDIPSA, we compare the proposed algorithm, HYBRID, with PDIPSA. As described in Section 6.2, HYBRID uses Mehrotra's Predictor-Corrector method to calculate an interior point. In

addition, we have implemented the primal revised simplex algorithm (RSA) and we use it in the computational study as a reference point for the comparison. All algorithms use the same preprocessing, scaling and basis update methods. Their major difference is how they select the entering and leaving variable. In the simplex implementation, we use Dantzig's pivoting rule; however, if degeneracy is detected, then simplex will automatically switch to the steepest-edge pivoting rule and the problem will be perturbed. We selected to use Dantzig's pivoting rule since the steepest edge variant that we implemented is quite expensive and thus, we use it only when degeneracy is detected. When stalling occurs, our algorithm automatically perturbs the upper and lower bounds by adding a small positive number (we use a value of 1e-6) to the bounds. After the solution of the perturbed problem, we remove the perturbation by resetting the problem to its original values. All algorithms share the same data structures and sparse linear algebra routines. All LPs have been presolved and scaled prior to the execution of each algorithm using the equilibration scaling technique. The basis update method used in all algorithms is the Product Form of the Inverse [97]. In order to guarantee the accuracy, we compute from scratch the inverse of the basis every 80 iterations.

Table 6.3 presents the execution time and the number of iterations of each algorithm over the Netlib and Mészáros set of LPs, while figure 6.1 presents the performance profile based on the execution time of the algorithms. The performance profile is displayed in logarithmic scale with base 2 using a tool developed in [31]. We also report the number of iterations performed by Mehrotra's Predictor-Corrector method ("Interior Iter") in order to initialize the proposed algorithm (HYBRID). A limit of $1,000$ seconds was set, so symbol "-" denotes that this algorithm did not find an optimal solution in the specific time interval. HYBRID is able to solve all instances, while PDIPSA did not solve three instances (rail507, rail516, and rail582) and RSA did not solve eleven instances (aa5, aa6, d6cube, jendrec1, nsir2, p010, rail507, rail516, rail582, scfxm1-2b-64, and slptsk). In addition, we report the geometric mean of the execution time and number of iterations for all algorithms. We also report the geometric mean of the execution time and the number of iterations for the instances solved by all three algorithms (shown in parentheses in the last row of the table).

When considering all problems, HYBRID is 1.53× faster than PDIPSA and 2.1× faster than RSA. Moreover, HYBRID performs 1.36× less iterations than PDIPSA and 1.69× less iterations than RSA. When considering only the instances that all three algorithms can solve, HYBRID is 1.49× faster than PDIPSA and 1.73× faster than RSA. Moreover, HYBRID performs 1.34× less iterations than PDIPSA and 1.57× less iterations than RSA. Taking also into account that PDIPSA and RSA fail to solve some instances, HYBRID is superior to PDIPSA and RSA on these benchmark instances. Finally,

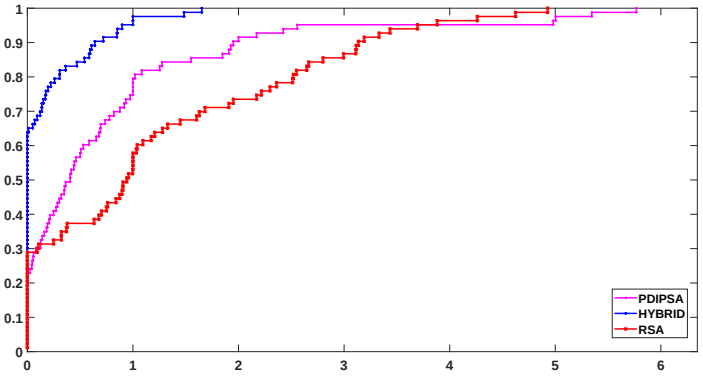

FIGURE 6.1: Performance profile based on the execution time of the three algorithms

HYBRID has better or equal performance than PDIPSA and RSA on 70 (84.3%) and 65 (78.3%) instances, respectively.

TABLE 6.3: Execution time and number of iterations

Problem	**PDIPSA**		**HYBRID**			**RSA**	
	Time	**Iter**	**Time**	**Iter**	**Interior Iter**	**Time**	**Iter**
aa4	17.31	4,565	15.49	4,430	2	49.70	14,833
aa5	102.29	8,096	77.24	6,826	2	-	-
aa6	46.26	5,377	40.54	5,053	2	-	-
adlittle	0.03	77	0.03	89	2	0.03	97
afiro	0.01	17	0.01	14	3	0.01	14
agg	0.10	149	0.06	77	7	0.06	83
agg2	0.38	296	0.18	173	5	0.10	138
agg3	0.36	291	0.20	195	5	0.10	138
aircraft	5.44	1,989	5.27	1,675	1	10.84	4,034
beaconfd	0.04	86	0.04	83	4	0.02	47
blend	0.02	59	0.01	37	5	0.02	76
bnl2	12.90	2,149	11.37	1,630	20	71.19	3,921
car4	3.86	2,798	3.50	2,163	10	17.21	10,349
cari	0.87	459	1.62	494	1	1.69	1,116
cr42	1.06	625	0.87	571	5	1.13	581
cre-a	11.93	2,993	10.34	2,892	1	72.01	4,899
d6cube	146.95	7,528	115.43	6,243	15	-	-
fffff800	0.20	170	0.21	191	1	0.25	399
fit1d	4.54	626	2.53	651	3	2.24	1,773
forplan	0.28	244	0.22	200	3	0.07	180

fxm2-6	4.19	1,303	2.59	895	3	8.01	1,868
fxm3-6	98.51	5,168	71.56	2,673	10	128.05	6,532
gen	33.06	8,162	16.83	5,278	1	96.48	15,304
gen1	35.75	8,162	19.47	6,434	2	210.56	20,112
gfrd-pnc	0.64	335	0.53	331	6	1.61	550
iiasa	2.97	2,448	2.19	1,953	8	2.83	1,966
israel	0.16	313	0.11	166	8	0.10	262
jendrec1	333.20	9,230	113.76	4,005	5	-	-
lotfi	0.09	195	0.07	180	3	0.17	123
maros-r7	50.87	2,631	43.84	2,419	10	82.11	3,310
nsic1	0.31	552	0.13	307	1	0.14	405
nsic2	0.19	283	0.18	270	6	0.16	432
nsir1	113.88	5,206	21.22	2,631	2	32.89	3,547
nsir2	46.48	2,845	32.86	2,676	5	-	-
osa-07	7.60	897	6.67	631	15	5.58	719
osa-14	31.19	1,879	34.30	1,421	10	52.83	2,512
osa-30	138.23	3,881	133.10	2,813	10	284.17	4,889
p05	27.97	1,829	34.60	1,820	1	536.28	3,118
p010	175.27	3,648	190.61	3,539	1	-	-
pgp2	10.17	5,138	9.62	4,855	2	17.91	6,024
primagaz	12.30	2,253	11.95	2,208	1	109.09	6,098
r05	34.33	1,763	35.56	1,761	1	506.04	3,101
rail507	-	-	266.36	3,635	5	-	-
rail516	-	-	222.02	4,734	5	-	-
rail582	-	-	258.71	3,231	5	-	-
rat1	7.70	1,613	7.05	1,589	10	44.71	2,901
rat5	23.80	2,015	21.83	1,928	12	35.51	3,107
rat7a	81.57	2,866	66.17	2,475	15	127.08	4,221
recipe	0.02	32	0.02	23	6	0.02	48
rosen2	3.14	990	2.21	734	1	17.61	4,161
rosen7	0.21	195	0.11	159	3	0.30	517
rosen8	1.09	484	0.45	338	2	1.13	988
rosen10	14.91	1,828	7.04	1,329	4	36.15	4,777
sc105	0.03	76	0.03	66	5	0.02	68
sc205	0.17	180	0.10	159	2	0.09	166
sc205-2r-400	5.37	924	0.28	47	1	0.17	51
sc205-2r-800	39.88	1,685	1.48	87	1	0.98	91
sc205-2r-1600	314.74	3,364	11.44	167	1	9.81	171
sc50a	0.02	39	0.01	32	5	0.02	27

sc50b	0.01	32	0.01	31	5	0.01	29
scagr25	0.3	215	0.17	149	12	0.79	462
scagr7	0.04	82	0.03	76	2	0.03	82
scagr7-2b-64	126.85	2,617	21.56	2,937	3	36.23	4,278
scagr7-2r-216	13.32	2,616	13.43	2,503	3	24.95	4,653
scagr7-2r-432	98.28	5,238	94.37	5,248	3	188.36	9,629
scfxm1	0.36	344	0.32	248	10	0.21	349
scfxm1-2b-64	828.32	4,820	625.38	3,878	1	-	-
scfxm3	4.57	1,310	2.83	829	9	3.36	1,090
scrs8	0.56	414	0.63	450	3	0.55	583
sctap1	0.12	282	0.12	241	1	0.22	387
sctap2	0.36	362	0.65	429	1	3.11	1,042
sctap3	0.95	623	1.48	636	1	5.56	1,155
share1b	0.11	148	0.09	112	1	0.07	155
share2b	0.05	97	0.04	84	5	0.05	136
share12l	0.22	176	0.32	303	2	0.45	204
share12s	0.15	326	0.16	335	1	0.16	311
slptsk	143.47	1,301	99.37	1,193	5	-	-
standata	0.26	362	0.18	228	1	0.13	216
stocfor1	0.01	17	0.01	22	2	0.02	30
stocfor2	4.29	784	5.3	914	2	9.9	1,214
stocfor3	330.85	4,955	259.15	2,984	2	584.23	10,142
testbig	43.07	804	2.21	803	1	0.79	804
zed	0.01	29	0.01	25	3	0.02	50
Average	2.94 (1.49)	791.58 (585.85)	1.92 (1.01)	583.10 (438.67)	3 (3)	4.04 (1.74)	984.16 (690.60)

6.5 Conclusions

Some combinations of LP algorithms have been already proposed in the literature. In this paper, we study the combination of an IPM and an EPSA algorithm. More specifically, we used Mehrotra's Predictor-Corrector method and PDIPSA, a primal-dual interior point simplex algorithm. Our hybrid approach starts running Mehrotra's Predictor-Corrector method for a number of iterations in order to calculate a "good" interior point. Then, it initializes PDIPSA with a dual feasible basic partition and the interior point. Finally, PDIPSA continues solving the problem. Our aim is to take full advantage of both LP algorithms; use Mehrotra's Predictor-Corrector method at the first iterations

which lead to significant enhancement of the objective function's value and then, use PDIPSA at the latter iterations which lead to fast convergence to an optimal solution. PDIPSA was utilized because of its behavior to the problem of stalling and cycling which enhances its computational performance and makes it one of the most efficient variations of EPSA.

A computational study was also presented with experiments over the Netlib (optimal and Kennington) and the Mészáros collection. Computational results showed that the proposed hybrid algorithm can improve PDIPSA's execution time significantly. More specifically, the proposed hybrid algorithm is 1.53× faster than PDIPSA and it performs 1.36×% less iterations than PDIPSA. In addition, the proposed hybrid algorithm is on average 2.1× faster than the primal revised simplex algorithm.

Chapter 7

Conclusions

This chapter presents the most significant conclusions of the current thesis. A thesis that focuses on Linear Programming which shows the differences between the three main categories of linear programming algorithms, simplex type algorithms, interior point methods and exterior point simplex algorithms.

7.1 Results

At the beginning, we include a short presentation of some general discussion about linear programming, operations research and the history of LP. Moreover, notations and the main parts of a LP is presented in order to help readers with no experience in linear programming, illustrative examples are also included for the same purpose. Another significant section that is of great value, is the duality and its unique characteristics which triggered the explosion of a new group of algorithms, the primal dual algorithms. In addition, we present some very useful information about the geometry of LP.

Furthermore, we have presented three different groups of linear programming algorithms. The primal revised simplex algorithm and an exterior point simplex algorithm, both of them belong to simplex-type algorithms. The third category includes an interior point method based on Mehrotra's Predictor-Corrector method. As its name shows, an IPM moves through the interior of the feasible region towards the optimal solution. Consequently, this is the main difference with the simplex algorithm, which follows a sequence of adjacent boundary points to the optimal solution. In contrast to IPM and Simplex algorithm, EPSA uses basic points which are not feasible. Consequently, it moves exterior to feasible region, something which can lead to significant reduction of algorithm's iterations.

A significant section in linear programming is the preprocessing techniques which are of great value especially in large scale LPs. The main goal of preprocessing is to construct a new equivalent LP with significant reduced dimensions which can lead to enhancements of the software's computational performance. Preprocessing can improve the computational performance of a LP solver with many ways: (i) Reduction of the dimensions of LP, (ii) Enhancement of some arithmetic and computational characteristics of LP, (iii) Detection if a LP is infeasible or unbounded and (iv) Detection of specific characteristics and forms of LP that are detected only when presolve methods are used. For all of these reasons, we include a complete presolve routine consisting of three different scaling techniques and eight presolve methods. Furthermore, a new presolve technique is introduced with significant computational results from the analysis of primal feasibility conditions. Computational results with a set of optimal benchmark problems from the NETLIB set are also presented and they clearly show that the presolve techniques are highly successful in reducing the size of a LP. Moreover, the computational results for the new presolve technique presented in this thesis are quite satisfactory. The application of the proposed presolve technique reduces significantly the problems' dimensions and at the same time decreases the total solver's solution time. On average, the new presolve technique has achieved 26.64% reduction in the number of constraints, 48.03% reduction in the number of variables and 30.73% reduction in the number of non-zeros. Furthermore, the new presolve technique has achieved 5.47% reduction in the number of iterations and 5.99% reduction in the execution time.

As it is mentioned in the introduction of the current thesis, we investigate the combination of linear programming algorithms from different groups and more specifically we combine an IPM with EPSA, as a result we have presented a computational comparison of PDEPSA, a variation of EPSAs and we conclude to PDIPSA which showed a very promising computational behaviour. PDIPSA attempts to avoid the problem of stalling and cycling which can lead to a more effective algorithm with better computational performance. The computational results showed that PDIPSA is superior to EPSA and the simplex algorithm in all densities and dimensions. Apart from that, the performance of PDIPSA is getting better comparing to the simplex algorithm while the size of the LPs increases. Moreover, the computational performance of PDIPSA is much better in very sparse problems. The difference between simplex algorithm and PDIPSA is greater in sparser problems. This is a strong and significant advantage of PDIPSA because in real problems the level of density is very low.

We have also studied the combination of an IPM and a variation of EPSAs. We used Mehrotra's Predictor-Corrector method due to its significant computational performance and PDIPSA which showed its superiority in Chapter 6 and it is a primal-dual exterior

point simplex algorithm. Our hybrid approach starts running Mehrotra's Predictor-Corrector method for a number of iterations in order to calculate a "good" interior point y for the initialization of PDIPSA which continues solving the problem. Our aim is to take full advantage of both LP algorithms; use Mehrotra's Predictor-Corrector method at the first iterations which lead to significant enhancement of the objective function's value and then, we use PDIPSA at the latter iterations which lead to fast convergence to the optimal solution. Finally, the results from the computational survey with experiments over the Netlib (optimal and Kennington) and the Mészáros collection showed that the proposed hybrid algorithm can improve PDIPSA's execution time significantly. More specifically, the proposed hybrid algorithm is 1.53× faster than PDIPSA and it performs 1.36×% less iterations. Finally, the proposed hybrid algorithm is on average 2.1× faster than the primal revised simplex algorithm.

7.2 Future Research

A very interesting topic for further research and work is to investigate the possible relation between the dimensions of LP and the number of iteration of the IPM which is used in the first stage of the hybrid algorithm.

Furthermore, in the current thesis, we used Mehrotra's Predictor-Corrector method for the IPM, it would be very useful to investigate if the combination of PDIPSA with other IPM would be more computational effective. For example, a computational comparison between the combination of PDIPSA with each of one algorithm of the three different categories of IPMs: (i) affine-scaling methods, (ii) potential reduction methods, and (iii) central trajectory methods could conclude to significant conclusions.

Moreover, last years there is significant research work on field of the parallel implementation of linear programming algorithms due to the availability of cost-effective parallel computers. Researchers had mainly focused on the attempt to implement parts of the Simplex algorithm like the inverse of matrix A. However, it would extremely useful to compare parallel implementation of hybrid algorithms with its sequential implementation and with parallel Simplex.

Bibliography

[1] Ahuja, R.K., Magnanti, T.L. & Orlin, J.B. (1993). Network flows, algorithms and applications, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[2] Adler, M., Resende, C., Veiga, G. & Karmarkar, N. (1989). An implementation of Karmarkar's algorithm for linear programming. Mathematical Programming, 44, 297–335.

[3] Al-Najjar, C. & Malakooti, B. (2011). Hybrid–LP: Finding advanced starting points for simplex and pivoting LP methods. Computers & Operations Research 38(2), 427–434.

[4] Andersen, E.D. & Ye, Y. (1996). Combining interior–point and pivoting algorithms for linear programming. Management Science 42(12), 1719–1731.

[5] Anderson, E.D. & Anderson, K.D. (1995). Presolving in Linear Programming, Mathematical Programming, 71, 221–245.

[6] Anstreicher, K. & Terlaky, T. (1994). A monotonic build up simplex algorithm for linear programming, Operations Research, 42, 556–561.

[7] Arsham, H., Cimplerman, G., Nadja, D., Talib, D. & Janez, G. (2005). A computer implementation of the push and pull alogrithm and it's computational comparison with lp implex method, Applied Mathematics and Computation, 170, 36–63.

[8] Baricelli, P., Mitra, G. & Nygreen, B. (1998). Modelling of augmented makespan assignment problems (AMAPs): Computational experience of applying integer presolve at the modeling stage, Annals of Operations Research, 82, 269–288.

[9] Benzi, M. (2002). Preconditioning techniques for large linear systems A survey, Journal of Computational Physics, 182, 418–477.

[10] Bertsimas, D. & Tsitsiklis, J. N. (1997). Introduction to linear optimization. Belmont, MA: Athena Scientific, 6, 65–67.

[11] Bixby, R. E. (1992). Implementing the simplex method: The initial basis. ORSA Journal on Computing, 4(3), 267–284.

[12] Bixby, R.E., Gregory, J.W., Lustig, I.J., Marsten, R.E. & Shanno, D.F. (1992). Very Large–scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods. Oper. Res. 10(5), 885–897.

[13] Bixby, R.E. & Saltzman, M.J.: Recovering an Optimal Basis from an Interior Point Solution. (1993). Oper. Res. Letter 15(4), 169–178.

[14] Borgwardt, K.H. (1982a). Some distribution-dependent results about the asymptotic order of the arerage number of pivot steps of the simplex method. Math Oper Res, 7, 441–462.

[15] Borgwardt, K.H. (1982b). The average number of pivot steps required by the simplex method is polynomial. Z Oper Res, 26, 157–177.

[16] Carolan, W.J., Hill, J.E., Kennington, J.L., Niemi, S. & Wichmann, S.J. (1990). An Empirical Evaluation of the KORBX Algorithms for Military Airlift Applications. Operations Research 38(2), 240–248.

[17] Carstens, D. M. (1968). Crashing techniques. W. Orchard-Hays, Advanced Linear-Programming Computing Techniques, McGraw-Hill, New York, 131–139.

[18] Chan, K. (2005). Matrix Preconditioning Techniques and Applications. Cambridge University Press.

[19] Chang, S.F. & McCormick, S.T. (1992). A hierarchical algorithm for making sparce matrices sparser, Mathematical Programming, 56, 1–30.

[20] Chen, D., Pardalos, P. & Saunders M. (1994). The simplex algorithm with a new primal and dul pivot rule, Operations Research Letters, 16, 121–127.

[21] Chvatal, V. (1983). Linear programming. Macmillan.

[22] Cottle, R.W., Johnson, E & Wets, R. (2007) George B. Dantzig (1914–2005). Not AMS 54:344–362 CPLEX ILOG (2007) 11.0 User's manual. ILOG SA, Gentilly, France

[23] Curtis, A. R. & Reid, J. K. (1972). On the automatic scaling of matrices for Gaussian elimination. J. Inst. Math. Appl., 10, 118–124.

[24] Dantzig, G. B. (1947). Maximization of a linear function of variables subject to linear inequalities. T.C. Koopmans: Activity Analysis of Production and Allocation, New York, 1947, 339–347.

[25] Dantzig, G. B. (1949). Programming in linear structure. Econometrica, 17, 73–74.

[26] Dantzig, G. B. & Wood, M. K.(1951). Programming of interdependent activities, i: general discussion. Econometrica, 17, 193–199.

[27] Dantzig, G. B. (1963). Linear programming and extensions. Princeton, NJ: Princeton University Press.

[28] Dikin, I. I. (1967). Iterative solution of problems of linear and quadratic programming, Soviet Mathematics Doklady, 8, 674–675.

[29] Dikin, I. I. (1974). On the convergence of an iterative process, Soviet Mathematics Doklady, 12, 60–64.

[30] de Buchet, J. (1966). Experiments and statistical data on the solving of large–scale linear programs. In Hertz, D.A., Melese, J. (eds.) Proceedings of the Fourth International Conference on Operational Research, Wiley–Interscience, New York, 3–13.

[31] Dolan, E. D. & More, J. J. (2002).Benchmarking optimization software with performance profiles. Mathematical Programming, 91(2), 201–213.

[32] Dongarra, J. & Sullivan, F. (2000). Guest Editors' Introduction: The Top 10 Algorithms. Computing in Science & Engineering, 22–23.

[33] Forrest, J.J.H. & Goldfarb, D. (1992). Steepest edge simplex algorithm for linear programming. Math Program, 57, 341–374

[34] Fourer, R. & Gay, D.M. (1993). Experience with a Primal Presolve Algorithm, AT&T Bell Labooratories, Technical Report.

[35] Frisch, K. R. (1954). Principles of linear programming: With particular refence to the bouble gradient form of the logarithmic potential method, Memorandum, University Institute of Economics, Oslo, Norway.

[36] Frisch, K. R. (1955). The logarithmic potential method of convex programming. Technical report, University Institute of Economics, Oslo, Norway.

[37] Fulkerson, D. R. & Wolfe, P. (1962). An algorithm for scaling matrices. SIAM Rev., 4, 142–146.

[38] Gay, D.M. (1985). Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Newsletter 13, 10–12.

[39] Glavelis, Th., Paparrizos, K. & Samaras, N. "Computational experience with presolve techniques", In Proc. (electronic form) 9th Balkan Conference on Operational Research, 02-06 September, Constanta, Romania, (2009).

[40] Glavelis, Th., Ploskas, N. & Samaras, N. (2018) "Improving a primal–dual simplex-type algorithm using interior point methods", Optimization, Vol. 67 (12), pp. 2259-2274.

[41] Glavelis, T., Ploskas, N. & Samaras, N.: Combining interior and exterior simplex type algorithms. In Proceedings of the 17th Panhellenic Conference on Informatics, pp. 174–179. ACM (2013).

[42] Glavelis, Th. & Samaras, N. (2013). "An experimental investigation of a primal-dual exterior point simplex algorithm", Optimization: A Journal of Mathematical Programming and Operations Research, Vol. 62(8), pp. 1143–1152.

[43] Gondzio, J. (1992). Splitting dense columns of constraint matrix in interior point methods for large-scale linear programming. Optimization, 24, 285–297.

[44] Gondzio, J. (1996). Multiple centrality corrections in a primal-dual method for linear programming. Computational Optimization and Applications, 6, 137–156.

[45] Gondzio, J. (1997). Presolve Analysis of Linear Programs Prior to Applying an Interior Point Method, INFORMS Journal on Computing, 9, 73–91.

[46] Gondzio, J. (2012). Interior point methods 25 years later. European Journal of Operational Research 218(3), 587-601.

[47] Gould, N. I. & Reid, J. K. (1989). New crash procedures for large systems of linear constraints. Mathematical Programming, 45(1-3), 475–501.

[48] Hall, J.A.J. & McKinnon, K.I.M.: PARSMI, a parallel revised simplex algorithm incorporating minor iterations and Devex pricing. In Wasniewski, J., Dongarra, J., Madsen, K., Olesen, D. (eds.) Applied Parallel Computing, Lecture Notes in Computer Science, Springer, 1184, 67–76 (1996).

[49] Hall, J.A.J. & McKinnon, K.I.M. (1998). ASYNPLEX, an asynchronous parallel revised simplex algorithm. Annals of Operations Research 81, 27–49.

[50] Hall, J. & McKinnon, K. (2005). Hyper-Sparcity in the revised simplex method and how to exploit it. Computational Optimization and Applications, 32, 259–283.

[51] Harris, P.M.J. (1973). Pivot selection methods of the Devex LP code. Math Program, 5, 1–28

[52] Hitchock, F. L., (1941). The distribution of a product from several sources to numerous localities. Journal of Mathematical Physics, 20, 224–230.

[53] Hoffman, A. J., Mannos, M., Sokolowsky, D., & Wiegman, N. (1953). Computational experience in solving linear programs. Journal of the Society for Industrial and Applied Mathematics, 1, 17–33.

[54] Huard, P. (1967). Resolution of mathematical programming with nonlinear constraints by the method of centers, In Abadie, J. Nonlinear Programming, 207–219.

[55] Huard, P. (1970). A method of centers by upper-bounding functions with applications, In Rose, Nonlinear Programming: Proceedings of a symposium held at the University of Wisconsin, adison, USA, 1–30.

[56] Ioslovich, I. (2004). Robust reduction of a class of large-scale linear programs, SIAM Journal on Optimization, 12, 262–282.

[57] Iri, M., (1960). A new method of solving transportation-network problems. Journal of the Operations Research Society of Japan 3(1), 2.

[58] Kantorovich, L.V. (1960). Mathematical methods in the organization and planning of production. Manag Sci 6, 550–559. Original Russian version appeared in 1939.

[59] Kardani, O., Lyamin, A.V., & Krabbenhøft, K. (2013). Preconditioned Conjugate Gradient for Large Sparse Systems Arising from Optimization Problems in Geomechanics," in Proceedings of World Congress on Engineering, 216–221.

[60] Karmarkar, N.K. (1984). A new polynomial time algorithm for linear programming. Combinatorica 4, 373–395.

[61] Khachiyan, L. G. (1965). The intrinsic computational difficulty of functions. in Logic Methodology and Philosophy of science, Bar, Hillel, North-Holland, Amsterdam, The Netherlands, 24–30.

[62] Khachiyan, L. G. (1965). Paths, trees and flowers. Canadian Journal of Mathematics, 17, 449–467.

[63] Khachiyan, L. G. (1979). A polynomial algorithm in linear programming. Soviet Mathematics Doklady, 20, 191–194.

[64] Klee, V. & Minty G.J. (1972). How good is the simplex algorithm? In: Shisha O Inequalities–III, Academic, New York, 159–175.

[65] Kortanek, K.O. & Zhu, J. (1988). New Purification Algorithms for Linear Programming. Naval Res. Logistic Quarterly 35, 571–583.

[66] Kuhn, H.W. (1955). The Hungarian method for the assignment problem. Naval research logistics quarterly 2(1-2), 83-97.

[67] Lalami, M.E., El-Baz, D. & Boyer, V.: Multi gpu implementation of the simplex algorithm. In Proceedings of the 2011 IEEE 13th International Conference on High Performance Computing and Communications (HPCC), Banff, Canada, 179–186 (2011)

[68] Larsson, T. (1993). On scaling linear programs – Some experimental results. Optimization, 27, 335–373.

[69] Lemke, E. C. (1954). The dual method of solving the linear programming problem. Naval Research Logistics Quarterly, 1, 36–47.

[70] Lustig, I. (1990). Feasibility issues in a primal-dual interior point algorithm for linear programming. Mathematical Programming, 49, 154–162.

[71] Van Loan, N. J. (2000). Introduction to scientific computing. Prentice-Hall, Inc,. Upper Saddle River, N. J.

[72] Lustig, J. (1989). An analysis of an available set of linear programming test problems, Computers and Operations Research, 16(2), 173–184.

[73] Lustig, I. J., Marsten, R. E., & Shanno, D. F. (1992). On implementing Mehrotra's predictor corrector interior point method for linear programming. SIAM J. Optimization, 2, 435–449.

[74] Mahajan, A. (2011) Presolving mixed integer linear programs. Wiley Encyclopedia of Operations Research and Management Science, John Wiley & Sons, Inc., 4141–4149.

[75] Malakooti, B., & Najjar, A.C. (2010). The complex interior boundary method for linear and nonlinear programming with linear constraints, Applied Mathematics and Computation, 216, 1903–1917.

[76] Maros, I. (2003). Computational Techniques of the Simplex Method, International Series in Operations Research & Management Science, 61, Kluwer Academic Publishers, Boston, MA.

[77] McCormick, S.T. (1990). Making sparse matrices sparser: Computational results, Mathematical Programming, 49, 91–111.

[78] Megiddo, N.: On Finding Primal– and Dual–optimal Bases. ORSA J. Computation 3(1), 63–65 (1991)

[79] Mehrotra, S. (1992). On the implementation of a primal-dual interior point method. SIAM J. Optimization, 2, 575–601.

[80] Meyer, X., Albuquerque, P. & Chopard, B.: A multi–GPU implementation and performance model for the standard simplex method. In Proceedings of the 1st International Symposium and 10th Balkan Conference on Operational Research, Thessaloniki, Greece, 312–319 (2011)

[81] Mészáros, C.: Linear programming test problems. http://www.sztaki.hu/~meszaros/public_ftp/lptestset/misc/ (2016). Accessed 5 May 2016.

[82] Mészáros, C. & Suhl, U. H. (2003). Advanced preprocessing techniques for linear and quadratic programming. OR Spectrum, 25(4), 575–595.

[83] Najjar, A.C. & Malakooti, B. (2011). Hybrid-lp: Finding advanced starting points for simplex and pivoting lp methods, Computers & Operations Research, 38, 427–434.

[84] Von Neumann, J. (1953). A certain zero-sum two-person game equivalent to the optimal assignment problem, Contributions to the Theory of Games, II, Kuhn, W.H., Tucker, W.A., Annals of Mathematics Studies, 28, 5–12.

[85] Orchard-Hays, W. (1954). Background development and extensions of the revised simplex method. Report RM 1433, The Rand Corporation, Santa Monica.

[86] Ordóñez, F. & Freund, R. (2003). Computational experience and the explanatory value of condition measures for linear optimization. SIAM J. Optimization 14(2), 307–333.

[87] Pan, P.Q. (1990). Practical finite pivoting rules for the simplex method. OR Spektrum, 12, 219–225.

[88] Pan, P.Q. (2008). Efficient nested pricing in the simplex algorithm. Operations Research Letters, 36, 309–313.

[89] Pan, P.Q. (2008). largest-distance pivot rule for the simplex algorithm. European Journal of Operational Research, 187, 393–402.

[90] Pan, P.Q. (2013). An affine-scaling pivot algorithm for linear programming. Optimization 62(4), 431–445.

[91] Paparrizos, K. (1991). An infeasible exterior point simplex algorithm for assignment problems. Mathematical Programming 51(1–3), 45–54.

[92] Paparrizos, K. (1993). An exterior point simplex algorithm for (general) linear programming problems. Annals of Operations Research 47, 497–508.

[93] Paparrizos, K., Samaras, N. & Stephanides, G. (2003). A new efficient primal dual simplex algorithm. Computers & Operations Research, 30(9), 1383-1399.

[94] Paparrizos, K., Samaras, N. & Tsiplidis, K. (2009). Pivoting algorithms for (LP) generating two paths. Encyclopedia of optimization, 2nd edition, 2965–2969.

[95] Ploskas, N., Samaras, N. & Sifaleras, A. (2009). A Parallel Implementation of an Exterior Point Algorithm for Linear Programming Problems. In Proceedings of the 9th Balcan Conference on Operational Research (BALCOR 2009), 2-6 September, Constanta, Romania.

[96] Ploskas, N., Samaras, N. & Margaritis, K.: A Parallel Implementation of the Revised Simplex Algorithm Using OpenMP: Some Preliminary Results. In Optimization Theory, Decision Making, and Operations Research Applications, 163–175. Springer New York (2013)

[97] Ploskas, N. & Samaras, N. (2013). A Computational Comparison of Basis Updating Schemes for the Simplex Algorithm on a CPU–GPU System. American Journal of Operations Research 3, 497–505.

[98] Ploskas, N. & Samaras, N. (2013). The Impact of Scaling on Simplex Type Algorithms. In Proceedings of the 6th Balkan Conference in Informatics, ACM, 19–21 September, Thessaloniki, Greece, 17–22.

[99] Ploskas, N. & Samaras, N. (2015). A computational comparison of scaling techniques for linear optimization problems on a graphical processing unit. International Journal of Computer Mathematics, 92(2), 319–336.

[100] Ploskas, N. & Samaras, N. (2014). GPU accelerated pivoting rules for the simplex algorithm. Journal of Systems and Software 96, 1–9

[101] Ploskas, N. & Samaras, N. (2015). Efficient GPU-based implementations of simplex type algorithms. Applied Mathematics and Computation, 250, 552–570.

[102] Ploskas, N. & Samaras, N. (2017). Linear Programmins Using Matlab. Springer Optimization and Its Applications.

[103] Reid, J.K. (1971). On the method of conjugate gradients for the solution of large sparse systems of linear equations, Proceedings of the conference on Large Sparse Sets of Linear Equations, 231–254.

[104] Rose, J.D. & Willoughby, A.R. (1972). Sparse matrices and their applications, Plenum Press.

[105] Samaras, N (2001). Computational improvements and efficient implementation of two path pivoting algorithms, PhD Dissertation, University of Macedonia.

[106] Smale, S. (1983a). On the average number of steps of the simplex method of linear programming. Math Program 27, 241–262.

[107] Smale, S. (1983b). The problem of the average speed of the simplex method. In: Bachem A, Grotschel M, Korte B, (eds) Mathematical programming, the state of the art. Springer, Berlin, 530–539.

[108] Stigler, G. J. (1945). The cost of subsistence. Journal of Farm Economics, 27, 303–314.

[109] Swietanowski, A. (1995). A modular presolve procedure for large scale linear programming. International Institute for Applied Systems Analysis. url: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.8206

[110] Terlaky, T. (1985). A convergent criss-cross method. Optimization 16(5), 683–690.

[111] Terlaky, T. & Zhang S. (1993). Pivot rules for linear programming: A survey on recent theoretical developments, Annals of Operations Research, 46, 203–233.

[112] Terlaky, T. (Ed.): Interior point methods of mathematical programming, 5, Springer Science & Business Media (2013)

[113] Thomadakis, M.E., Liu, J.C.: An efficient steepest-edge simplex algorithm for SIMD computers. In Proceedings of the 10th International Conference on Supercomputing (ICS 1996), Philadelphia, Pennsylvania, USA, 286–293 (1996)

[114] Tomlin, J. A. (1975). On scaling linear programming problems. Math. Program. Stud., 4, 146–166.

[115] Triantafyllidis, C.P.: A Non–Monotonic Infeasible Interior–Exterior Point Algorithm for Linear Programming. PhD thesis, University of Macedonia, Greece (2014)

[116] Urdaneta, J., Perez, L.G., Gomez, J.F., Feijoo, B. & Gonzalez, M. (2000). Presolve analysis and interior point solutions of the linear programming coordination problem of directional overcurrent relays, Electrical Power and Energy Systems, 23, 819–825.

[117] Von Neumann, J. (1947). On a maximization problem. Technical report, Institute for Advanced Study, Princeton, NJ, USA.

[118] Ye, Y. (1989). Eliminating Columns in the simplex Method for linear programming, Journal of Optimization Theory and Applications, 63(1), 69–77.

[119] Weispfenning, V. (2004). Solving constraints by elimination methods, Automated Reasoning, 3097, 336–341.

[120] Wright, S.J.: Primal-dual interior-point methods. Siam (1997)

[121] Williams, H. (1992). The elimination of integer variables. The Journal of the Operational Research Society, 43(5), 387–393.

[122] Zionts, S. (1969). The criss-cross method for solving linear programming problems. Management Science 15(7), 426–445.