2017

Duality and a Closer Look at Implementation of Linear Optimization Algorithms

Naga Venkata Chaitanya Gudapati
Lehigh University

LEHIGH UNIVERSITY

Duality and a Closer Look at Implementation of Linear Optimization Algorithms

by

Chaitanya Gudapati

A thesis submitted in partial fulfillment for the degree of
Master of Science, Industrial and Systems Enginnering

in the

Department of Industrial and Systems Engineering

May 2017

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

Date

Tamás Terlaky, Thesis Advisor

Tamás Terlaky, Chairperson of Department

"We are like dwarfs on the shoulders of giants, so that we can see more than they, and things at a greater distance, not by virtue of any sharpness of sight on our part, or any physical distinction, but because we are carried high and raised up by their giant size."

- Bernard De Chartres

LEHIGH UNIVERSITY

Abstract

Department of Industrial and Systems Engineering

Master of Science, Industrial and Systems Enginnering

by Chaitanya Gudapati

Duality played, and continues to play a crucial role in the advancement of solving Linear Optimization (LO) problems. In this thesis, we first review the history of LO and various software to solve LO problems. In the next chapter, we discuss Pivot Algorithms, basis tableaus, primal and dual Simplex methods and their computational implementation. Then we discuss Interior Point Methods (IPM) and the numerical linear algebra involved in their implementation. The next chapter discusses duality in significant detail, and the role of duality in LO software design. We also describe the dualizing scheme used to dualize the NETLIB test problems. We then discuss the computational results on specially constructed problems and the primal and dual NETLIB set using some of the leading LO software packages including CPLEX, GuRoBi and MOSEK.

In this thesis, the first chapter deals with the history of LO and LO software packages. The second chapter talks about basis tableau, pivot algorithms — primal and dual Simplex methods and some computational methodology. Chapter 3 discuses about Interior Point Methods and the numerical linear algebra involved. In Chapter 4, we explore duality, the role of duality LO software development, and the techniques used to dualize the standard LO optimization problems in the NETLIB set. In Chapter 5, we present the computational experiments on specially constructed problems. We also present the experiments on primal and dual NETLIB set. We finally present our conclusions in Chapter 6.

Acknowledgements

I would like to sincerely thank my adviser Prof. Tamás Terlaky for being an incredible mentor for that past two years. His knowledge about optimization is second to none and I can not thank him enough for his patience.

I also want to thank the members of the Inmate Assignment project — Prof. Tamás Terlaky, Prof. Louis Plebani, Prof. George Wilson and Ph.D. student Mohammad Shahabsafa. Though not directly related to my research, the work I had done on the project has been extremely rewarding and furthered my knowledge of optimization.

Contents

List of Tables

Chapter 1

Introduction

1.1 A Brief History of Linear Optimization and its Algorithms

The first Linear Optimization (LO) problem was documented by A. N. Tolstoi [25] in the 1930 article, *Methods of finding the minimal total kilometrage in cargo-transportation planning in space* published by the Commissariat of Transportation of the Soviet Union. In this paper he studied transportation problems and suggested a few approaches to solve the problems, and made quite a few remarks on the optimality of the solution. Another Russian mathematician, Leonid Kantorovich [14] in the later years of 1930s and early 1940s worked on LO. In recognition of his pioneering work, he was awarded the Nobel Prize in Economics along with Koopmans in 1975.

Tolstoi's and Kantorovich's work primarily focused on the Economic theory, and the mathematical and algorithmic foundations of linear programming were developed by George B. Dantzig. Dantzig [7] was also solving the analogous resource allocation problems but his approach was more robust and could be applied to solve various real world problems which was in stark contrast to the economists' work. Dantzig has invented the simplex method to solve LO problems and to this date, simplex method remains one of the most efficient ways to solve LO and Mixed Integer LO (MILO) problems.

The theory of duality was developed by von Neumann [8] after meeting Dantzig. The dual simplex method was first proposed by Lemke [18] but it was not actually used to solve optimization problems like the primal simplex method was used. It was initially used in MILO, and it was used a general purpose solver first in the early 1990s when Forrest and Goldfarb [10] refined their steepest edge pricing method.

The question of whether an LO problem can be solved in polynomial time was answered by Leonid Khachiyan [16] in 1979. The Ellipsoid method uses shrinking ellipsoids around the optimal set (if it is non-empty). The Ellipsoid method is a significant milestone in the theory of LO as it gave the first polynomial upper bound on the number of arithmetic operations that is needed to solve LO problem with rational data.

Kachiyan's work suffered from severe drawbacks when implemented to solve practical problems. In computational practice, the simplex method was still much faster than the Ellipsoid method. In 1984, Narendra Karamakar [15] came up with an algorithm, which not only had improved the polynomial-time bound, but it was much more suitable for implementing in computational practice. Karamrkar's work ignited renewed interest in LO. Thousands of papers on algorithmic variants and extensions were published in the following decades.

1.2 History of Linear Optimization Software

LO algorithms, namely simplex methods were first computationally implemented at the National Bureau of Standards on their Standard Eastern Automatic Computer which could solve small instances of LO problems with 10 constraints and 20 variables [22]. This computer was built using vacuum tubes and solid-state diode logic. Dantzig left the U.S. Air-Force in 1952 and started work at the RAND Corporation to research on LO. At RAND, he met Orchard-Hays and together they have come up with the fundamentals and foundations of the computational methodology of the simplex method.

Bixby [5] detailed the problems Dantzig and Orchard-Hayes faced when implementing the simplex method at RAND. The initial implementation used to calculate the basis inverse at every iteration which is a very expensive computational operation. The next software versions had the product form of the inverse, which enabled larger problems to be solved in reasonable amount of time. The implementations of various simplex methods kept getting better and better. The implementations could handle 512 constraints when run on IBM 700 series computers and by the late 1950s, LO software was commercialized and the oil and gas industry became major users of the software. The LP/90 software by Orchard-Hayes was of important significance because of the efficient implementation of Dantzig's revised simplex method and it could solve much larger problems having 1024 constraints [22].

The late 1960s and early 1970s were important not only for the computational methods of LO but also to the whole computational world. The IBM 360 mainframe computer with memory from 8 to 128 KB went on to become one of the most successful computers in the history. For LO scientists, this computer meant two things — bigger problems can be solved faster and some algorithmic aspects, which could not be implemented on the previous machines due to memory limitations, could be implemented now. The 70s also meant that there were further developments for the simplex method. Some changes to bounds treatment like generalized upper bounded techniques, pre-processing, sparse matrix technology, better LU-factorization and pricing strategies lead to more stable and faster solution times. This was also during this time when dual simplex was first implemented in commercial LO software but it was used only in branch and bound methods of MILO. All these algorithms were written in machine code for specific platforms. APEX, FMPS, LPS and MPS are such software. These software have been further updated in the following years.

The late 70s also saw the optimization software written in portable programming languages. MINOS, which is a non-linear optimization software also had a primal simplex method and XMP which had a good implementation of simplex method were written in FORTRAN. The early 80s had another computer revolution when IBM has introduced the personal computer (PC). These platforms can be used to generate and solve large optimization problems. LINDO, XpressMP and CPLEX were released during this time. This was also the time when simplex method has reached its maturity and the interior-point revolution has started.

Khachiyan's ellipsoid method of 1979 was never popular for solving large scale linear optimization problems and then in 1984, came Karmakar's interior point methods (IPMs). As a first such action, surprising many academics, AT&T patented the algorithm and decided to sell the KORBX system which was based on the Karmakar's Algorithm. Lustig, Marsten, and Shanno [19] implemented a primal-dual log barrier interior point method in OB1 which trounced KORBX. During this time, IBM updated their optimization code and released OSL. In the early 1990s CPLEX also introduced a barrier solver in their optimization package. During this time, the improvements included the implementation of dual simplex method as a general purpose solver, Cholesky Factorization for IPMs, improved sparse linear algebra.

Because of their parallel nature in the core of the linear algebra part, barrier methods can be heavily parallelized and hence with more availability of processors, barrier methods have become faster especially for very large, very sparse problems. The barrier method converges to an exact optimal solution. However, in case of degeneracy, it does not give an optimal basis solution. So all the IPM implementations also have primal and dual simplex methods for the identification of the optimal basis.

Unlike IPMs, the simplex method can not be parallelized. Even though there is continuous improvement of software and every increasing processing power, there has not been any revolutionary change in the LO algorithms since mid 2000s. Various new software companies have come up with software which contain efficient implementation of simplex and IPMs. MOSEK which was first introduced in 1999 and GuRoBi which was introduced in 2009. MOSEK's primary focus was on IPMs but they also include very efficient implementations of simplex methods. GuRoBi started with pivot algorithms but quickly included the IPMs into their software suite.

Chapter 2

Pivot Algorithms

Pivot algorithms encompass many variants of the simplex method and the criss-cross algorithm which was independently developed by Chang [6], Terlaky [23] and Wang [26]. Most of the well known pivot algorithms, as documented by Terlaky and Zhang [24] are variants of the simplex method and they ensure that the feasibility of the basis is preserved. Unlike simplex method variants, the criss-cross method can start with any basic solution and there is no guarantee that the basis will be feasible till optimality, or the infeasibility is detected. Computationally, the simplex method is much more efficient than the criss-cross method. In this chapter we will look at variants of the simplex method — both the primal and dual simplex method, and we will demonstrate that the dual simplex method is exactly the same as the primal simplex method applied to the dual problem.

Dantzig [7] discovered the simplex method in 1947 while working at the United States Air Force. Within short time the simplex method was realized to be a powerful algorithm which can solve a lot of real-life LO problems. Klee & Minty [17] proved that in the worst case scenario, the simplex method is exponential. However, for average case practical problems, simplex method is highly efficient in computational practice.

2.1 Terminology

Any optimization problem, through algebraic transformations, can be converted to the following standard form (2.1), which we will call the primal form. The goal of the LO optimization problem is to find the minimum of a linear function of n variables subject to m linear constraints, while all the linear variables are non-negative.

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0, \end{aligned} \tag{2.1}$$

where $A \in \mathbb{R}^{m \times n}$, $c, x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. Throughout this thesis, it is assumed that the rows of A are linearly independent, i.e. , $\text{rank}(A) = m$. The dual of this problem is given as

$$\begin{aligned} \text{maximize} \quad & b^T y \\ \text{subject to} \quad & A^T y \leq c, \\ & y \geq 0, \end{aligned} \tag{2.2}$$

where $y \in \mathbb{R}^m$. By introducing slack variables, the dual problem can be equivalently written as,

$$\begin{aligned} \text{maximize} \quad & b^T y \\ \text{subject to} \quad & A^T y + s = c, \\ & s \geq 0, \end{aligned} \tag{2.3}$$

where $s \in \mathbb{R}^n$.

There are other forms like the **symmetric standard form**, which is given as:

$$\begin{aligned} \text{(Primal)}\ \min \quad & c^T x \\ & Ax \geq b, \\ & x \geq 0, \end{aligned} \qquad\qquad \begin{aligned} \text{(Dual)}\max \quad & b^T y \\ & A^T y \leq c, \\ & y \geq 0. \end{aligned}$$

By introducing slack variables to both the primal and dual problems, they can be transformed as

$$\begin{aligned} \min \quad & c^T x \\ & Ax - z = b, \\ & x, z \geq 0, \end{aligned} \qquad\qquad \begin{aligned} \max \quad & b^T y \\ & A^T y - s = c, \\ & y, s \geq 0. \end{aligned}$$

2.1.1 Feasible, Basic and Optimal Solutions

Any x which satisfies the equality constraints, $Ax = b$ and the non-negativity constraint, $x \geq 0$ is called a feasible solution. Hence, the feasible solution set can be written mathematically as,

$$\mathcal{F} = \{x : Ax = b, x \geq 0\}.$$

If $\mathcal{F}$ is empty, then we have an infeasible problem. It should also be noted that whenever we have a non-empty $\mathcal{F}$, then it is a convex set.

It is easy to observe that each column in the matrix A corresponds to a variable x. If we take m linearly independent columns of A, then the resulting $m \times m$ matrix forms a **basis**, and the variables which are associated with these m selected columns are called **basic variables**.

Let us divide the set of all variables I, into basic and non-basic variables. Let the basic index set be represented by $I_\mathcal{B}$ and non-basic index set be represented by $I_\mathcal{N}$. Using the same logic, we can rearrange and partition the matrix A , into two parts — $A_\mathcal{B}$ contains the columns appearing in the index set $\mathcal{B}$. Similarly, $A_\mathcal{N}$ contains the columns appearing in the index set $I\mathcal{N}$, i.e.

$$A = [A_\mathcal{B}|A_\mathcal{N}].$$

We can also partition the variables into $x_\mathcal{B}$ and $x_\mathcal{N}$ using the same logic.

The constraint equation set can be presented as,

$$A_\mathcal{B}x_\mathcal{B} + A_\mathcal{N}x_\mathcal{N} = b.$$

The basis matrix $A_\mathcal{B}$ is non-singular, so we can multiply both sides of the above equation by $A_\mathcal{B}^{-1}$ and after some rearrangement, we get

$$x_\mathcal{B} = A_\mathcal{B}^{-1}(b - A_\mathcal{N}x_\mathcal{N}).$$

If all the non-basic variables are set to 0, then we have

$$x_\mathcal{B} = A_\mathcal{B}^{-1}b.$$

The above solution with $x_{\mathcal{N}} = 0$, is called a **basic solution**, if additionally $x_{\mathcal{B}} \geq 0$ holds, then it is called **basic feasible solution** (BFS).

We say that a given BFS $x_{\mathcal{B}}$ is an optimal solution, when the value of $c_{\mathcal{B}}^T x_{\mathcal{B}}$ is the lowest for all $x \in \mathcal{F}$. It is known that if an optimal solution exists, then an optimal basic feasible solution exists too **??**.

2.1.2 Pivoting

Pivoting is the fundamental transformation in all simplex methods. A linear transformation is called pivot when one variable leaves the basis and another variable enters the basis. The following text expands on the idea of pivoting on the element in the k^{th} row and ℓ^{th} column in the basis tableau.

Let $I_{\mathcal{B}}$ be the index set of the basic variables before pivoting and $I_{\mathcal{B}}^{'}$ be the new index set of basic variables after a pivot. A pivot where k leaves and variable ℓ enters the basis. The changes in the basis and the basis tableau are as follows

$$I_{\mathcal{B}}^{'} = I_{\mathcal{B}} \cup \{\ell\} \setminus \{k\},$$

$$\tau_{ij}^{'} = \tau_{ij} - \frac{\tau_{i\ell}\tau_{kj}}{\tau_{k\ell}}, \ \forall i \in I_{\mathcal{B}}^{'} \setminus \{\ell\}; \ j \in I_{\mathcal{N}}^{'} \setminus \{k\},$$

$$\tau_{ik}^{'} = -\frac{\tau_{i\ell}}{\tau_{k\ell}}, \ \forall i \in I_{\mathcal{B}}^{'} \setminus \{\ell\},$$

$$\tau_{\ell j}^{'} = \frac{\tau_{kj}}{\tau_{k\ell}}, \ \forall j \in I_{\mathcal{N}} \setminus \{\ell\},$$

$$\tau_{\ell k}^{'} = \frac{1}{\tau_{k\ell}}.$$

TABLE 2.1: Pivot operations and changes to the basis tableau.

$$\begin{array}{c|c|cc|} & & j & l \\ \hline & & & \\ \hline i & & \tau_{ij} & \tau_{il} \\ & & & \\ k & & \tau_{kj} & \tau_{kl} \\ \hline \end{array} \quad \begin{array}{c} \text{pivot} \\ \Rightarrow \\ (k,l) \end{array} \quad \begin{array}{c|c|cc|} & & j & k \\ \hline & & & \\ \hline i & & \tau'_{ij} & \tau'_{ik} \\ & & & \\ l & & \tau'_{\ell j} & \tau'_{lk} \\ \hline \end{array}$$

The following tables shows us when one can do primal pivots.

TABLE 2.2: Primal feasible tableau, primal pivot.

	+
⊕	
⊕	+
⊕	

2.2 Primal Simplex Method

The idea of the simplex method builds on the fact that if there exists an optimal solution, then there exists an optimal basic feasible solution too. So if we find the best basic feasible solution (BFS), we get an optimal solution. The primal simplex method will move from one BFS to the next adjacent BFS until it reaches an optimal solution, while the objective function value monotonically decreases or remains the same in case of degenerate problems.

We have the value of $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}(b - A_{\mathcal{N}}x_{\mathcal{N}})$. Now the objective function can be split into $c_{\mathcal{B}}^T x_{\mathcal{B}} + c_{\mathcal{N}}^T x_{\mathcal{N}}$ and substituting the value of $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}b$ into the objective function and rearranging the items gives us

$$c^T x = c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} b + (c_{\mathcal{N}}^T - c_{\mathcal{B}}^T \mathcal{B}^{-1} A_{\mathcal{N}}) x_{\mathcal{N}}.$$

This brings us to a new concept of *reduced cost* of the non-basic variables, which is given by

$$s_j = c_j - c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} a_j \ \forall\ j \in I_{\mathcal{N}}, \tag{2.4}$$

where a_j corresponds to the column of j^{th} non-basic variable in A. It can be clearly seen by simple substitution that the reduced cost of the basic variables is 0.

A sufficient condition of optimality of primal basic feasible solutions is given by

$$s_j \geq 0 \ \forall \ j \in \mathcal{I}_{\mathcal{N}}. \tag{2.5}$$

The simplex method tries to find a BFS which satisfies (2.5) and thus delivering an optimal solution.

If a given feasible basis fails to satisfy condition (2.5), then there is a basis which can be found with a better objective value. The algorithm moves from basis to basis while maintaining the feasibility of all the basic variables and improve the objective function value, or at least not make it worse.

To ensure that feasibility of the basis, the so called ratio test is applied in selecting the pivot element. We do not spend lot of time on the theoretical details of the dual simplex method and the ratio test as it can be found in many books,see Bertsimas [3]

The pseudo-code described in Algorithm 1, which can be found in most of the textbooks, details the mechanism of the primal simplex method.

Algorithm 1 Primal Simplex Method

```
1: Initialization:
2: Let $A_{\mathcal{B}}$ be a primal feasible basis, i.e. $x_{\mathcal{B}} \geq 0$;
3: $I_{\mathcal{B}}$ resp. $I_{\mathcal{N}}$ is the index set of the basis and non-basis variables;
4: while true do
5:     if $s_{\mathcal{N}} \geq 0$ then
6:         stop: the current solution solves the LO problem
7:     else
8:         let $q \in I_{\mathcal{N}}$ be an index with $s_q < 0$;
9:         if the $q$-column of the tableau is non-positive then
10:            stop: (LO) is inconsistent;
11:        else
12:            let $\vartheta := \min\{\frac{x_i}{\tau_{iq}} : i \in I_{\mathcal{B}} \text{ and } \tau_{iq} > 0\}$;
13:            let $p \in I_{\mathcal{B}}$ be such that $\frac{x_p}{\tau_{pq}} = \vartheta$; (ratio test)
14:        end if
15:    end if
16:    perform a pivot: $I_{\mathcal{B}} := I_{\mathcal{B}} \cup q \setminus p$;
17: end while
```

The simplex tableau for the primal problem is given as follows

	0	$-s_{\mathcal{N}}^T$
$x_{\mathcal{B}}$	I	$A_{\mathcal{B}}^{-1}A_{\mathcal{N}}$

2.3 Dual Simplex Method

If we have a primal problem as given in the form (2.1), the dual problem can be written as follows:

$$\begin{aligned} &\text{maximize} \quad b^T y \\ &\text{subject to} \quad A^T y \leq c, \end{aligned} \tag{2.6}$$

which is equivalent to

$$\begin{aligned} &\text{maximize} \quad b^T y \\ &\text{subject to} \quad A^T y + s = c, \\ &\qquad\qquad\qquad\quad s \geq 0. \end{aligned} \tag{2.7}$$

After rearranging the equality constraint in (2.3) and partitioning it, we get

$$s_{\mathcal{B}}^T = c_{\mathcal{B}}^T - y^T A_{\mathcal{B}}, \tag{2.8}$$

$$s_{\mathcal{N}}^T = c_{\mathcal{N}}^T - y^T A_{\mathcal{N}}. \tag{2.9}$$

To make $s_{\mathcal{B}} = 0$ i.e. to construct a complementary primal-dual solution, we need to choose $y^T = c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1}$, which further gives.

$$s_{\mathcal{N}}^T = c_{\mathcal{N}}^T - c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1} A_{\mathcal{N}}. \tag{2.10}$$

It can be observed that $s_{\mathcal{N}}$ in (2.10) is the same as the s in (2.4).

The idea of the dual simplex method is that if the current basis is dual feasible , the analogous to the primal simplex method, we would like to preserve the dual feasibility. If we have $A_{\mathcal{B}}$ as dual feasible and not primal feasible, then we **do not** have optimality.

Now, there exists a neighboring dual feasible basis with better (or same) objective value. There are lot of rules which can be used to shift from one dual BFS to an adjacent dual BFS and one of them is described in Algorithm 2 2, which can also be found in most text books.

Algorithm 2 Dual simplex method

```
1:  Initialization:
2:  let $A_{\mathcal{B}}$ be a primal feasible basis, i.e. $s_{\mathcal{N}} \geq 0$;
3:  $I_{\mathcal{B}}$ resp. $I_{\mathcal{N}}$ is the index set of the basis and non-basis variables;
4:  while true do
5:      if $x_{\mathcal{B}} \geq 0$ then
6:          stop: the current solution solves the LO problem
7:      else
8:          let $p \in I_{\mathcal{B}}$ be an index with $x_p < 0$;
9:          if the $p$-row of the tableau is non-negative then
10:             stop: (LO) is inconsistent;
11:         else
12:             let $\vartheta := \min\{\frac{s_j}{-\tau_{pj}} : i \in I_{\mathcal{N}} \text{ and } \tau_{pj} < 0\}$;
13:             let $q \in I_{\mathcal{N}}$ be such that $\frac{s_q}{-\tau_{pq}} = \vartheta$; (ratio test)
14:         end if
15:     end if
16:     perform a pivot: $I_{\mathcal{B}} := I_{\mathcal{B}} \cup q \setminus p$;
17: end while
```

The acceptable dual simplex pivots are given below:

TABLE 2.3: Dual feasible tableau, dual pivot.

	⊖	⊖	⊖
⊕			
−		−	
⊕			

The simplex tableau for the dual problem is given as follows:

	$0 \ldots 0$	$-x_{\mathcal{B}}^T$	$0 \ldots 0$
y	I	$A_{\mathcal{B}}^{-T}$	0
z	0	$-A_{\mathcal{N}}^T A_{\mathcal{B}}^{-T}$	I

2.4 Primal and Dual Simplex Equivalence

It can be seen from the **simplex tableau for the primal problem** and **simplex tableau for the dual problem** that the dual simplex method is just the primal simplex method applied on the dual problem. So in theory, we can see that primal simplex and dual simplex methods are exactly the same. However, in the subsequent chapters, we can see that computational practice does not fully match the theory.

In the next chapter, we discuss the theory and computational aspects of IPMs.

Chapter 3

Interior Point Methods

Till the discovery of polynomial time Interior Point Methods (IPMs) by N. Karmakar [15], simplex method was considered to be the most suitable method to solve large scale LO problems. Karmakar's method proved that large scale LO problems can be solved in polynomial time. Soon after Karmakar made his discovery, it was shown by Gill, Murray, Tomlin and Wright [12] that his interior point algorithm was very similar to the log-barrier method which was explained by Fiacco and McCormick [9]. In the decades after Karmakar's seminal paper lot of research was done to explore the commonality of linear and non-linear interior point methods.

Efficient implementation of IPMs involves using a stable and robust linear algebra system. In this chapter, we discuss the theory behind IPMs and the most important linear algebra factors that substantially affect a software's performance.

3.1 A Brief Overview of IPMs

Though polynomial IPMs were discovered by Karamakar in 1984, none of the modern software implement his algorithm. AT&T's KORBX was probably the only commercial software system which implemented Karmakar's algorithm and it was immediately superseded by Lustig, Marsten and Shanno's OB1 package. Most of the current software implementations use both infeasible primal-dual interior point algorithm (for brevity we will call this infeasible primal-dual algorithm) and feasible primal-dual interior point algorithm.

The Infeasible primal-dual algorithms can be used to solve large-scale LO problems efficiently. Though they are very attractive theoretically and computationally, they have some shortcomings [2], like choosing a good starting point and infeasibility detection. To overcome these shortcomings, a homogeneous self-dual model was proposed [27].

Let us consider the LO problem and its dual in the standard form at discussed at (2.1)

$$\begin{aligned} &\text{minimize} \quad c^T x && \qquad \text{maximize} \quad b^T y \\ &\text{subject to} \quad Ax = b, && \qquad \text{subject to} \quad A^T y + s = c, \\ &\qquad\qquad\quad x \geq 0, && \qquad\qquad\qquad\qquad s \geq 0, \end{aligned}$$

The homogeneous self dual model given at 3.1 was first studied by Goldman and Tucker [13].

$$\begin{aligned} Ax - b\kappa &= 0 \\ -A^T y \qquad + c\kappa &\geq 0 \\ b^T y - c^T x \qquad &\geq 0, \end{aligned} \tag{3.1}$$

where y is free, $x \geq 0$, $\kappa \geq 0$.

Let s, a vector and ρ a scalar denote the slacks for the second and third inequality constraints of (3.1). Goldman and Tucker have proved that the homogeneous model in (3.1) always has a non-trivial so-called strictly complementary solution (x^*, y^*, κ^*) such that

$$x^* s^* = 0, \qquad x_j^* + s_j^* > 0 \ \ \forall\, j, \tag{3.2}$$

$$\rho^* \kappa^* = 0, \qquad \rho^* + \kappa^* > 0, \tag{3.3}$$

where $x^* s^*$ denotes the component wise product of the vectors x^* and s^*.

Goldman and Tucker further showed that only one of the following can occur:

- $\kappa > 0$ if and only if (2.1) has an optimal solution,
- $\rho > 0$ if and only if (2.1) is primal or dual infeasible.

The homogeneous self-dual method provides all the necessary information to get an optimal solution or a certificate of infeasibility. Instead of solving (2.1), we aim to solve (3.1).

While (2.1) is self-dual, it can not satisfy the interior point condition. One can easily see that if x and y are primal feasible, and $\kappa \geq 0$, then due to weak duality property, the third inequality is always satisfied as equality.

We can modify (3.1) such that, while ensuring that the self dual property is preserved, we will have an interior point solution.

The following homogeneous self-dual model can be obtained from (3.1):

$$\begin{array}{llllllll}
\min & & & \beta\theta & & & & \\
\text{s.t} & Ax & -b\kappa & +\bar{b}\theta & & & & =0, \\
-A^Ty & & +c\kappa & -\bar{c}\theta & -s & & & =0, \\
b^Ty & -c^Tx & & -\alpha\theta & & -\rho & & =0, \\
-\bar{b}^Ty & +\bar{c}^Tx & -\alpha\kappa & & & & -\nu & =-\beta,
\end{array}$$
$$y \text{ is free, } x\geq 0,\ s\geq 0,\ \rho\geq 0,\ \theta\geq 0,\ \nu\geq 0,\ \kappa\geq 0, \tag{3.4}$$

where

$$\begin{aligned}
\bar{b} &= b+e-Ae, \\
\bar{c} &= c-e-A^Te, \\
\alpha &= 1+c^Te-b^Te, \\
\beta &= m+n+2.
\end{aligned}$$

We can see that $y^0=e,\ x^0=s^0=e,\ \theta^0=\rho^0=\kappa^0=\nu^0=1$ is an **interior feasible solution**. Hence we can say that the interior point condition holds. It is also easy to see that the embedding model is self dual. If an interior point exists for an LO problem, then the central path exists. For the above model, the central path is given by the set of solutions the equation system

$$\begin{array}{lllllll}
 & Ax & -b\kappa & +\bar{b}\theta & & & =0, \\
-A^Ty & & +c\kappa & -\bar{c}\theta & -s & & =0, \\
b^Ty & -c^Tx & & -\alpha\theta & & -\rho & =0, \\
-\bar{b}^Ty & +\bar{c}^Tx & -\alpha\kappa & & & -\nu & =-\beta,
\end{array} \tag{3.5}$$
$$\begin{aligned}
xs &= \mu e, \\
\kappa\rho &= \mu e, \\
\theta\nu &= \mu e,
\end{aligned}$$

$$y \text{ is free, } x\geq 0,\ s\geq 0,\ \rho\geq 0,\ \theta\geq 0,\ \nu\geq 0,\ \kappa\geq 0.$$

This system, for all $\mu > 0$ has a unique solution, $x(\mu),\ y(\mu),\ s(\mu),\ \kappa(\mu),\ \rho(\mu),\ \theta(\mu),\ \nu(\mu)$. It should be noted that xs represents the coordinate wise product of the vectors x and s.

The solution for the above system is usually difficult to calculate exactly. So, instead of solving the above central path system exactly, we use the Newton's method to solve it and get an approximate solution.

3.2 Newton System

The Newton direction is determined by the following system:

$$\begin{array}{rrrrrrrl}
 & A\Delta x & -\,b\Delta\kappa & +\,\bar{b}\Delta\theta & & & & = 0, \\
-\,A^T\Delta y & & +\,c\Delta\kappa & -\,\bar{c}\Delta\theta & -\,\Delta s & & & = 0, \\
b^T\Delta y & -\,c^T\Delta x & & -\,\alpha\Delta\theta & & -\,\Delta\rho & & = 0, \\
-\,\bar{b}^T\Delta y & +\,\bar{c}^T\Delta x & -\,\alpha\Delta\kappa & & & & -\,\Delta\nu & = 0, \\
 & s\Delta x & & & +\,x\Delta s & & & = \mu e - xs, \\
 & & \rho\Delta\kappa & & & +\,\kappa\Delta\rho & & = \mu e - \kappa\rho, \\
 & & & \nu\Delta\theta & & & +\,\theta\Delta\nu & = \mu e - \theta\nu.
\end{array} \tag{3.6}$$

The **Full Newton System** is as follows:

$$\left[\begin{array}{c|c|c|c|c|c|c}
 & A & -b & \bar{b} & & & \\ \hline
-A^T & & c & -\bar{c} & \mathrm{I} & & \\ \hline
b^T & \text{-c} & & -\alpha & & \text{-1} & \\ \hline
-\bar{b}^T & \bar{c}^T & \alpha & & & & \text{-1} \\ \hline
 & S & & & X & & \\ \hline
 & & \rho & & & \kappa & \\ \hline
 & & & \nu & & & \theta
\end{array}\right]
\left[\begin{array}{c}
\Delta y \\ \hline \Delta x \\ \hline \Delta\kappa \\ \hline \Delta\theta \\ \hline \Delta s \\ \hline \Delta\rho \\ \hline \Delta\nu
\end{array}\right]
=
\left[\begin{array}{c}
0 \\ \hline 0 \\ \hline 0 \\ \hline 0 \\ \hline \mu e - xs \\ \hline \mu - \rho\kappa \\ \hline \mu - \nu\theta
\end{array}\right]$$

The matrices X and S are diagonal matrices withe the vectors x and s as the principal diagonal, respectively

By pivoting on the last three sections, we have the **extended augmented system**:

$$\left[\begin{array}{c|c|c|c} & A & -b & \bar{b} \\ \hline -A^T & -X^{-1}S & c & -\bar{c} \\ \hline b^T & \text{-c} & \kappa^{-1}s & -\alpha \\ \hline -\bar{b}^T & \bar{c}^T & \alpha & \theta^{-1}\nu \end{array}\right] \left[\begin{array}{c} \Delta y \\ \hline \Delta x \\ \hline \Delta\kappa \\ \hline \Delta\theta \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mu X^{-1} - s \\ \hline \mu\kappa^{-1} - \rho \\ \hline \mu\theta^{-1} - \nu \end{array}\right]$$

We also have the following results:

$$\Delta\nu = \theta^{-1}\mu - \nu - \theta^{-1}\nu\Delta\theta,$$

$$\Delta\rho = \kappa^{-1}\mu - \rho - \kappa^{-1}\rho\Delta\kappa,$$

$$\Delta s = X^{-1}\mu - s - X^{-1}S\Delta x.$$

After pivoting on $-X^{-1}S$ block, the **extended normal equation** system is as follows:

$$\left[\begin{array}{c|c|c} AS^{-1}XA^T & r_1 & r_2 \\ \hline -r_1^T & \bar{\kappa} & \bar{\alpha} \\ \hline -r_2^T & \text{-}\bar{\alpha} & \bar{\theta} \end{array}\right] \left[\begin{array}{c} \Delta y \\ \hline \Delta\kappa \\ \hline \Delta\theta \end{array}\right] = \left[\begin{array}{c} \beta_1 \\ \hline \beta_2 \\ \hline \beta_3 \end{array}\right]$$

where,

$$\begin{aligned}
r_1 &= -b + AXS^{-1}c, \\
r_2 &= -\bar{b} - AXS^{-1}\bar{c}, \\
\bar{\kappa} &= \kappa^{-1}S - c^T S^{-1} X c, \\
\bar{\alpha} &= -\alpha + c^T S^{-1} X \bar{c}, \\
\bar{\theta} &= \theta^{-1}\nu - \bar{c}^T S^{-1} X c, \\
\beta_1 &= \mu A S^{-1} - Ax, \\
\beta_2 &= -\mu c S^{-1} e + cx + \mu\kappa^{-1} - \rho, \\
\beta_3 &= \mu \bar{c}^T S^{-1} e - \bar{c}^T x + \mu\theta^{-1} - \nu.
\end{aligned}$$

As documented by Maros and Mészáros [20], the normal equation system can be solved efficiently as it is only a system of linear equations with a symmetric positive definite coefficient matrix which can be solved efficiently with Cholesky factorization. The large scale problems can be solved efficiently provided that there are no dense columns in A. But if there are dense columns in A, then the Cholesky factorization will also result in dense factors which makes solving smaller problems difficult too.

The extended augmented system can be also be written (by using different variables to represent the blocks) as:

$$\left[\begin{array}{c:c} AS^{-1}XA^T & R \\ \hdashline -R^T & Q \end{array}\right]$$

Block pivoting on Q, we have

$$\underbrace{AS^{-1}XA^T}_{\text{Normal eqn. Matrix}} + \underbrace{RQ^{-1}R^T}_{\text{rank 2 update}}$$

We know that $\Delta\theta$ can be calculated explicitly ($\because\ \theta = \gamma\mu \implies \Delta\theta = \gamma\Delta\mu$). Hence, we only need a Rank-1 update. as explained by Anderesen et. al. [1]. The above low rank update can be computed efficiently by Sherman-Morrison formula. It should also be noted that if there are any dense columns, they can be reordered to the R part and then use the low rank update for efficient calculation

While the augmented system solves some of the problems associated with the Normal equations, it is still imperative that an efficient Cholseky factorization, which reduces the number of non zero elements, should be efficiently implemented in the LO software package. Most of the commercial IPM software packages will be performing pivot steps to identify an optimal basis in case a basis solution is preferred. This implies that a software should have good pivot algorithms for the optimal basis identification. We can see in the Chapter 5 how this affects the performance of the software.

The algorithmic framework is presented below. It should be noted that we are using the Mehrotra's Predictor-Corrector algorithm [21] in this algorithmic description

Algorithm 3 Interior Point Methods Algorithm

```
1:  **Initialization:**
2:  Given the homogeneous self-dual embedding model, a neighborhood $\mathcal{N}$ around the
    central path. Chose $\epsilon > 0$ for stopping criteria.
3:  $k = 0$
4:  **while** true **do**
5:      Do the Cholesky factorization for $x^k, s^k$
6:      **Predictor (virtual) step:**
7:      Let $\tau = 0$
8:      Solve the system (3.4) to get $\Delta x_p,\ \Delta s_p$ using the factorization in line 5.
9:      Find the largest $\alpha_p$ s.t. $x_p^{k+1}, s_p^{k=1} \in \mathcal{N}$. **Do not move**
10:     Calculate the corresponding $\mu_p^k$. The new $\tau^k = \left(\frac{\mu_p^k}{\mu^k}\right)^3 \mu^k$
11:     **Corrector Step**
12:     Start from the $x^k, s^k$ (in line 5).
13:     Use the above $\tau^k$ and solve the (3.4) to get $\Delta x_c,\ \Delta s_c$
14:     Find the largest $\alpha_p$ s.t. $(x^{k+1}, s^{k+1}) \in \mathcal{N}$. **Move now**
15:     **break** when $(x^{k+1})^T s^{k+1} \leq \epsilon\mu$ (stopping criteria)
16:     $k = k + 1$
17: **end while**
```

Chapter 4

Duality

Let us revisit the LO problem in standard form as given by (2.1):

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

Let us derive the dual from the primal problem. We have $c^T x = c^T x + 0$.

We also reformulate the constraint system as $Ax - b = 0$. Multiplying both the L.H.S and R.H.S with y^T,

$$y^T \big(Ax - b\big) = 0,$$

which can also be written as

$$0 = y^T \big(b - Ax\big).$$

Let us use this value of 0 in the modified objective function

$$c^T x = c^T x + y^T \big(b - Ax\big).$$

Rearranging the terms yields

$$c^T x = b^T y + \big(c - A^T y\big)^T x.$$

The primal problem includes the constraint $x \geq 0$, so if we have $\left(c - A^T y\right) \geq 0$, then we can get a lower bound

$$c^T x \geq b^T y, \text{ if } c - A^T y \geq 0.$$

In other words, the best lower bound on the primal objective gives us the dual problem.

$$\begin{aligned} \text{maximize} \quad & b^T x \\ \text{subject to} \quad & A^T y \leq c. \end{aligned}$$

The above approach is different from what von Neumann proposed to Dantzig [8]. As Dantzig recalled, von Neumann was working with Morgenstern on *Theory of Games* and he realized that what Dantzig's problem was equivalent to the problem he developed for game theory. Though von Neumann is credited for the origin of the concept of duality, Tucker, Kuhn and Gale [11] are credited for giving the first rigorous proof. It was also discovered later that the Farkas Lemma is equivalent to the strong duality theorem [3].

4.1 Weak and Strong Duality

The weak duality theorem states that if x is a primal feasible solution, and y is a dual feasible solution, then

$$c^T x \geq b^T y.$$

- In addition, if we have $c^T x = b^T y$, then both x and y are optimal.
- If $\left(c - A^T y\right)^T x = 0$, then considering $x \geq 0$ and $\left(c - A^T y\right) \geq 0$ we have, $\left(c - A^T y\right)_i x_i = 0 \; \forall i = 1 \ldots n$. Let us represent the vector $\left(c - A^T y\right)$ by the slack vector s, then we have $s \geq 0$ and $s^T x = 0$ implies $s_i x_i = 0 \; \forall i = 1 \ldots n$. This is also known as **complementary slackness**.

The **Strong Duality** theorem states that if the LO problem is both primal feasible and dual feasible, then there exists x^* which is primal feasible, y^* which is dual feasible with $c^T x^* = b^T y^*$.

As mentioned before, even though dual simplex method was discovered in 1954 by Lemke [18], the dual simplex method was primarily used in Mixed Integer Linear Optimization (MILO) to re-optimize sub-problems in the Branch and Bound tree as detailed in [5]. The two phase simplex method was developed for the primal simplex method aims to produce an initial BFS find BFS. The **Phase I** of the two phase simplex method deals with the problem of getting a feasible basic solution. The dual simplex method was not implemented in most of the commercial LO software even in the late 1980s. Bixby [4] noted that there is also another factor which made the dual simplex method very attractive — the "steepest-edge" rule. The steepest-edge rule is a pricing rule which determines the best variable to leave the basis at the end of a dual simplex iteration. The steepest edge rule was extensively discussed in [10]. Bixby also claimed that dual simplex with the"steepest-edge" pricing is more efficient than primal simplex method.

4.2 Taking the Duals of LO Problems

Most LO textbooks offer dualization schemes to get the dual form of a given problem. We also present a dualization scheme below.

$$\begin{array}{ll} \text{minimize} \quad c^T x & \qquad\qquad \text{maximize} \quad b^T y \\ Ax \left\{\begin{matrix} \le \\ \ge \\ = \end{matrix}\right\} b, & \qquad\qquad y \left\{\begin{matrix} \le \\ \ge \\ \text{free} \end{matrix}\right\} 0, \\ x \left\{\begin{matrix} \le \\ \ge \\ \text{free} \end{matrix}\right\} 0, & \qquad\qquad A^T y \left\{\begin{matrix} \le \\ \ge \\ = \end{matrix}\right\} b. \end{array}$$

Primal Problem	**Dual Problem**
minimize	maximize
Constraints	**Variables**
$\sum_{j=1}^n a_{ij}x_j \ge b_i$	$y_i \ge 0$
$\sum_{j=1}^n a_{ij}x_j = b_i$	y_i is free
$\sum_{j=1}^n a_{ij}x_j \le b_i$	$y_i \le 0$
Variables	**Constraints**
x_j is free	$a_j^T y = c_j$
$x_j \ge 0$	$a_j^T y \le c_j$
$x_j \le 0$	$a_j^T y \ge c_j$
$x_j = 0$	no constraint

The dualization scheme presented above can be used to dualize any given LO problem. As you can see, the objective function coefficients become the right hand side of the dual problem. The whole A matrix is transposed and the sign of the constraints determine the sign of the dual variables. Similarly the sign of the primal variables determine the sign of the dual constraints.

4.3 More on Dualization

We have used a few more dualizing steps and created a python program called `dualize.py` which dualizes
the problems in the NETLIB set. The NETLIB[1] site contains a collection of LO problems , mostly real world problems which are used to test the LO software. Most of the problems are degenerate and very sparse. The problems in the NETLIB set can be either involve the maximization or minimization of the objective function.

It is very straightforward to dualize LO problems without bounds,

Let us consider the following optimization problem:

$$
\begin{aligned}
\text{minimize} \quad & x_1 + x_2 + 3x_3 + x_4 + 3x_5 + x_6 + 3x_7 + x_8 + x_9 + x_{10} \\
\text{subject to} \quad & x_1 + 2x_3 + 3x_4 + 2x_7 + 3x_8 + 9x_9 \geq 20, \\
& 2x_2 + 4x_6 + 3x_5 + 4x_9 + 9x_{10} \geq 30, \\
& 3x_3 + 4x_6 + x_9 \geq 10, \\
& x_5 + 3x_8 + 4x_{10} \geq 15, \\
& 2x_1 + 3x_3 + x_9 \geq 5, \\
& x_1 + x_6 + x_9 \leq 20, \\
& x_1 + 2x_6 + 2x_9 + 3x_{10} = 10, \\
& x_1, x_2 \ldots x_{10} \geq 0.
\end{aligned} \tag{4.1}
$$

The MPS file format is developed by IBM to represent the LO problems. the format takes the advantage of sparsity and represents only those values which are not zero.

Using the techniques described in Section 4.2, the objective function will be on the right hand side of the dual problem. Since all the variables are non-negative, the above techniques dictate that all the constraints will be of the type “Less than or equal”. The sixth constraint is of the type “less than or equal” and hence the dual variable associated with it y_6 is less than or equal to zero. Similarly, the last constraint is an equality constraint and the dual variable associated with the constraint is free.

Thus, the dual problem is:

[1] http://www.netlib.org/lp/data/index.html

$$
\begin{aligned}
\text{maximize} \quad & 20y_1 + 30y_2 + 10y_3 + 15y_4 + 5x_5 + 20y_6 + 10y_7 \\
\text{subject to} \quad & y_1 + 2y_5 + y_6 + y_7 \leq 1, \\
& 2y_2 \leq 1, \\
& 2y_1 + 3y_3 + 3y_5 \leq 3, \\
& 3y_1 \leq 1, \\
& y_4 \leq 3, \\
& 4y_2 + 4y_3 + y_6 + 2y_7 \leq 1, \\
& 2y_1 \leq 3, \\
& 3y_1 + 3y_2 + 3y_4 \leq 1, \\
& 9y_1 + 4y_2 + y_3 + y_5 + y_6 + 2y_7 \leq 1, \\
& 9y_2 + 4y_4 + 3y_7 \leq 1, \\
& y_1, y_2 \dots y_5 \geq 0, \\
& y_6 \leq 0, \\
& y_7 \text{ free.}
\end{aligned} \tag{4.2}
$$

The corresponding `.mps` file is given in Appendix A

When bounds are involved, we need a two step process for the dualization. We need to convert the bounds to constraints and then dualize the modified problem.

There can be 4 different cases of variable bounds:

- variables having fixing bounds
- variables only having upper bounds
- variables only having lower bounds
- variables having both upper and lower bounds

Fixing bounds: When we encounter variables having equal lower and upper bounds, we can substitute its fix value and remove the variables from the dualization process.

Non-negative variables with upper bounds: These bounds can be transformed as additional constraints to the original problem

Only lower bounds: These bounds are going to be transformed by the use of substitution. For instance, if we have the bounds $5 \leq x$, we do the following manipulation.

$$5 \leq x_i \implies x_i - 5 \geq 0 \implies \bar{x}_i \geq 0, \text{ where } \bar{x}_i = x_i - 5$$

Now if we replace x with $\bar{x} + 5$ in the original system, we have transformed the lower bound to zero, thus the new variable $\bar{x}_i$ is standard non-negative in the transformed problem.

Both upper and lower bounds: We use a combination of variable substitution and constraint addition to change the primal problem and then dualize it. For instance,if we have $5 \leq x_i \leq 8$.

$$0 \leq x_i - 5 \leq 8 - 5,$$

$$0 \leq \bar{x}_i \leq 3, \text{ where } \bar{x}_i = x_i - 5,$$

$$\implies \bar{x}_i \geq 0 \text{ and } \bar{x}_i \leq 3.$$

Replacing x by $\bar{x} + 5$ and adding the $\bar{x} \leq 3$ constraint for the non-negative variable $\bar{x}_i$. Thus we have reformulated the problem into a problem without any bounds, which can be dulaized as described in the Section 4.2.

Let us see a dualizing example using the above step:

$$\begin{aligned} \text{minimize} \quad & -x_1 - 2x_2 - 3x_3 - x_4 \\ \text{subject to} \quad & -x_1 + x_2 + x_3 + 10x_4 \leq 20, \\ & x_1 - 3x_2 + x_3 \leq 30, \\ & x_2 - 3.5x_4 = 10, \\ & 2 \leq x_4 \leq 4, \\ & x_1, x_2, x_3 \geq 0. \end{aligned} \tag{4.3}$$

The dual problem obtained by using the dualization techniques discussed at Section 4.3 is as follows:

$$\begin{aligned} \text{maximize} \quad & 0y_1 + 30y_2 + 7y_3 + 2y_4 - 2 \\ \text{subject to} \quad & -y_1 + y_2 \leq -1, \\ & y_1 - 3y_2 + y_3 \leq -2, \\ & y_1 + y_2 \leq -3, \\ & 3.5y_3 + y_4 \leq -1.0, \\ & y_1, y_2, y_4 \leq 0, \\ & y_3 \text{ is free.} \end{aligned} \tag{4.4}$$

Here y_4 is the dual variable associated with the bound constraint. The MPS file can be found in the Appendix A.

In the next chapter, we present some numerical examples and discuss how various commercial software solves them.

Chapter 5

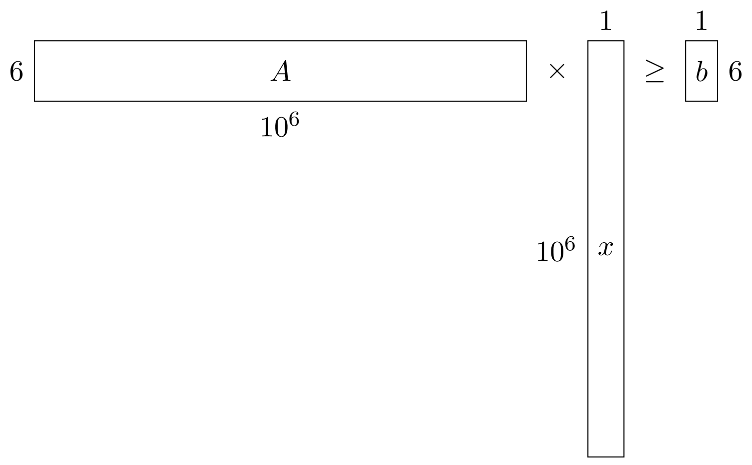
Numerical Experiments

In this chapter, we will discuss some numerical experiments which highlight the importance of duality in software implementation. We present examples where CPLEX, GuRoBi and MOSEK have some issues. We hypothesize the cause for such issues and report the results below

5.1 Unbalanced Problems: Large Number of Columns

The unbalanced problems have very small number of rows and significantly large number of columns. We will see how each software deals with this kind of LO problems

$$\text{Primal problem:} \qquad \min \quad c^T x \quad \text{s.t.} \quad Ax \geq b, \quad x \geq 0$$



We have used MATLAB to generate the prototype unbalanced problem. For the sake of discussion, we will call the form represented above as primal and then also generate the corresponding dual of this primal problem via MATLAB.

Using the Primal Simplex and Dual Simplex Algorithms without presolve for GuRoBi, CPLEX and MOSEK, and for both the primal and dual unbalanced problems:

Table 5.1: Unbalanced problems solved using GuRoBi.

Problem	PS Time	PS Iter	DS Time	DS Iter
Primal	2.19	16	6.46	47
Dual	8.3	49	6.22	23

Table 5.2: Unbalanced problems solved using MOSEK.

Problem	PS Time	PS Iter	DS Time	DS Iter
Primal	2.49	60	13.52	51
Dual	18.99	34	11.95	18

Table 5.3: Unbalanced problems solved using CPLEX.

Problem	PS Time	PS Iter	DS Time	DS Iter
Primal	0.91	36	No Solve	No Solve
Dual	No Solve	No Solve	5.73	19

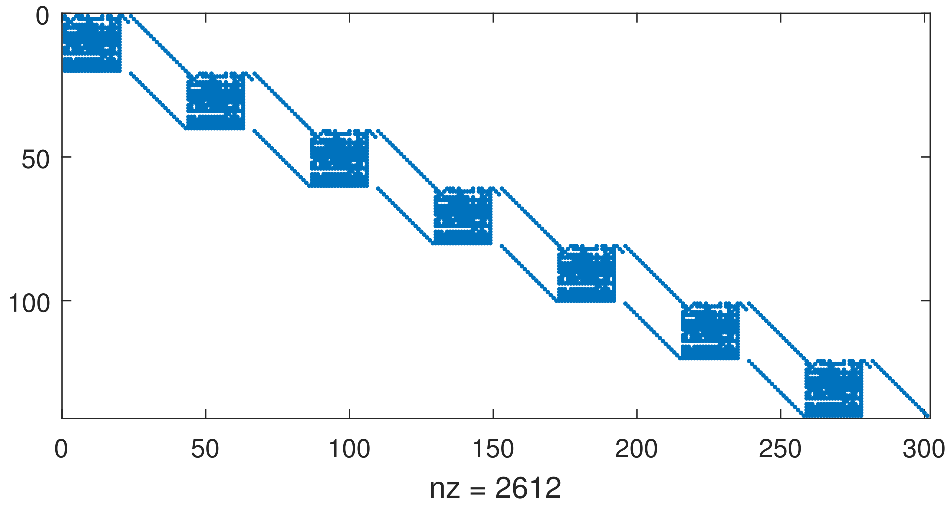
The tables (5.1), (5.2) and (5.3) detail the results of solving the unbalanced problem with GuRobi, MOSEK and CPLEX, respectively. As one can see, CPLEX struggles to solve such problems. Even when the presolve is allowed CPLEX fails to solve the problem in a reasonable amount of time.

CPLEX, when asked to solve the unbalanced primal problem with Dual simplex, the model tries to dualize and form the basis tableau with all the 10^6 columns and with probable lack of dual sifting in CPLEX's implementation of dual simplex, it becomes very difficult to solve. Analogously, as duality predicts, the primal simplex of CPLEX is struggling to solve the dual problem.

5.2 Staircase/Grow Problems

The staircase problems are a small subset of the NETLIB continuous optimization test set namely, grow7, grow15 and grow22. The structure of the grow problems is displayed at fig 5.1. The shaded region represents the position of non-zero entities in the constraint matrix.

FIGURE 5.1: Grow problem structure.



Using MATLAB, we have extended the problems to include more "stairs" and created grow problems with 36, 71, 107 and 176 blocks. The problems can be dualized using `Dualize.py` python program. The growxxP problem name refers to the modified primal problem and growxxDual referes to the Dual of this modified problem. The problem data is presented in Table 5.4

TABLE 5.4: Extended Grow problems data.

Problem	Rows	Cols	NZs
grow36	720	1548	13516
grow36P	2160	1548	14956
grow36Dual	1548	2160	14956
grow71	1420	3053	26667
grow71P	4260	3053	29516
grow71Dual	3053	4260	29516
grow107	2120	4558	39836
grow107P	6360	4558	44076
grow107Dual	4558	6360	44076
grow176	3520	7568	66156
grow176P	10560	7568	73196
grow176Dual	7568	10560	73196

The results have been tabulated in the Table 5.5. As one can see GuRoBi fails with the dualized problems of the grow problem. CPLEX does not face similar issues. The main issues could be some implementation issues with the linear algebra core of GuRoBi's IPM. The dual form of the problem is causing serious issues to the linear algebra core of the IPM of GuRoBi.

Table 5.5: Grow problems solved using IPMS of GuRoBi and CPLEX.

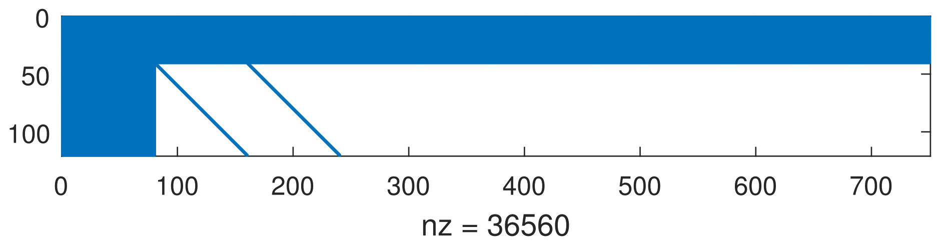
Problem	GuRoBi				CPLEX			
	AA' nz	Fac. Nz	Time	Iter	AA' nz	Fac. Nz	Time	Iter
grow36	7540	25290	0.06	13	7540	21861	0.11	15
grow36P	19360	61250	0.08	12	7540	21861	0.12	15
grow36D	32584	138400	0.12	17	32584	71376	0.29	17
grow71	14890	50550	0.09	16	14890	43561	0.21	20
grow71P	38230	123100	0.15	17	14890	43561	0.28	20
grow71D	64609	331400	2.65	223	64609	149012	0.48	20
grow106	22240	75720	0.14	21	22240	65310	0.33	22
grow106P	57090	184000	0.23	19	22240	65310	0.37	22
grow106D	96634	370200	8.41	401	96634	328450	2.69	28
grow176	36940	126200	0.2	21	36940	108661	0.63	25
grow176P	94820	305900	0.38	20	36940	108661	0.61	25
grow176D	160700	844800	4.81	146	160684	451502	2.05	28

It can be seen from the Table 5.5 Gurobi takes extremely large number of solutions to find the solution. It should also be noted that for grow71D, grow106D and grow176D (the duals of the grow problems), the barrier algorithm terminates suboptimaqlly. The Cholesky factorization of the A matrix is also inefficient when compared to CPLEX. In-fact the number of non zeros in the Cholesky factors are almost double the number of non zeros of CPLEX's factor.

5.3 L-Shaped Problems

We use the following L-shaped LO structure as a template to generate problems with various column densities. The structure of the L-Shaped problems is given at the fig 5.2. The shaded region indicates the non zero entries in the coefficient matrix.

Figure 5.2: L-Shaped problem structure.



These problems highlight the problems that can be caused by inefficient implementation of linear algebra core for interior point methods.

TABLE 5.6: L-shaped problems solved by IPM MOSEK.

Problem	Factor NZ	Time	Iter	Basis Identification
C=5 Primal + No Dualization	1.10E+004	38.72	53	Works
C=5 Primal + Dualization	4.51E+006	39.26	17	Not Working
C=5 Dual + No Dualization	4.51E+006	38.21	17	Not Working
C=5 Dual + Dualization	1.10E+004	33.56	53	Works
C=15 Primal + No Dualization	2.11E+004	36.26	44	Works
C=15 Primal + Dualization	4.53E+006	35.22	14	Not Working
C=15 Dual + No Dualization	4.53E+006	35.59	14	Not Working
C=15 Dual + Dualization	2.11E+004	32.1	44	Works
C=25 Primal + No Dualization	3.13E+004	35.67	47	Works
C=25 Primal + Dualization	4.55E+006	28.18	12	Not Working
C=25 Dual + No Dualization	4.55E+006	28.48	12	Not Working
C=25 Dual + Dualization	3.13E+004	33.43	47	Works
C=45 Primal + No Dualization	5.20E+004	27.27	37	Works
C=45 Primal + Dualization	4.59E+006	78.34	30	Not Working
C=45 Dual + No Dualization	4.59E+006	76.4	30	Not Working
C=45 Dual + Dualization	5.20E+004	26.01	37	Works

The Table 5.6 gives us the results of solving the L-shaped problems using MOSEK. The value of C denotes the number of dense columns. Let us consider the case when the problem has 45 dense columns, and we force the software to take a dual and solve the problem[1], we notice a few interesting things.

- The reduction in μ is very very slow. It takes 21 iterations to get an order of magnitude reduction in the μ value.
- The basis identification doesn't progress when performed by dual simplex method

The slow reduction in μ can be attributed to the presence of dense columns in the dual. This probably leads to the decrease in the quality of the search direction. The basis identification for the dual problem, which can be obtained by either solving the dual problem

[1]The MOSEK parameters to solve the problem can be found in the Appendix

or dualizing the primal problem using the software, uses the dual simplex method to identify the basis. However, as it can be observed from the MOSEK experiments on the NETLIB set, the dual simplex method of MOSEK is not very stable compared to the primal simplex method. It often switches to primal simplex method as it probably loses feasibility. Because of this, the basis identification is not progressing.

5.4 Experiments with the NETLIB Test Set:

For pivot algorithms, we see in practice that when the number of columns is approximately same as the number of rows, the primal simplex and dual simplex method perform about the same.

Table 5.7: Results of solving Finnis.problems with GuRoBi, CPLEX and MOSEK.

Problem	rows	columns	NZs	GuRoBi			
				PS Time	PS Iter	DS Time	DS Iter
FINNIS	497	614	2310	0.01	595	0	467
FINNIS_P	533	569	2128	0.01	645	0.01	529
FINNIS_D	569	533	2128	0.01	538	0.01	572

Problem	rows	columns	NZs	CPLEX			
				PS Time	PS Iter	DS Time	DS Iter
FINNIS	497	614	2310	0.01	463	0.01	382
FINNIS_P	533	569	2128	0.01	488	0.01	397
FINNIS_D	569	533	2128	0.01	408	0.02	494

Problem	rows	columns	NZs	MOSEK			
				PS Time	PS Iter	DS Time	DS Iter
FINNIS	497	614	2310	0.02	603	0.01	341
FINNIS_P	533	569	2128	0.02	571	0.01	349
FINNIS_D	569	533	2128	0.02	507	0.02	422

However, by utilizing the structure of specific problems, some methods are very very effective in solving those problems compared to other methods. From the NETLIB set,

TABLE 5.8: Results of solving Firt2d problems with GuRoBi, CPLEX and MOSEK.

Problem	rows	cols	NZs	GuRoBi			
				PS Time	PS Iter	DS Time	DS Iter
FIT2D	25	10500	129018	0.34	27853	0.09	268
FIT2D_P	10525	10500	139518	5.4	19755	3.18	9608
FIT2D_D	10500	10525	139518	3.2	11605	10.18	13920

Problem	rows	columns	NZs	CPLEX			
				PS Time	PS Iter	DS Time	DS Iter
FIT2D	25	10500	129018	0.46	13938	0.28	189
FIT2D_P	10525	10500	139518	13.23	18090	8.4	11432
FIT2D_D	10500	10525	139518	6.74	12345	8.2	6171

Problem	rows	columns	NZs	MOSEK			
				PS Time	PS Iter	DS Time	DS Iter
FIT2D	25	10500	129018	1.88	50104	0.17 + 0.02	150+30(PS)
FIT2D_P	10525	10500	139518	24.17	40260	0.96 +0.04	4810+22(PS)
FIT2D_D	10500	10525	139518	7.25	5246	15.39+0.03	5738+8(PS)

Despite the structure being favorable to the dual simplex method, MOSEK's dual simplex method switches to primal simplex method to solve the problem.

From the tables in the Appendix on solving the NETLIB test problems using simplex algorithms, we can see similar results displayed in Table 5.7. For example, the tabulated results of problems **E226**, **ISRAEL** from Table A.5 and problems **SCRRS8** and **SHARE2B** from Table A.6 show that both the simplex methods take roughly the same number of iterations to solve the problems.

We can further find some more problems from the NETLIB tables in the Appendix which give similar results to Table 5.8. For instance, the results of **MODSZK1**, **FIT1D** and **CYCLE** show that dual simplex method is very effective in solving the problems for all the three software packages. The dual simplex method of MOSEK again faces troubles and switches to primal simplex method for the **CYCLE** problem.

Chapter 6

Conclusions

In Chapter 2 we demonstrated that the dual simplex method is exactly same as the primal simplex method applied to the dual LO problem, at least in theory. However, the same can not be said in practice. As discussed in Chapter 1, while the core of the algorithm has not changed, there have been many modifications to the algorithm, for instance the implementation of various pricing rules. In practice, machine accuracy and floating point arithmetic also leads to issues while implementing LO algorithms.

One significant outcome of the computational experiments is that the commercial software's implementation of the primal simplex and dual simplex methods is mostly the same. However, for some special problems, like the unbalanced problems for CPLEX, it is highlighted that their implementations are widely different. Similarly, MOSEK's dual simplex implementation leaves a lot to be desired when compared to the primal simplex implementation. The dual simplex algorithm of MOSEK seems to encounter numerical difficulties more often than the primal simplex and uses perturbation, and frequently switches to primal simplex method to finish solving the problem.

For IPMs, the linear algebra core is extremely important. As we can see, the Cholesky factorization of GuRoBi when compared to CPLEX's interior point method is less efficient. This sometimes leads to less reliable search directions and sometimes sub-optimal termination of the barrier methods.

The IPMs can not provide the optimal basis solution and whenever the basis solution is needed, pivots are performed to identify the basis. Therefore it is necessary to have a good implementation of the pivot algorithms too. We saw the importance of the pivot algorithms for basis identification while solving the L-shaped problems in Chapter 5

As we can observe from the results of our experiments on the NETLIB test set, all the three software packages can solve the problems with relative ease. Most of the times the number of primal simplex iterations and number of dual simplex iterations are roughly the same except for a few problems where structure can be exploited much efficiently by the dual simplex method. The NETLIB set only contains the "primal" version of the problem. By taking the duals of these primal problems and solving them using the software packages we can see that the various simplex methods of different software packages perform similarly. Considering the number of iterations, the primal simplex method on primal problems performs similarly to the dual simplex method on dual problems. Analogously, the primal simplex method's performance on the dual problems is similar to the dual simplex method's performance on the primal problems. For these type of problems, we can see that the cost of pivoting by the primal simplex method and dual simplex method is roughly the same and so the total time to solve the problems is also roughly the same for either algorithm. The results also highlighted the dual simplex implementation problems in MOSEK.

We have highlighted a few shortcomings of the implementation of various algorithms in different LO software packages. Be it the unbalanced problems for CPLEX, or the staircase problems for GuRoBi, or the the basis identification issues for MOSEK, it is important to know that due to different implementations, computational practice may differ significantly from theory.

In practice, we do not know if we are getting a primal problem or a dual problem. Any modifications that can be done to the primal simplex method can be done to the dual simplex method too. As we have seen there is no full symmetry between implementations with respect to duality, in practice. The linear algebra core makes a huge difference in the efficiency of IPMs. As we saw, Gurobi's IPM was struggling to solve the dual form of the grow problems. Further MOSEK constantly switches to primal simplex method despite asking to solve the problem by dual simplex method and never the other way around.

Overall the goal of this thesis was to highlight how duality can be exploited while designing LO software. We have contacted the three software vendors and showed the results to them and they were able to reproduce these issues and informed us that they are working on resolving them.

Appendix A

A.1 Primal and Dual .mps files

The `.mps` file format for primal problem at (4.1) is given below:

```
     NAME           TESTPROB
ROWS
 N  COST
 G  LIM1
 G  LIM2
 G  LIM3
 G  LIM4
 G  LIM5
 L  LIM6
 E  LIM7
COLUMNS
    X1        COST                 1   LIM1                 1
    X1        LIM5                 2   LIM6                 1
    X1        LIM7                 1
    X2        LIM2                 2   COST                 1
    X3        COST                 3   LIM1                 2
    X3        LIM3                 3   LIM5                 3
    X4        LIM1                 3   COST                 1
    X5        COST                 3   LIM4                 1
    X6        LIM2                 4   LIM3                 4
    X6        COST                 1   LIM6                 1
    X6        LIM7                 2
    X7        COST                 3   LIM1                 2
    X8        LIM1                 3   LIM2                 3
    X8        LIM4                3   COST                1
    X9        LIM1                9   LIM2                4
    X9        LIM3                1   LIM5                1
    X9        COST                1   LIM6                1
    X9        LIM7                2
    X10       LIM2                9   LIM4                4
    X10       COST                1   LIM7                3
RHS
    RHS1      LIM1               20   LIM2               30
    RHS1      LIM3               10   LIM4               15
    RHS1      LIM5                5   LIM6               20
    RHS1      LIM7               10
ENDATA
```

The `.mps` file format for dual problem at (4.2) is given below:

```
     NAME          TESTDUAL
OBJSENSE
    MAX
ROWS
 N  RHS1
 L  X1
 L  X2
 L  X3
 L  X4
 L  X5
 L  X6
 L  X7
 L  X8
 L  X9
 L  X10
COLUMNS
    LIM1      RHS1      20.0
    LIM1      X1        1.0
    LIM1      X3        2.0
    LIM1      X4        3.0
    LIM1      X7        2.0
    LIM1      X8        3.0
    LIM1      X9        9.0
    LIM2      RHS1      30.0
    LIM2      X2        2.0
    LIM2      X6        4.0
    LIM2      X8        3.0
    LIM2      X9        4.0
    LIM2      X10       9.0
    LIM3      RHS1      10.0
    LIM3      X3        3.0
    LIM3      X6        4.0
    LIM3      X9        1.0
    LIM4      RHS1      15.0
    LIM4      X5        1.0
    LIM4      X8        3.0
    LIM4      X10       4.0
    LIM5      RHS1      5.0
    LIM5      X1        2.0
    LIM5      X3        3.0
    LIM5      X9        1.0
    LIM6      RHS1      20.0
    LIM6      X1        1.0
    LIM6      X6        1.0
    LIM6      X9        1.0
    LIM7      RHS1      10.0
    LIM7      X1        1.0
    LIM7      X6        2.0
    LIM7      X9        2.0
    LIM7      X10       3.0
RHS
    COST      X1        1.0
    COST      X2        1.0
    COST      X3        3.0
    COST      X4        1.0
    COST      X5        3.0
    COST      X6        1.0
    COST      X7        3.0
    COST      X8        1.0
    COST      X9        1.0
    COST      X10       1.0
```

```
BOUNDS
 MI B1         LIM6
 UP B1         LIM6      0
 FR B1         LIM7
ENDATA
```

The `.mps` file format for the primal problem at (4.3) is given below:

```
 NAME           TESTPROB
ROWS
 N  obj
 L  c1
 L  c2
 E  c3
COLUMNS
    x1         obj                 -1    c1                  -1
    x1         c2                   1
    x2         obj                 -2    c1                   1
    x2         c2                  -3    c3                   1
    x3         obj                 -3    c1                   1
    x3         c2                   1
    x4         obj                 -1    c1                  10
    x4         c3                -3.5
RHS
    RHS1       c1                  20     c2                 30
BOUNDS
 LO BND1       x4                   2
 UP BND1       x4                   4
ENDATA
```

The `.mps` file format for the dual problem at (4.4) is given below:

```
NAME           TESTDUAL
OBJSENSE
    MAX
ROWS
```

```
 N  RHS1
 L  x1
 L  x2
 L  x3
 L  x4
COLUMNS
    c1        RHS1      0.0
    c1        x1        -1.0
    c1        x2        1.0
    c1        x3        1.0
    c1        x4        10.0
    c2        RHS1      30.0
    c2        x1        1.0
    c2        x2        -3.0
    c2        x3        1.0
    c3        RHS1      7.0
    c3        x2        1.0
    c3        x4        -3.5
    x4BC      RHS1      2.0
    x4BC      x4        1
RHS
    obj       x1        -1.0
    obj       x2        -2.0
    obj       x3        -3.0
    obj       x4        -1.0
    obj       RHS1       -2.0
BOUNDS
 MI B1        c1
 UP B1        c1        0
 MI B1        c2
 UP B1        c2        0
 FR B1        c3
 MI B1        x4BC
 UP B1        x4BC      0
ENDATA
```

A.2 NETLIB Experiments

Table A.1: NETLIB Linear optimization test set data

Name	Rows	Equlaities	Inequlities	Cols	UP bnd	LO bnd	Free	FX	nnz
25FV47	822	516	305	1571	–	0	–	0	11127
80BAU3B	2263	0	2262	9799	2731	71	0	498	29063
ADLITTLE	57	15	41	97	0	0	0	0	465
AFIRO	28	8	19	32	0	0	0	0	88
AGG	489	36	452	163	0	0	0	0	2541
AGG2	517	60	456	302	0	0	0	0	4515
AGG3	517	60	456	302	0	0	0	0	4531
BANDM	306	305	0	472	0	0	0	0	2659
BEACONFD	174	140	33	262	0	0	0	0	3476
BLEND	75	43	31	83	0	0	0	0	521
BNL1	644	232	411	1175	0	0	0	0	6129
BNL2	2325	1327	997	3489	0	0	0	0	16124
BORE3D	234	214	19	315	11	1	0	1	1525
BRANDY	221	166	54	249	0	0	0	0	2150
CAPRI	272	142	129	353	131	0	14	16	1786
CYCLE	1904	1389	514	2857	77	0	7	0	21322
CZPROB	930	890	39	3523	0	0	0	229	14173
D2Q06C	2172	1507	664	5167	0	0	0	0	35674
D6CUBE	416	415	0	6184	0	1	0	0	43888
DEGEN2	445	221	223	534	0	0	0	0	4449
DEGEN3	1504	717	786	1818	0	0	0	0	26230
DFL001	6072	6071	0	12230	13	0	0	0	41873
E226	224	33	190	282	0	0	0	0	2767
ETAMACRO	401	272	128	688	135	45	0	82	2489
FFFFF800	525	350	174	854	0	0	0	0	6235
FINNIS	498	47	450	614	36	41	0	45	2714
FIT1D	25	1	23	1026	1026	0	0	0	14430
FIT1P	628	627	0	1677	399	0	0	0	10894
FIT2D	26	1	24	10500	0	0	0	0	138018
FIT2P	3001	0	0	13525	0	0	0	0	60784
GANGES	1310	1284	25	1681	0	7	0	0	7021
GFRD-PNC	617	548	68	1092	256	0	0	0	3467
GREENBEA	2393	2199	193	5405	290	39	0	103	31499
GREENBEB	2393	2199	193	5405	291	26	4	115	31499

Table A.2: NETLIB Linear optimization test set data (contd.)

Name	Rows	Equlaities	Inequlities	Cols	UP bnd	LO bnd	Free	FX	nnz
GROW15	301	300	0	645	600	0	0	0	5665
GROW22	441	440	0	946	880	0	0	0	8318
GROW7	141	140	0	301	280	0	0	0	2633
ISRAEL	175	0	174	142	0	0	0	0	2358
KB2	44	16	27	41	9	0	0	0	291
LOTFI	154	95	58	308	0	0	0	0	1086
MAROS	847	323	523	1443	0	6	0	35	10006
MAROS-R7	3137	3136	0	9408	0	0	0	0	151120
MODSZK1	688	687	0	1620	0	0	0	0	4158
PEROLD	626	495	130	1376	266	7	88	64	6026
PILOT	1442	233	1208	3652	1041	90	0	167	43220
PILOT.JA	941	661	279	1988	333	0	88	311	14706
PILOT.WE	723	661	279	2789	333	0	88	311	9218
PILOT4	411	287	123	1000	247	0	88	30	5145
PILOT87	2031	233	1797	4883	1400	115	0	180	73804
PILOTNOV	976	701	274	2172	340	0	0	204	13129
RECIPE	92	67	24	180	50	4	0	24	752
SC105	106	45	60	103	0	0	0	0	281
SC205	206	91	114	203	0	0	0	0	552
SC50A	51	20	30	48	0	0	0	0	131
SC50B	51	20	30	48	0	0	0	0	119
SCAGR25	472	300	171	500	0	0	0	0	2029
SCAGR7	130	84	45	140	0	0	0	0	553
SCFXM1	331	187	143	457	0	0	0	0	2612
SCFXM2	661	374	286	914	0	0	0	0	5229
SCFXM3	991	561	429	1371	0	0	0	0	7846
SCORPION	389	280	108	358	0	0	0	0	1708
SCRS8	491	384	106	1169	0	0	0	0	4029
SCSD1	78	77	0	760	0	0	0	0	3148
SCSD6	148	147	0	1350	0	0	0	0	5666
SCSD8	398	0	0	2750	0	0	0	0	11334
SCTAP1	301	120	180	480	0	0	0	0	2052
SCTAP2	1091	470	620	1880	0	0	0	0	8124
SCTAP3	1481	620	860	2480	0	0	0	0	10734
SHARE1B	118	89	28	225	0	0	0	0	1182
SHARE2B	97	13	83	79	0	0	0	0	730
SHELL	537	534	2	1775	117	9	0	250	4900
SHIP04L	403	354	48	2118	0	0	0	0	8450
SHIP04S	403	354	48	1458	0	0	0	0	5810
SHIP08L	779	698	80	4283	0	0	0	0	17085
SHIP08S	779	698	80	2387	0	0	0	0	9501
SHIP12L	1152	1045	106	5427	0	0	0	0	21597
SHIP12S	1152	1045	106	2763	0	0	0	0	10941

Table A.3: NETLIB Linear optimization test set data (contd.)

SIERRA	1228	528	699	2036	2036	0	0	0	9252
STAIR	357	209	147	467	6	0	6	82	3857
STANDATA	360	160	199	1075	104	0	0	16	3038
STANDGUB	362	0	0	1184	0	0	0	0	3147
STANDMPS	468	268	1199	1075	104	0	0	16	3686
STOCFOR1	118	63	54	111	0	0	0	0	474
STOCFOR2	2158	1143	1014	2031	0	0	0	0	9492
STOCFOR3	16676	0	0	15695	0	0	0	0	74004
TRUSS	1001	0	0	8806	0	0	0	0	36642
TUFF	334	292	41	587	24	0	2	3	4523
VTP.BASE	199	0	0	203	0	0	0	0	914
WOOD1P	245	243	1	2594	0	0	0	0	70216
WOODW	1099	1085	13	8405	0	0	0	0	37478

A.3 NETLIB problems – No Bounds

Table A.4: NETLIB problems without Bounds

Problem	CPLEX PS		CPLEX DS		GuRoBi PS		GuRoBi DS		MOSEK PS		MOSEK DS	
	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter
25FV47	0.34	2406	0.28	1888	0.2	2218	0.16	2089	0.96	4972	0.85	3350+1(PS)
25FV47Dual	0.5	2982	0.38	2113	31	2513	27	2217	0.83	4323	0.91	2172 +2(PS)
ADLITTLE	0	94	0	89	0	126	0.02	81	0	114	0	91
ADLITTLEDUal	0.01	111	0	86	0	90	0.02	91	0	184	0.01	81
AFIRO	0	13	0	13	0	14	0	10	0	9	10	0
AFIRODual	0	22	0	16	0	20	0	16	0	13	0	15
AGG	0.01	105	0.01	125	0	130	0	185	0.01	101	0.01	69
AGGDual	0.01	241	0.01	144	0	245	0.02	124	0	156	0.01	110
AGG2	0.01	120	0.01	133	0	181	0	174	0.01	171	0.01	139
AGG2Dual	0.03	383	0.01	186	0	297	0.02	207	0	139	0.01	182
AGG3	0.01	149	0.01	133	0	177	0	172	0.01	201	0.01	133
AGG3Dual	0.03	401	0.01	181	0	267	0	202	0	154	0.01	174
BANDM	0.02	309	0.04	395	0.02	339	0.02	427	0.03	542	0.03	361
BANDMDual	0.04	546	0.04	566	0.02	589	0.02	627	0.05	723	0.05	415
BEACONFD	0	31	0.01	170	0.02	34	0.02	220	0	54	0.01	141
BEACONFDDual	0.01	178	0.01	122	0	277	0	171	0.01	183	0.01	106
BNL1	0.11	1816	0.07	819	0.09	2366	0.03	1018	0.21	1625	0.12	1117
BNL1Dual	0.09	1312	0.12	1188	0.08	1578	0.11	1262	0.1	1132	0.26	1306
BNL2	0.73	4201	0.15	1408	0.37	4784	0.08	1833	1.24	5475	0.24	1852
BNL2Dual	0.17	2199	0.59	3731	0.37	3251	0.25	3329	0.22	2085	1.28	2993
BRANDY	0.02	117	0.02	181	0.02	195	0.02	419	0.02	302	0.02	298
BRANDYDual	0.03	338	0.02	278	0.02	315	0	342	0.03	494	0.02	211
D2Q06C	2.78	6803	1.95	4874	1.61	8279	1.02	5129	7.35	20489	4.01+0.02	6651 + 13(PS)
D2Q06CDual	4.68	9115	2.95	6672	2.36	7602	1.92	7613	11.42	18288	6.66	5983
DEGEN2	0.08	1255	0.04	392	0.03	667	0.02	533	0.09	1009	0.07	574
DEGEN2Dual	0.06	886	0.04	679	0.03	765	0.03	840	0.07	924	0.11	847
DEGEN3	0.83	4431	0.31	1419	0.39	4346	0.19	1556	7521	1.87	0.64	1992
Degen3Dual	0.38	2768	0.62	2866	0.22	2348	0.23	2716	0.89	4525	1.4	3422

Table A.5: NETLIB problems without bounds

Problem	CPLEX PS		CPLEX DS		GuRoBi PS		GuRoBi DS		MOSEK PS		MOSEK DS	
	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter
SCTAP3	0.02	648	0.62	373	0.02	632	0.02	620	0.02	557	0.02	399
SCTAP3Dual	0.04	1271	0.05	1169	0.02	1105	0	1112	0.04	1144	0.13	1729 + 1(PS)
MAROS-R7	2.85	4743	5.73	6633	0.81	2999	9.15	6103	3.39	4501	4.17+0.30	3626+290(PS)
MAROS-R7Dual	8.11	4762	4.31	4974	4.53	4907	4.37	6798	6728	15.87	4.37+0.02	6178+7(PS)
E226	0.03	357	0.03	314	0.01	343	0.02	438	0.03	589	0.03	374
E226duAL	0.03	433	0.02	315	0.02	392	0.01	283	0.03	454	0.03	317
FFFFF800	0.03	791	0.05	566	0.01	286	0.02	474	0.04	631	0.02	187+4 (PS)
FFFFF800Dual	0.06	753	0.04	622	0.04	695	0.04	648	0.07	1078	0.07	734+13(PS)
ISRAEL	0.01	188	0.01	128	0.01	159	0.02	123	0.01	288	0.02	152
ISRAELDual	0.02	261	0.01	158	0.01	242	0.01	189	0.01	254	0.02	178
LOTFI	0.01	187	0.01	190	0	192	0.01	241	0.01	242	0.01	203+17 (PS)
LOTFIDual	0.02	425	0.01	198	0.01	337	0.01	245	0.02	358	0.02	220
SC105	0	60	0	76	0	53	0	83	0	61	0	52
SC105Dual	0	130	0	95	0	112	0	104	0.01	128	0	55
SC205	0.01	130	0.01	144	0.01	134	0	179	0.01	185	0.01	204
SC205Dual	0.01	287	0.01	190	0.03	228	0.01	215	0.01	253	0.01	134
SC50A	0	30	0	33	0	32	0	34	0	34	0	34
SC50ADual	0	53	0	45	0	57	0	45	0	50	0	27
SC50B	0	33	0	31	0	35	0	35	0	34	0	31
SC50BDual	0	54	0	48	0	54	0	48	0	51	0	31
SCAGR25	0.04	520	0.03	485	0.01	413	0.02	648	0.03	455	0.02	465
SCAGR25Dual	0.03	477	0.03	632	0.01	635	0.02	707	0.03	537	0.04	343 +1(PS)
SCAGR7	0	84	0.01	163	0	98	0	203	0.01	136	0.01	138
SCAGR7Dual	0.01	177	0.01	145	0	198	0	176	0.01	144	0.04	343
SCFXM1	0.02	351	0.02	327	0.01	348	0.01	381	0.02	383	0.02	371
SCFXM1Dual	0.03	635	0.03	400	0.02	550	0.01	551	0.03	640	0.03	336
SCFXM2	0.04	794	0.04	734	0.06	797	0.02	780	0.05	826	0.05	768
SCFXM2Dual	0.07	1225	0.06	817	0.07	1374	0.04	1074	0.07	1041	0.07	722

Table A.6: NETLIB problems without bounds

Problem	CPLEX PS		CPLEX DS		GuRoBi PS		GuRoBi DS		MOSEK PS		MOSEK DS	
	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter
SCFXM3	0.05	1032	0.06	1056	0.03	1195	0.04	1149	0.07	1130	0.09	1185
SCFXM3Dual	0.11	1899	0.08	1240	0.1	1980	0.06	1618	0.1	1877	0.14	1136
SCORPION	0.01	274	0.01	267	0.01	235	0.01	322	0.01	194	0.01	266
SCORPIONDual	0.01	341	0.02	374	0.01	367	0.01	368	0.01	352	0.01	173
SCRS8	0.03	425	0.02	426	0.01	653	0.01	535	0.07	800	0.03	565
SCRS8Dual	0.04	599	0.04	603	0.02	811	0.03	657	0.03	584	0.05	413
SCSD1	0.01	303	0.01	171	0.05	162	0	95	0.01	283	0.01	97
SCSD1Dual	0.01	97	0.02	171	0.01	147	0.01	173	0.01	101	0.05	351
SCSD6	0.02	566	0.01	255	0.01	446	0.01	272	0.03	722	0.03	336
SCSD6Dual	0.04	327	0.03	363	0.02	359	0.03	406	0.08	694	0.08	482
SCSD8	0.05	1771	0.08	934	0.02	426	0.06	1106	0.23	2905	0.18	1155
SCSD8Dual	0.14	1032	0.16	1102	0.09	932	0.17	1125	0.33	2057	0.54	1542
SCTAP1	0.01	205	0.01	149	0	224	0	190	0.01	293	0.01	221
SCTAP1Dual	0.01	380	0.01	253	0	358	0.01	351	0.02	583	0.01	350
SCTAP2	0.03	593	0.02	278	0.01	472	0.01	340	0.02	353	0.02	330
SCTAP2Dual	0.04	1053	0.02	373	0.02	878	0.01	794	0.03	868	0.08	1219
SCTAP3	0.02	648	0.01	373	0.02	632	0.02	620	0.02	557	0.02	339
SCTAP3Dual	0.04	1271	0.05	1169	0.02	1105	0.03	1112	0.04	1144	0.13	1729+1(PS)
SHARE1B	0.01	137	0.01	120	0.01	158	0.01	149	0.01	209	0.01	0.01
SHARE1BDual	0.01	211	0.01	176	0.01	340	0.01	191	0.02	290	0.02	212
SHARE2B	0	92	0	81	0	114	0	86	0.01	114	0	112
SHARE2BDual	0	124	0.01	97	0	129	0	101	0	110	0	93
SHIP04L	0.01	386	0.01	408	0.01	458	0.01	554	0.01	303	0.02	436
SHIP04LDual	0.03	572	0.04	422	0.03	597	0.01	578	0.03	627	0.05	681
SHIP04S	0.01	250	0.01	408	0	331	0	409	0.01	210	0.01	445
SHIP04SDual	0.02	511	0.03	416	0.01	534	0.01	487	0.02	560	0.03	594
SHIP08L	0.03	843	0.03	717	0.02	923	0.02	745	0.01	284	0.03	699
SHIP08LDual	0.04	877	0.06	676	0.03	1091	0.05	1410	0.03	752	0.06	898
SHIP08S	0.02	439	0.03	677	0.01	488	0.01	509	0.03	601	0.03	731
SHIP08SDual	0.03	720	0.04	614	0.02	896	0.02	1019	0.06	995	0.14	1229
SHIP12L	0.04	1185	0.05	1212	0.01	1235	0.02	1406	0.04	810	0.05	1153
SHIP12LDual	0.05	1396	0.07	1016	0.03	1700	0.05	2037	0.09	1559	0.21	1851
STOCFOR1	0	35	0	67	0	28	0	79	0	45	0.01	40
STOCFOR1Dual	0	101	0	89	0	111	0	85	0.01	76	0.01	48
STOCFOR2	0.08	1321	0.13	1463	0.1	909	0.08	1641	0.15	1100	0.28	1444
STOCFOR2Dual	0.09	1792	0.16	1671	0.06	2148	0.08	2156	0.1	1050	0.22	1138
WOOD1P	0.06	592	0.04	160	0.04	396	0.03	157	0.09	578	0.27 + 0.01	504+27(PS)
WOOD1PDual	0.47	971	0.27	530	0.49	617	0.17	503	0.81	1296	0.42	454
WOODW	0.12	1711	0.36	1496	0.15	2555	0.42	2316	0.31	2034	0.35+0.04	1655+414(PS)
WOODWDual	1.93	4415	0.48	1758	0.95	2777	0.52	2085	2.99	4467	2.43+0.01	2537+5(PS)

A.4 NETLIB problems – Bounds

Table A.7: NETLIB problems with Bounds

Problem	CPLEX PS		CPLEX DS		GuRoBi PS		GuRoBi DS		MOSEK PS		MOSEK DS	
	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter
KB2	0	33	0	38	0	45	0	50	0	53	0	48
KB2ModPrimal	0	33	0.01	46	0	56	0	37	0	47	0	40
KB2Dual	0	61	0	56	0	63	0	59	0	101	0	38
RECIPE	0	15	0	45	0	44	0	47	0	31	0	43
RECIPEModPrimal	0.02	19	0	83	0	24	0	76	0	30	0	54
RECIPEDual	0	78	0	92	0	62	0	65	0	45	0	47
BORE3D	0.02	86	0.01	165	0.01	125	0.01	144	0.01	125	0.01	163
BORE3DModPrimal	0	91	0.01	184	0.01	146	0.01	165	0.01	118	0.01	155
BORE3DDual	0	270	0.01	346	0.01	233	0.01	261	0.01	288	0.02	99
CAPRI	0.02	285	0.01	219	0.01	399	0.01	186	0.01	343	0.01	297
CAPRIModPrimal	0.02	241	0.02	257	0.02	427	0.01	266	0.02	288	0.02	360
CAPRIDual	0	441	0.01	441	0.01	399	0.01	384	0.02	390	0.02	447
GROW7	0.02	215	0.02	464	0.02	332	0.02	344	0.03	480	0.06	600
GROW7ModPrimal	0	207	0.02	450	0.01	290	0.02	279	0.03	380	0.05	285
GROW7Dual	0.03	703	0.02	334	0.04	580	0.01	261	0.06	830	0.03	187
ETAMACRO	0.01	647	0.02	703	0.01	605	0.02	746	0.04	672	0.04	587
ETAMACROModPrimal	0.01	645	0.01	832	0.01	555	0.03	819	0.04	694	0.04	640+1(PS)
ETAMACRODual	0.01	962	0.01	766	0.03	853	0.03	746	0.09	1699	0.04	526
FINNIS	0.01	595	0	467	0.01	463	0.01	382	0.02	603	0.01	341
FINNISModprimal	0.01	645	0.01	529	0.01	488	0.01	397	0.02	571	0.01	349
FINNISDual	0.01	538	0.01	572	0.01	408	0.02	494	0.02	507	0.02	422
GFRD-PNC	0.01	860	0.02	558	0.02	669	0.02	575	0.02	531	0.03	563
GFRD-PNCModPrimal	0.01	909	0.02	625	0.01	672	0.03	636	0.03	527	0.03	633
GFRD-PNCDual	0.02	878	0.02	1047	0.03	629	0.03	967	0.03	660	0.04	495
STANDATA	0.01	146	0	119	0	96	0.01	138	0	54	0	60
STANDATAModPrimal	0.01	141	0	120	0	68	0.01	146	0	51	0.01	142
STANDATADual	0	236	0	205	0	71	0	139	0	74	0.01	121
STAIR	0.02	381	0.02	335	0.04	361	0.02	253	0.07	602	0.06	405
STAIRModPrimal	0.03	370	0.01	198	0.02	205	0.01	145	0.05	292	0.02	116
STAIRDual	0.03	529	0.03	795	0.03	565	0.04	507	0.06	551	0.06	381
TUFF	0	212	0.01	148	0.01	416	0	105	0.03	224	0.02	251+11(PS)
TUFFModPrimal	0	273	0.01	218	0.01	319	0.01	148	0.02	147	0.03	324+15(PS)
TUFFDual	0.01	360	0.01	330	0.03	570	0.02	382	0.03	488	0.05	397
STANDMPS	0.01	310	0.01	304	0	241	0.01	301	0.01	322	0.01	233
STANDMPSModPrimal	0	324	0.01	348	0	244	0.02	369	0.01	324	0.02	289
STANDMPSDual	0	404	0.01	416	0.01	433	0.01	341	0.04	535	0.04	487
GROW15	0.04	472	0.05	879	0.06	688	0.11	1189	0.1	1020	0.27	1483
GROW15ModPrimal	0.05	486	0.08	978	0.07	738	0.12	943	0.15	1244	0.17	591
GROW15Dual	0.19	2210	0.04	761	0.28	1919	0.06	633	0.12	1159	0.2	859

Table A.8: NETLIB problems with Bounds

Problem	CPLEX PS		CPLEX DS		GuRoBi PS		GuRoBi DS		MOSEK PS		MOSEK DS	
	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter
SHELL	0.01	718	0.01	662	0.01	459	0.01	445	0.01	409	0.02	477
SHELL ModPrimal	0.02	569	0.02	590	0.01	493	0.01	443	0.02	418	0.02	465
SHELL Dual	0.03	836	0.03	293	0.02	706	0.02	779	0.04	613	0.05	759+1(PS)
MODSZK1	0.06	1111	0.02	167	0.07	1104	0.01	101	0.09	922	0.06	660
MODSZK1Modprimal	0.05	1111	0	167	10	1459	0.01	109	0.11	1199	0.06	676
MODSZK1Dual	0.02	730	0.08	1574	0.02	685	0.1	1264	0.04	679	0.21	801
PILOT4	0.05	811	0.05	739	0.08	1125	0.09	923	0.13	1193	0.10+0.01	613+59(PS)
PILOT4ModPrimal	0.07	1056	0.09	1249	0.09	1219	0.14	1324	0.2	1772	0.06+0.17	461+1426(PS)
PILOT4Dual	0.16	1786	0.1	1299	0.15	1292	0.12	1208	0.43	3042	0.24	1117
PEROLD	0.19	2165	0.11	1209	0.31	2539	0.15	1392	0.4	2698	0.32+0.01	1440+66(PS)
PEROLDModprimal	0.21	2361	0.13	1556	0.34	2629	0.18	1546	0.39	2410	0.36+0.05	1537+279(PS)
PEROLDDual	0.18	1747	0.23	2376	0.29	2278	0.29	2237	0.81	4331	1.21	3687
GROW22	0.08	747	0.24	2317	0.12	973	0.29	2051	0.38	2964	0.63	2682
GROW22Modprimal	0.49	3776	0.49	3776	0.16	1201	0.99	4864	0.33	2225	0.29	768
GROW22Dual	2.3	13054	0.09	1190	0.72	3522	0.11	938	0.92	6148	0.39	1223
FIT1D	0.02	1607	0.01	100	0.02	1128	0.01	87	0.08	2056	0.01	70
FIT1DModprimal	0.06	1184	0.03	636	0.07	1063	0.05	634	0.21	2103	0.04	484
FIT1DDual	0.04	886	0.06	566	0.07	867	0.08	542	0.12	1099	0.11	491+3(PS)
GANGES	0.02	1451	0.02	865	0.02	638	0.01	373	0.05	987	0.06	1245
GANGESModPrimal	0.02	1273	0.03	1076	0.03	830	0.03	642	0.06	1112	0.08	1476
GANGESDual	0.07	2032	0.05	2239	0.07	1767	0.06	1578	0.08	1610	0.12	1263
FIT1P	0.03	447	0.05	611	0.04	670	0.06	552	0.07	671	0.08	495+3(PS)
FIT1PModprimal	0.04	814	0.05	586	0.05	722	0.08	762	0.1	1022	0.14	691+3(PS)
FIT1PDual	0.06	1438	0.1	1712	0.12	1398	0.08	1120	0.31	2039	0.07	962+1(PS)
MAROS	0.05	1522	0.07	1485	0.06	1431	0.11	1439	0.18	2020	0.18+0.01	1341+69(PS)
MAROSModprimal	0.05	1401	0.07	1423	0.07	1305	0.1	1308	0.15	1711	0.15+0.01	1255+74(PS)
MOROSDual	0.09	2033	0.06	1402	0.2	2391	0.13	1471	0.31	2259	0.27	1138
SIERRA	0.01	427	0.01	550	0.01	415	0.01	545	0.02	484	0.03	572
SIERRAModPrimal	0.01	377	0.01	602	0.01	478	0.02	577	0.03	507	0.04	582
SierraDual	0.01	754	0.02	1076	0.01	651	0.02	900	0.02	655	0.04+0.01	683+77(PS)
PILOTNOV	0.15	1634	0.19	1705	0.23	2237	0.2	1795	0.33	1516	0.3	1251
PILOTNOVModprimal	0.14	1631	0.21	1712	0.49	3250	0.22	1471	0.37	1508	0.37	1327
PILOTNOVDual	0.17	1732	0.29	2489	0.4	2353	0.4	2168	0.48	2108	1.03	2466
CZPROB	0.03	1907	0.06	1328	0.06	1074	0.06	1097	0.11	2032	0.06	1070
CZPROBModPrimal	0.03	2014	0.07	1403	0.05	1015	0.06	1110	0.11	1927	0.06	1098
CZPROBDual	0.08	1587	0.1	1876	0.11	1743	0.13	1500	0.22	1721	0.35	2792
CYCLE	0.03	1053	0.01	233	0.05	1159	0.02	138	0.21	2165	0.06+0.01	620+126(PS)
CYCLEModprimal	0.03	1116	0.03	452	0.04	974	0.02	172	0.15	1774	0.06+0.01	563+1376(PS)
CYCLEDual	0.07	1972	0.08	1957	0.5	3469	0.16	1383	0.65	3201	0.56	1423

Table A.9: NETLIB problems with Bounds

Problem	CPLEX PS		CPLEX DS		GuRoBi PS		GuRoBi DS		MOSEK PS		MOSEK DS	
	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter
D6CUBE	1.07	20092	0.1	552	3.08	24358	0.12	442	8.16	32149	0.24+0.01	450+6
D6CUBEModPrimal	0.97	19715	0.09	574	2.36	24490	0.12	442	7.05	29829	0.21	410+3
D6CUBEDual	0.95	2979	3.95	13654	0.78	1540	9.45	14852	1.83	2959	10.62	8760
GREENBEA	0.61	6854	0.49	4350	1.27	6436	1.71	5067	3.21	14101	0.78+0.01	3183+2(PS)
GREENBEAModprimal	0.52	6456	0.45	4422	1.37	6865	1.46	4676	3.02	12712	0.99+0.01	3623+5(PS)
GREENBEADual	2.25	10198	2.06	10279	4.42	10774	3.73	10433	5.23	10296	7.26	8759
GREENBEB	0.63	6312	0.79	5631	0.95	5645	2.68	7044	3.58	11355	1.8	5241
GREENBEBModPrimal	0.5	5786	0.81	5717	1.03	5441	2.37	6612	3.1	10674	1.83	5354
GREENBEBDual	2.05	9119	0.97	6612	4.14	10585	1.66	6045	14.17	10846	3.78	5046
PILOT	1.55	5422	1.3	4271	3.32	5841	2.72	5071	7.5	10003	2.97+0.01	3382+11(PS)
PILOTmodPrimal	1.84	5827	1.41	4313	3.7	5886	2.7	4431	7.65	9263	3.68+0.02	3873+12(PS)
PILOTDual	1.87	4673	2.49	5272	3.61	6072	3.97	5057	12755	9.56	7.46	5124
80BAU3B	0.23	11594	0.14	4557	0.56	7652	0.28	4074	0.78	10166	0.33	3467+1(PS)
80BAU3BModprimal	0.3	11751	0.57	9147	0.83	8249	0.35	5806	1.1	11533	0.44	4258+1(PS)
80BAU3BDual	0.94	10328	0.6	6753	0.95	6956	1.05	6318	1.33	6069	1.75+0.01	5702+1(PS)
FIT2D	0.34	27853	0.09	268	0.46	13938	0.28	189	50104	1.88	0.17+0.02	150+30(PS)
FIT2Dmodprimal	5.4	19755	3.18	9608	13.23	18090	8.4	11432	40260	24.17	0.96+0.04	4810+22(PS)
FIT2DDual	3.2	11605	10.18	13920	6.74	12345	8.2	6171	5246	7.25	15.39+0.03	5738+8(PS)
PILOT87	4.46	7512	8.91	11474	7.93	7316	13.53	13293	21	19957	20.27+0.96	10120+722(PS)
PILOT87ModPrimal	5.08	8100	8.97	11032	9.27	7675	15.06	11254	27.51	21956	21.14+1.02	9474+757(PS)
Pilot87DUal	17.56	17712	6.48	7882	20.41	12580	13.03	6920	39.43	28178	17.57	6241
PILOT.JA	0.39	3090	0.25	2226	0.71	3939	0.34	2114	0.8	3643	0.42+0.02	1599+71(PS)
PILOT.JAMoDPrimal	0.56	3880	0.33	2076	0.91	4898	0.38	2053	0.96	4123	0.52+0.01	1802+9(PS)
PILOT.JADual	0.31	2246	0.38	2891	0.69	3085	0.64	3120	1.22	4884	1.9	4042+5(PS)

A.5 MOSEK Solve parameters

Mosek parameter to solve only using primal Simplex

```
BEGIN MOSEK
MSK_IPAR_PRESOLVE_USE MSK_PRESOLVE_MODE_OFF
MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_PRIMAL_SIMPLEX
MSK_IPAR_SIM_SOLVE_FORM  MSK_SOLVE_PRIMAL
```

Mosek parameter to solve only using Dual Simplex

```
BEGIN MOSEK
MSK_IPAR_PRESOLVE_USE MSK_PRESOLVE_MODE_OFF
MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_DUAL_SIMPLEX
MSK_IPAR_SIM_SOLVE_FORM  MSK_SOLVE_PRIMAL
```

Mosek parameter to solve using IPM (solve the primal form)

```
BEGIN MOSEK
MSK_IPAR_PRESOLVE_USE MSK_PRESOLVE_MODE_OFF
MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_INTPNT
MSK_IPAR_INTPNT_SOLVE_FORM  MSK_SOLVE_PRIMAL
END MOSEK
```

MOSEK parameter to solve using IPM (take Dual through software)

```
BEGIN MOSEK
MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_INTPNT
MSK_IPAR_INTPNT_SOLVE_FORM  MSK_SOLVE_DUAL
END MOSEK
```

Bibliography

[1] E. D. Andersen, C. Roos, T. Terlaky, T. Trafalis, and J. P. Warners. The use of low-rank updates in interior-point methods. In Y. Yuan, editor, *Numerical linear algebra and optimization : proceedings of the 2003' International Conference on Numerical Organization and Numerical Linear Algebra*, pages 3–14. Chinese Academy of Sciences, 2004.

[2] E. D. Anderson, J. Gondzio, C. Mészáros, and X. Xu. Implementation of interior-point methods for large scale linear programs. In T. Terlaky, editor, *Interior Point Methods of Mathematical Programming*, pages 189–252. Springer US, 1996.

[3] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 1st edition, 1997.

[4] R. E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50(1):3–15, 2002.

[5] R. E. Bixby. A Brief History of Linear and Mixed-Integer Programming Computation. *Documenta Mathematica*, pages 107–121, 2012.

[6] Y. Y. Chang. Least index resolution of degeneracy in linear complementarity problems. *Technical Report 79–14, Systems Optimization Laboratory, Stanford University*, 1979.

[7] G. B. Dantzig. Programming in a linear structure. *Econometrica*, 17:73–74, 1949.

[8] G. B. Dantzig. Reminiscences about the origins of linear programming. *Operations Research Letters*, 1(2), 1982.

[9] A. V. Fiacco and G. P. McCormick. Programming under nonlinear constraints by unconstrained minimization: A primal-dual method. *The Research Analysis Corporation*, RAC-TP-96, 1963.

[10] J. J. Forrest and D. Goldfarb. Steepest-edge simplex algorithms for linear programming. *Mathematical Programming*, 57(3):341–374, 1992.

[11] D. Gale, H. W. Kuhn, and A. W. Tucker. Linear programming and the theory of games. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 317–329. Wiley New York, 1951.

[12] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright. On projected newton barrier methods for linear programming and an equivalence to karmarkar's projective method. *Mathematical Programming*, 36(2):183–209, 1986.

[13] A. J. Goldman and A. W. Tucker. Theory of linear programming. In H. W. Kuhn and A. W. Tucker, editors. *Linear Inequalities and related Systems*, RAC-TP-96:53–97, 1956.

[14] L. V. Kantorovich. *Mathematical Methods in the Organization and Planning of Production.* Leningrad State University, 1939. Translated in Management Science 6(4):336-422, 1960.

[15] N. Karmakar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[16] L. G. Khachiyan. A polynomial algorithm in linear programming. *Proceedings of the USSR Academy of Sciences*, 244:1093–1096, 1979. Translated in Soviet Mathematics Doklady 20:191-194, 1979.

[17] V. Klee and G. J. Minty. How good is the simplex algorithm? In R. G. Jeroslow, editor, *Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin)*, pages 159–175. Academic Press, New York, 1972.

[18] C. E. Lemke. The dual method of solving the linear programming problem. *Naval Research Logistics Quarterly*, 1(1):36–47, 1954.

[19] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra and Its Applications*, 152:191 – 222, 1991.

[20] I. Maros and C. Mészáros. The role of the augmented system in interior point methods. *European Journal of Operational Research*, 107(3):720 – 736, 1998.

[21] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[22] W. Orchard-Hayes. History of the development of LP solvers. *Interfaces*, 20(4):61–73, 1990.

[23] T. Terlaky. A convergent criss-cross method. *Optimization*, 16(5):683–690, 1985.

[24] T. Terlaky and S. Zhang. Pivot rules for linear programming: A survey on recent theoretical developments. *Annals of Operations Research*, 46(1):203–233, 1993.

[25] A. N. Tolstoi. Methods of finding the minimal total kilometrage in cargo transportation planning in space. *Transportation Planning*, 1:23–55, 1930.

[26] Z. Wang. A conformal elimination free algorithm for oriented matroid programming. *Chinese Annals of Mathematics*, 8(B1):120–125, 1987.

[27] Y. Ye, M. J. Todd, and S. Mizuno. An $O(\sqrt{n}L)$ -iteration homogeneous and Self-dual linear programming algorithm. *Mathematics of Operations Research*, 19(1):53–67, 1994.

Biography

Naga Venkata Chaitanya Gudapati was born in coastal town of Visakhaptnam, Andhra Pradesh, India in 1992. He obtained his Bachelors in Engineering from Indian Institute of technology, Madras in 2013 majoring in Chemical Engineering. In 2015, he joined the Industrial and Systems Engineering department at Lehigh University in 2015. At Lehigh University, he worked under the guidance of Tamás Terlaky on the topic *Duality and a Closer Look at Implementation of Linear Optimization Algorithms*.

He also worked on the *Inmate Assignment Project* along with Prof. Tamás Terlaky, Prof. Louis Plebani, Prof. George Wilson and Ph.D. student Mohammad Shahabsafa.

His hobbies include speed cubing, orienteering and hiking.