วิธีซิมเพล็กซ์แบบไร้ตัวแปรเทียมสำหรับตัวแบบกำหนดการเชิงเส้นหลักและคู่ควบ

นางสาวเอื้ออารี บุญเพิ่ม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต

สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2556

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

บทคัดย่อและแฟ้มข้อมูลฉบับเต็มของวิทยานิพนธ์ตั้งแต่ปีการศึกษา 2554 ที่ให้บริการในคลังปัญญาจุฬาฯ (CUIR)

เป็นแฟ้มข้อมูลของนิสิตเจ้าของวิทยานิพนธ์ที่ส่งผ่านทางบัณฑิตวิทยาลัย

The abstract and full text of theses from the academic year 2011 in Chulalongkorn University Intellectual Repository(CUIR)

are the thesis authors' files submitted through the Graduate School.

ARTIFICIAL-VARIABLE-FREE SIMPLEX METHOD FOR PRIMAL AND DUAL LINEAR PROGRAMMING MODELS

Miss Aua-aree Boonperm

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Applied Mathematics and Computational Science
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2013

Copyright of Chulalongkorn University

Thesis Title	ARTIFICIAL-VARIABLE-FREE SIMPLEX METHOD FOR PRIMAL AND DUAL LINEAR PROGRAMMING MODELS
By	Miss Aua-aree Boonperm
Field of Study	Applied Mathematics and Computational Science
Thesis Advisor	Assistant Professor Krung Sinapiromsaran, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

.....Dean of the Faculty of Science
(Professor Supot Hannongbua, Dr.rer.nat.)

THESIS COMMITTEE

.....Chairman
(Assistant Professor Khamron Mekchay, Ph.D.)

.....Thesis Advisor
(Assistant Professor Krung Sinapiromsaran, Ph.D.)

.....Examiner
(Associate Professor Pornchai Satravaha, Ph.D.)

.....Examiner
(Phantipa Thipwiwatpotjana, Ph.D.)

.....External Examiner
(Assistant Professor Titirut Thipbharos, Ph.D.)

เอื้ออารี บุญเพิ่ม : วิธีซิมเพล็กซ์แบบไร้ตัวแปรเทียมสำหรับตัวแบบกำหนดการเชิงเส้นหลักและคู่ควบ. (ARTIFICIAL-VARIABLE-FREE SIMPLEX METHOD FOR PRIMAL AND DUAL LINEAR PROGRAMMING MODELS) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ.ดร.กรุง สินอภิรมย์สราญ, 136 หน้า.

การแก้ปัญหากำหนดการเชิงเส้นโดยใช้ขั้นตอนวิธีซิมเพล็กซ์ด้วยการเพิ่มตัวแปรเทียมเป็นการเพิ่มปริภูมิการค้นหาให้มีขนาดใหญ่ขึ้น วิทยานิพนธ์นี้นำเสนอเทคนิคการผ่อนปรนเงื่อนไขบังคับที่ไม่ใช่มุมแหลมซึ่งไม่เพียงแต่กำจัดความต้องการตัวแปรเทียมเท่านั้น ยังลดเวลาเริ่มต้นของการแก้ปัญหาผ่อนปรน การรับประกันผลเฉลยเหมาะที่สุดหรือไม่มีผลเฉลยหรือไม่มีขอบเขตของปัญหากำหนดการเชิงเส้น ทำได้โดยขั้นตอนวิธีจะนำเงื่อนไขบังคับที่ไม่ใช่มุมแหลมกลับเข้ามารวมในปัญหาผ่อนปรน ซึ่งผลลัพธ์ของขั้นตอนวิธีนี้ดีกว่าขั้นตอนวิธีซิมเพล็กซ์แบบดั้งเดิมซึ่งใช้ตัวแปรเทียม เมื่อปัญหากำหนดการเชิงเส้นที่ปัญหาผ่อนปรนมีคำตอบที่เหมาะที่สุดก่อนการนำเงื่อนไขที่ไม่ใช่มุมแหลมเข้ามา

ภาควิชา	คณิตศาสตร์ และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต	.....
		ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก	.....
สาขาวิชา	คณิตศาสตร์ประยุกต์ และวิทยาการคณนา		
ปีการศึกษา	2556		

5273908523 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE KEYWORDS : ARTIFICIAL-FREE / GRADIENT VECTOR / LINEAR PROGRAMMING / NON-ACUTE CONSTRAINT RELAXATION

AUA-AREE BOONPERM : ARTIFICIAL-VARIABLE-FREE SIMPLEX METHOD FOR PRIMAL AND DUAL LINEAR PROGRAMMING MODELS. ADVISOR : ASST. PROF. KRUNG SINAPIROMSARAN, Ph.D., 136 pp.

Solving a general linear programming problem using the simplex algorithm relies on introducing artificial variables that deals with a large search space. This dissertation presents the non-acute constraint relaxation technique that not only eliminates the need for artificial variables but also reduces the start-up time to solve the initial relaxation problem. To guarantee the optimal solution or infeasibility or unboundedness of a linear programming problem, the algorithm reinserts the non-acute constraints back to the relaxation problem. The results of this algorithm are superior than the original simplex algorithm with artificial variables for a linear programming problem which the relaxed problem obtains the optimal solution before the the reinsertion of non-acute constraints.

Department : Mathematics and Computer Science Student's Signature : .....
Field of Study : Applied Mathematics and Computational Science Advisor's Signature : .....
Academic Year : 2013

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my advisor, Assistant Professor Dr. Krung Sinapiromsaran for his invaluable suggestion, understanding and support throughout this dissertation. Without his guidance and continued help, this dissertation would not have been possible.

I also would like to thank to my thesis committees, Assistant Professor Dr. Khamron Mekchay, Associate Professor Dr. Pornchai Satravaha and Dr. Phantipa Thipwiwatpotjana, and my thesis external examiner, Assistant Professor Dr. Titirut Thipbharos who is the lecturer at Faculty of Applied Science, Dhurakij Pundit University for the invaluable comments and attitudes which have significantly improved this dissertation. Moreover, I would like to thank all lecturers who instructed and taught me valuable knowledge.

Additionally, I also most gratefully thanks to the Development and Promotion of Science and Technology talents project (DPST) under the Institute for the Promotion of Teaching Science and Technology, for financial support throughout my Ph.D. study.

Finally, I most gratefully acknowledge my parents, my family, Roungroj Poomduang, Roungroj's family and my friends for all their support throughout the period of this dissertation, and I would like to thank those whose names are not mentioned here but greatly inspired and encouraged us until this dissertation comes to the end.

CONTENTS

page

page

page

LIST OF TABLES

TABLE **x**

LIST OF FIGURES

FIGURE **xii**

CHAPTER I
INTRODUCTION

1.1 Introduction to Linear Programming

Linear programming (LP) describes the mathematical programming type dealing with the optimal value of a linear objective function which is defined as minimizing or maximizing under linear equality or inequality constraints. Some real world problems such as an industrial production, a transportation problem, a production scheduling problem, an assignment problem, etc., can be constructed as a linear programming model searching for the optimal solution. Consider a general linear programming model:

$$\begin{array}{lccccccc} \text{Maximize} & c_1x_1 & + & c_2x_2 & +\cdots+ & c_nx_n & & \\ \text{subject to} & a_{11}x_1 & + & a_{12}x_2 & +\cdots+ & a_{1n}x_n & \leq & b_1 \\ & a_{21}x_1 & + & a_{22}x_2 & +\cdots+ & a_{2n}x_n & \leq & b_2 \\ & \vdots & & \vdots & +\cdots+ & \vdots & & \vdots \\ & a_{m1}x_1 & + & a_{m2}x_2 & +\cdots+ & a_{mn}x_n & \leq & b_m \\ & x_1, & & x_2, & \ldots, & x_n & \geq & 0. \end{array} \tag{1.1}$$

Denote the following column vectors $\mathbf{c}$ and $\mathbf{x}$ of size n, $\mathbf{b}$ of size m, and the $m \times n$ matrix $\mathbf{A}$:

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, \ \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \ \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

$\mathbf{A}_{i:} = [a_{i1}, a_{i2}, \ldots, a_{in}]$ is a coefficient vector or a gradient vector of the constraint i. The problem can be written in the matrix form:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{1.2}$$

In Problem 1.2, the function being maximized is called *the objective function* and $\mathbf{c}$ is a vector of coefficients of the objective function or, in other word, $\mathbf{c}$ is a gradient vector of the objective function. $\mathbf{x}$ is a vector of variables called *the decision variables*, $\mathbf{A}$ is a coefficient matrix of constraints, $\mathbf{b}$ is a vector of parameters called the *right-hand side* vector, n is the number of decision variables and m is the number of constraints.

A vector $\mathbf{x}$ is said to be *a feasible point* or *a feasible solution* if it satisfies all constraints, and a set of all feasible points is called *the feasible region.* The solution of a linear programming problem depends on the problem structure which can be summarized (for a maximization problem) below:

(i) *Optimal solution.* An optimal solution to a linear programming problem exists which provides the greatest objective value.

(ii) *Unbounded optimal solution.* A linear programming problem is unbounded when the optimal objective value is unbounded (with value ∞) and no optimal solution exists.

(iii) *Empty feasible region.* A linear programming problem is *infeasible* or has an *empty feasible region* when the system of constraints defining the feasible region is inconsistent.

Standard and Canonical Formats

There are two important representations of a linear programming problem for different uses: the standard form and the canonical form. The standard form of a linear programming will be useful for the simplex algorithm which is designed to work on the standard form while the canonical form is useful for duality relationship. These forms (in matrix form) are summarized in Table 1.1

Table 1.1: Standard and Canonical Forms

	Maximization Problem	Minimization Problem
Standard Form	Maximize $\mathbf{c^T x}$ subject to $\mathbf{Ax} = \mathbf{b}$ $\mathbf{x} \geq \mathbf{0}$	Minimize $\mathbf{c^T x}$ subject to $\mathbf{Ax} = \mathbf{b}$ $\mathbf{x} \geq \mathbf{0}$
Canonical Form	Maximize $\mathbf{c^T x}$ subject to $\mathbf{Ax} \leq \mathbf{b}$ $\mathbf{x} \geq \mathbf{0}$	Minimize $\mathbf{c^T x}$ subject to $\mathbf{Ax} \geq \mathbf{b}$ $\mathbf{x} \geq \mathbf{0}$

Either all constraints of the maximization or minimization problem in the standard form are of "equal to" types while all constraints of the maximization problem in the canonical form are of "less than or equal to" types, and all constraints of the minimization problem in the canonical form are of "greater than or equal to" types. The decision variables of all forms are nonnegative.

Problem Manipulation

Many linear programming problems in term of maximization or minimization and variables may be nonnegative, unrestricted in sign or bounded which may not match the standard or the canonical form. Since some algorithms deal with a specific form of a linear programming problem, a problem must be manipulated from one form to fit the required form.

- **Maximization and Minimization problems**: Converting a maximization problem to a minimization problem and conversely as:

$$\text{maximize } \mathbf{c^T x} = -\text{minimize } -\mathbf{c^T x}.$$

- **Inequality Constraints**: Consider a constraint given by $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$. This constraint can be transformed to an equality constraint as follow:

$$\sum_{j=1}^{n} a_{ij}x_j + s_i = b_i,$$

where $s_i \geq 0$ is called a **slack variable**.
While a constraint given by $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ can be converted to an equality constraint as follows:

$$\sum_{j=1}^{n} a_{ij}x_j - s_i = b_i,$$

where $s_i \geq 0$ is called a **surplus variable**.

- **Equality Constraints**: If a constraint has an equation form, i.e., $\sum_{j=1}^{n} a_{ij}x_j = b_i$, it can be transformed into two inequality constraints as follows:

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \quad \text{and} \quad \sum_{j=1}^{n} a_{ij}x_j \geq b_i$$

- **Nonnegativity of the Variables**: If the variable x_j can be positive, zero or negative, called **unrestricted** in sign, it can be converted to two new nonnegative variables as follows:

$$x_j = x'_j - x''_j \quad \text{where} \quad x'_j \geq 0, x''_j \geq 0.$$

Next, if the variable $x_j \geq l_j$, the new variable $x'_j = x_j - l_j$ is mathematically nonnegative. Similarly, if $x_j \leq u_j$, the new variable $x'_j = u_j - x_j$ is also nonnegative.

The problem manipulation of all cases are summarized in Table 1.2.

Methods for Solving LP

There are many methods to solve a linear programming problem such as the graphical method, the simplex method, the interior point method, etc.

The graphical method is a method to solve an LP problem by drawing a feasible region corresponding to all constraints. Then the hyperplane of the objective function is moved parallel in the direction that the objective value increases for the maximization problem (or toward the direction that the objective value decreases for the minimization problem) maintaining within the feasible region. The last

Table 1.2: Problem manipulations

	Original form	**Equivalent form**
objective function	maximize $\mathbf{c^T x}$	$-$minimize $-\mathbf{c^T x}$
constraints	$\sum_{j=1}^{n} a_{ij}x_j \leq b_i$	$\sum_{j=1}^{n} a_{ij}x_j + s_i = b_i, s_i \geq 0$
	$\sum_{j=1}^{n} a_{ij}x_j \geq b_i$	$\sum_{j=1}^{n} a_{ij}x_j - s_i = b_i, s_i \geq 0$
	$\sum_{j=1}^{n} a_{ij}x_j = b_i$	$\sum_{j=1}^{n} a_{ij}x_j \geq b_i, \sum_{j=1}^{n} a_{ij}x_j \leq b_i$
variables	x_j unrestricted	$x_j = x'_j - x''_j, x'_j \geq 0, x''_j \geq 0$
	$x_j \geq l_j$	$x'_j = x_j - l_j$
	$x_j \leq u_j$	$x'_j = u_j - x_j$

feasible point that the plane touches is the optimal solution. Since the graphical method requires the sketch of the feasible region be, it is suitable for two or three dimensional problems.

The simplex method published by George B. Dantzig [1] in 1947 is quite efficient in practice and has been popularly used for solving a linear programming problem. The simplex algorithm starts from the origin point if it is a feasible point. Otherwise, artificial variables will be added to obtain the initial feasible solution on a larger search space with $\mathbf{x} = \mathbf{0}$. Then, it moves from one corner point to an adjacent corner point along the boundary of the feasible region until the optimal solution is reached. However, in 1972, Klee and Minty [2] constructed a collection of linear programming problems which the simplex algorithm using Dantzig's rule took an exponential number of iterations with respect to the problem size. Karmarkar [3] responded to this in 1984 by proposing a new polynomial-time algorithm called the interior point method for a linear programming problem.

The interior point method creates the sequence of interior points which converges to the optimal solution along the improving direction. The number of iterations of the interior point method requires less than 100 iterations which does not increase with the the problem size [4, 5]. However, one iteration with the interior point method is much more complicated than one iteration with the simplex

method. In order to compute a direction at each iteration, $\mathbf{AD^2A^T}$ matrix must be formed where $\mathbf{D^2}$ is a diagonal matrix which changes at each iteration and the system is solved to find the direction. Computing $\mathbf{AD^2A^T}$ for dense columns causes it to converge slowly. So the density of this matrix is the main weakness point of the interior point method [4, 6, 7].

1.2 Motivation

"*Is there a polynomial time algorithm over the real numbers which decides the feasibility of the linear system of inequalities* $\mathbf{Ax} \geq \mathbf{b}$?" [8, 9] is still an open problem. This is one version of finding a feasible solution of an optimization problem of a linear programming problem. Moreover, improving the simplex method is still important and suitable for solving integer programming problems [21, 23]. Many researchers proposed the improved simplex algorithm by suggesting a new pivot rule [18, 23], a new initial solution [21, 22], a new method without artificial variables [10, 14, 11, 12, 15, 16, 17], etc. Some proposed their common technique to improve the simplex method, that is, constraints making small angles between it
cl

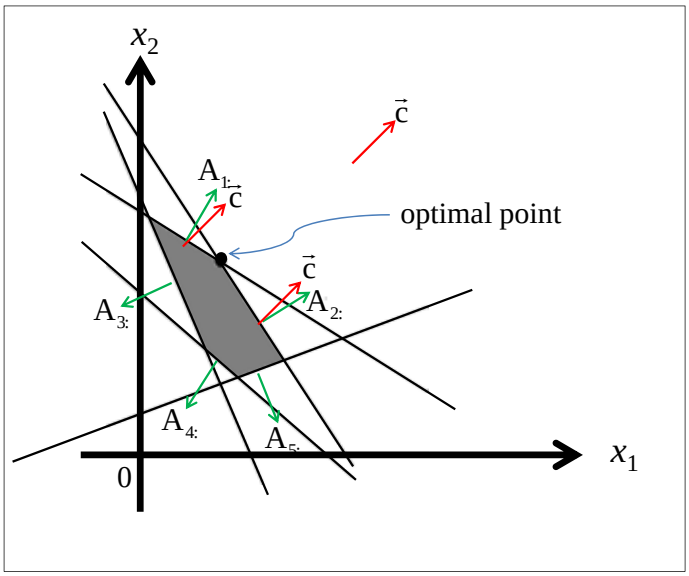


Figure 1.1: Common technique

In 2005, Junior and Lins [21] suggested the new initial basis formed by m variables associated with m constraints making the smallest angles between its gradient vector to the gradient vector of the objective function in the dual problem, respectively. The computational results of their algorithm were shown to be superior for small problems. However, in 2007, Hu [22] showed that Junior and Lins' algorithm could not start if this new initial basis is singular.

Then, in 2009, Yeh and Corley [23] proposed the new pivot rule. The variable which associated with the constraint making the smallest angle between its gradient vector to the gradient vector of the objective function in the dual problem was chosen to be an entering variable. However, computational results showed that their algorithm could improve the average number of iterations only 2.6597% with respect to the simplex algorithm.

Recently, in 2011, Wei Li and Haohao Li [24] proved that the optimal solution of 2-dimensional linear programming problem with no redundant constraint belongs to the constraint making the acute angle between its gradient vector to the gradient vector of the objective function.

Although a constraint making the acute angle technique is used to improve the simplex algorithm, one technique that researchers interested is eliminating the use of artificial variables. From Figure 1.1, artificial variables are needed to start the simplex algorithm at $\mathbf{x} = \mathbf{0}$. There are two well-known methods to deal with the artificial variable technique, i.e., the Big-M method and the Two-Phase method. If artificial variables are introduced, the primal space expands. Solving the LP problem without using artificial variable may reduce iterations and time.

The algorithm without artificial variables was first proposed by Zionts [10], in 1969, called the criss-cross algorithm. The criss-cross algorithm needed not maintain feasibility and no artificial variable required. But the criss-cross algorithm did not have the polynomial time-complexity and it slowly converges to the optimal solution.

In 1997, Arsham [11, 12] proposed the algorithm without using artificial variables. However, in 1998, Enge and Huhn [13] presented a counterexample, in

which Arsham's algorithm declared the infeasibility of a feasible linear programming problem.

In 2000, Pan [14] proposed another algorithm for solving a linear programming problem without introducing artificial variables. If the initial basis was neither primal nor dual feasible, then coefficients of objective function in primal will be perturbed for the dual feasibility and the dual simplex method will be used. The computational results were shown to be superior for small problems.

Then, in 2006, Arsham [15, 16] presented the improved algorithm by relaxing some constraints which the origin point does not satisfy. Therefore, the algorithm can start at the origin point without using artificial variables. After the solution is found, relaxed constraints are added for checking with the optimal point. The performance of his algorithm is shown by some examples. In that year, Corley, Rosenberge, Yeh and Sung [17] proposed the similar algorithm with Arsham. They solved a sequence of relaxed linear programming problems until the optimal solution of the original problem was found. The relaxed problem consisted of an original objective function subject to a single constraint which makes a largest cosine angle with the gradient vector of the objective function. At each subsequent iteration of the algorithm, the constraint which had the new maximum cosine angle with the gradient vector to the objective function among those constraints would be added and the dual simplex method were applied. But their research lacked a computational result and all linear programming problems were feasible and bounded.

Note that some constraints making acute angles may form an extreme point close to the optimal solution. In this dissertation, we classify constraints to two types.

- **The acute constraint** is the constraint which makes an acute angle between its gradient vector to the gradient vector of the objective function.

- **The non-acute constraint** is the constraint which makes an obtuse or orthogonal angle between its gradient vector to the gradient vector of the objective function.

We propose to remove non-acute constraints temporary and solve the problem called *the relaxed problem*. After the relaxed problem is solved, the algorithm reinserts all non-acute constraints for an optimal case and it reinserts one non-acute constraints at a time for an unbounded case. If the optimal solution of the original LP problem is achieved from the relaxed problem, we may reduce the solution time to solve the whole problem since non-acute constraints will satisfy this optim

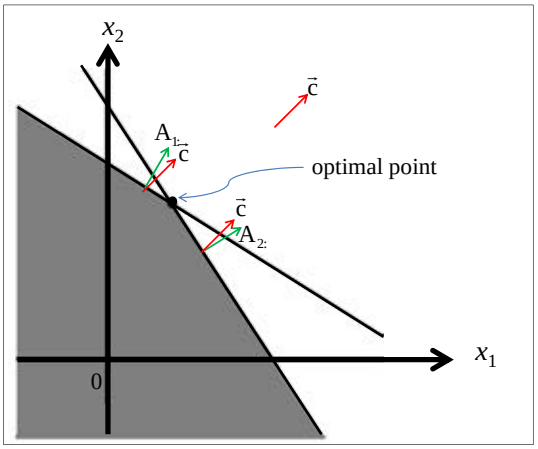
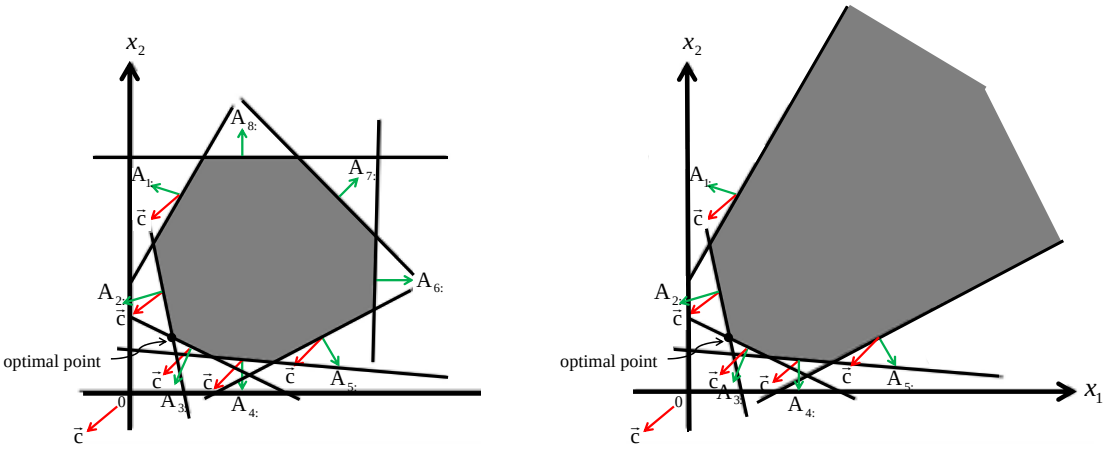


Figure 1.2: A feasible region of the relaxed problem in $\mathbb{R}^2$

In Figure 1.2, non-acute constraints which were not related to the optimal solution were relaxed. The simplex method can start at the origin point and find the optimal solution easily without using artificial variable. However, if the origin point of the relaxed problem is infeasible as in Figure 1.3b, the simplex method still needed to introduce artificial variables to start the algorithm. We can prove that the relaxed problem has a feasible solution, and then an artificial variable is not needed to start the simplex method.

Our algorithm starts by separating constraints into two collections: the collection of acute constraints and the collection of non-acute constraints. The original objective function with the collection of acute constraints, called the non-acute constraint relaxation problem, will be solved first. We can prove that the feasible region of this collection is nonempty and the feasible point can be identified by the mathematical formula. So the algorithm starts with an interior feasible solution



(a) The original problem. (b) The relaxed problem.

Figure 1.3: Example of the two-phase method

without artificial variables. Then, it transforms the interior feasible solution to the basic feasible solution of the equivalent linear programming problem and is solved using the simplex algorithm. In case that the relaxed problem reaches the optimal solution, the algorithm reinserts non-acute constraints back to the relaxed problem and uses the dual simplex method to identify the optimal solution or the infeasibility or the unbounded optimal solution. For the case of unbounded optimal solution of the relaxed problem, a single non-acute constraint is inserted one constraint at a time. Our algorithm is named *the **S**implex method based on the **N**on-**A**cute constraint **R**elaxation* or *SNAR*.

The aim of this dissertation is to reduce the number of iterations or time to solve the LP problem with respect to the simplex method using Dantzig's rule. The performance of our algorithms are shown by computational results which test with randomly generated linear programming problems.

1.3 Overview of the Dissertation

In chapter 2, we describe definitions and theorems used in our algorithm such as a direction, an extreme point, duality, sensitivity analysis, etc. Moreover, the simplex algorithm including the artificial variables techniques, two-phase method and big-M method, are in chapter 2. Additionally, related works which consist

of artificial variable techniques, the artificial-variable-free techniques will be described in detail. Some artificial-free variable techniques will be compared with our algorithm. In chapter 3, the main concept of our algorithm and theorems will be described. Next, the experimental design and computational results are shown and discussed in chapter 4. Finally, in the last chapter, our research and results are analysed and concluded.

CHAPTER II
LITERATURE REVIEWS

Some definitions and theorems of a linear programming problem that will be used for our dissertation are described, and related works are summarized and discussed in this chapter.

2.1 Definitions and Theorems of Linear Programming

The following definitions are useful for the simplex method, our algorithm and sensitivity analysis.

Definition 2.1. (**Hyperplane**): A *hyperplane* H in $\mathbb{R}^n$ is a set of the form $\{\mathbf{x} : \mathbf{p} \cdot \mathbf{x} = k\}$ where $\mathbf{p}$ is a nonzero vector in $\mathbb{R}^n$ and k is a scalar.

Here, $\mathbf{p}$ is called the *normal* or the *gradient* to the hyperplane.

Definition 2.2. (**Half-Spaces**): A *half-space* is a collection of points of the form $\{\mathbf{x} : \mathbf{p} \cdot \mathbf{x} \leq k\}$ or $\{\mathbf{x} : \mathbf{p} \cdot \mathbf{x} \geq k\}$.

Definition 2.3. (**Polyhedral Sets**): A *polyhedral set* is the intersection of a finite number of half-spaces.

Definition 2.4. (**Convex Sets**): A set $\mathbf{X}$ in $\mathbb{R}^n$ is called a *convex set* if given any two points $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\mathbf{X}$, then $\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in \mathbf{X}$ for each $\lambda \in [0, 1]$.

Any point of the form $\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in \mathbf{X}$ where $\lambda \in [0, 1]$ is called a *convex combination* of $\mathbf{x}_1$ and $\mathbf{x}_2$. If $\lambda \in (0, 1)$, then the convex combination is called *strict.*

Definition 2.5. (**Rays and Directions**): A *ray* is a collection of points of the form $\{\mathbf{x}_0 + \lambda\mathbf{d} : \lambda \geq 0\}$ where $\mathbf{d}$ is a nonzero vector. $\mathbf{x}_0$ is called the *vertex* of the ray, and $\mathbf{d}$ is the *direction of the ray.*

Definition 2.6. (**Directions of a Convex Set**): Given a convex set, a nonzero vector $\mathbf{d}$ is called a *recession direction of the set* if for each $\mathbf{x}_0$ in the set, the ray $\{\mathbf{x}_0 + \lambda\mathbf{d} : \lambda \geq 0\}$ also belongs to the set.

Definition 2.7. (**Extreme Directions of A Convex Set**): An *extreme direction* of a convex set is a direction of the set that cannot be represented as a positive combination of two distinct directions of the set.

Let the polyhedral set is in the following form:

$$\mathbf{X} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\}, \tag{2.1}$$

where $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{b}$ is an m-dimensional vector. We will use this form to explain some definitions below.

Theorem 2.8. *If the polyhedral set $\boldsymbol{X}$ is nonempty. Then a nonzero $\boldsymbol{d}$ is a recession direction of $\boldsymbol{X}$ if and only if*

$$\begin{aligned} \boldsymbol{A}(\boldsymbol{x} + \lambda\boldsymbol{d}) &\leq \boldsymbol{b} \\ \boldsymbol{x} + \lambda\boldsymbol{d} &\geq \boldsymbol{0} \end{aligned} \tag{2.2}$$

for each $\lambda \geq 0$ and each $\boldsymbol{x} \in \boldsymbol{X}$.

Theorem 2.8 will be used to prove our theorem in chapter 3.

The following definitions will lead to an extreme point associated to the optimal solution.

Definition 2.9. (**Defining Hyperplanes**): Let the hyperplanes associated with the $(m + n)$ defining half-spaces of $\mathbf{X}$ be referred to as *defining hyperplanes* of $\mathbf{X}$.

Note that the $(m + n)$ defining half-spaces of $\mathbf{X}$ consist of m constraints and n nonnegativity constraints.

Definition 2.10. (**Extreme Points**): A point $\bar{\mathbf{x}} \in \mathbf{X}$ is said to be an extreme point of set $\mathbf{X}$ if $\bar{\mathbf{x}}$ lies on some n linearly independent defining hyperplanes of $\mathbf{X}$.

The following theorem is one of the most important theorems of the linear programming problem.

Theorem 2.11. *(**Representation Theorem for the General case**): Let* $\boldsymbol{X} = \{\boldsymbol{x} \mid \boldsymbol{Ax} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ *be a nonempty polyhedral set. Then the set of extreme points is not empty and has a finite number of points, say* $\boldsymbol{x}_1$, $\boldsymbol{x}_2$,...,$\boldsymbol{x}_k$, *the set of extreme directions is empty if and only if* $\boldsymbol{X}$ *is bounded. If* $\boldsymbol{X}$ *is not bounded, then the set of extreme directions is nonempty and has a finite number of vectors, say* $\boldsymbol{d}_1$, $\boldsymbol{d}_2$,...,$\boldsymbol{d}_l$. *Moreover,* $\bar{\boldsymbol{x}} \in \boldsymbol{X}$ *if and only if it can be represented as a convex combination of* $\boldsymbol{x}_1$, $\boldsymbol{x}_2$,...,$\boldsymbol{x}_k$ *plus a nonnegative linear combination of* $\boldsymbol{d}_1$, $\boldsymbol{d}_2$,...,$\boldsymbol{d}_k$, *that is,*

$$\begin{aligned} \bar{\boldsymbol{x}} = \sum_{j=1}^{k} \lambda_j \boldsymbol{x}_j \quad &+ \quad \sum_{j=1}^{l} \mu_j \boldsymbol{d}_j \\ \sum_{j=1}^{k} \lambda_j \quad &= \quad 1 \\ \lambda_j \quad &\geq \quad 0, j = 1, 2, ..., k \\ \mu_j \quad &\geq \quad 0, j = 1, 2, ..., l. \end{aligned} \tag{2.3}$$

2.2 The Simplex Method

Consider the following standard linear programming problem :

$$\begin{aligned} \textbf{LP:} \quad \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.4}$$

where $\mathbf{A}$ is an $m \times n$ matrix with rank m. Since the simplex algorithm was designed to deal with the standard form, we will use this form in this section.

Definition 2.12. (**Basic Feasible Solutions**): Consider the system $\mathbf{Ax = b}$ and $\mathbf{x} \geq \mathbf{0}$, where $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{b}$ is an m-dimensional vector. Suppose that rank($\mathbf{A}$, $\mathbf{b}$) = rank($\mathbf{A}$) = m. After possibly rearranging the columns of $\mathbf{A}$, let $\mathbf{A} = [\mathbf{B, N}]$ where $\mathbf{B}$ is an $m \times m$ invertible matrix and $\mathbf{N}$ is an $m \times (n - m)$ matrix. The solution $\mathbf{x} = \begin{bmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{bmatrix}$ to the equation $\mathbf{Ax = b}$, where

$$\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} \quad \text{and} \quad \mathbf{x_N} = \mathbf{0}$$

is called a *basic solution* of the system.

If $\mathbf{x_B} \geq \mathbf{0}$, then $\mathbf{x}$ is called a *basic feasible solution* of the system.

Here $\mathbf{B}$ is called the *basic matrix* and $\mathbf{N}$ is called the *nonbasic matrix*.

The components of $\mathbf{x_B}$ are called *basic variables* and the components of $\mathbf{x_N}$ are called *nonbasic variables*.

If $\mathbf{x_B} > \mathbf{0}$, then $\mathbf{x}$ is called a *nondegenerate basic feasible solution*, and if at least one component of $\mathbf{x_B}$ is zero, then $\mathbf{x}$ is called a *degenerate basic feasible solution*.

The following theorems show a relation between an extreme point and a basic feasible solution, and some properties of extreme directions lead to the existence of the optimal solution.

Theorem 2.13. *The collection of extreme points corresponds to the collection of basic feasible solutions, and both are nonempty, provided that the feasible region is nonempty.*

The following theorem will be used to prove the unbounded optimal solution in our theorem.

Theorem 2.14. *Assume that a feasible region is nonempty. Then a finite optimal solution exists if and only if* $\boldsymbol{cd}_j \leq 0$ *for* $j = 1, 2, ..., l$, *where* $\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_l$ *are the extreme directions of the feasible region. Otherwise, the optimal solution value is unbounded.*

Theorem 2.15. *If an optimal solution exists, then an optimal extreme point exists.*

Theorem 2.16. *For every extreme point (basic feasible solution), there is a corresponding basis (not necessarily unique), and, conversely, for every basis there is a corresponding (unique) extreme point. Moreover, if an extreme point has more than one basis representing it, then it is degenerate. Conversely, a degenerate extreme point has more than one basis representing it if and only if the system* $\boldsymbol{Ax} = \boldsymbol{b}$ *itself does not imply that the degenerate basic variables corresponding to an associated basis are identically zero.*

2.2.1 Key to the Simplex Method

Consider a matrix $\mathbf{A}$ in the equation $\mathbf{Ax = b}$ being partitioned as $\mathbf{A} = [\mathbf{B}, \mathbf{N}]$, and let $\mathbf{x}^{\mathrm{T}} = [\mathbf{x}_{\mathbf{B}}^{\mathrm{T}},\ \mathbf{x}_{\mathbf{N}}^{\mathrm{T}}]$ and $\mathbf{c}^{\mathrm{T}} = [\mathbf{c}_{\mathbf{B}}^{\mathrm{T}},\ \mathbf{c}_{\mathbf{N}}^{\mathrm{T}}]$ with $\mathbf{x}_{\mathbf{B}}^{\mathrm{T}}$ and $\mathbf{c}_{\mathbf{B}}^{\mathrm{T}}$ are associated columns of $\mathbf{B}$ and $\mathbf{x}_{\mathbf{N}}^{\mathrm{T}}$ and $\mathbf{c}_{\mathbf{N}}^{\mathrm{T}}$ are associated with columns of $\mathbf{N}$. Then, the problem (2.4) can be written as follow:

$$\begin{array}{rrcrl} \text{Maximize} & \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{x}_{\mathbf{B}} & + & \mathbf{c}_{\mathbf{N}}^{\mathrm{T}}\mathbf{x}_{\mathbf{N}} & \\ \text{subject to} & \mathbf{B}\mathbf{x}_{\mathbf{B}} & + & \mathbf{N}\mathbf{x}_{\mathbf{N}} & = \mathbf{b} \\ & \mathbf{x}_{\mathbf{B}} & , & \mathbf{x}_{\mathbf{N}} & \geq \mathbf{0}. \end{array} \tag{2.5}$$

Suppose that we have a basic feasible solution $\begin{bmatrix} \mathbf{x}_{\mathbf{B}} \\ \mathbf{x}_{\mathbf{N}} \end{bmatrix} = \begin{bmatrix} \mathbf{B}^{-1}\mathbf{b} \\ \mathbf{0} \end{bmatrix} \geq \mathbf{0}$. Consider the equation

$$\mathbf{B}\mathbf{x}_{\mathbf{B}} + \mathbf{N}\mathbf{x}_{\mathbf{N}} = \mathbf{b} \tag{2.6}$$

$$\mathbf{x}_{\mathbf{B}} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_{\mathbf{N}}. \tag{2.7}$$

By substituting $\mathbf{x}_{\mathbf{B}}$ into the objective function in the problem (2.4) and letting z denote the objective function value, $J_{\mathbf{B}}$ and $J_{\mathbf{N}}$ are the the current set of the indices of the basic and nonbasic variables respectively, we get

$$z = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{x}_{\mathbf{B}} + \mathbf{c}_{\mathbf{N}}^{\mathrm{T}}\mathbf{x}_{\mathbf{N}} \tag{2.8}$$

$$= \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\left(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_{\mathbf{N}}\right) + \mathbf{c}_{\mathbf{N}}^{\mathrm{T}}\mathbf{x}_{\mathbf{N}} \tag{2.9}$$

$$= \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{b} - \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{N}\mathbf{x}_{\mathbf{N}} + \mathbf{c}_{\mathbf{N}}^{\mathrm{T}}\mathbf{x}_{\mathbf{N}} \tag{2.10}$$

$$= \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{b} - \left(\mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_{\mathbf{N}}^{\mathrm{T}}\right)\mathbf{x}_{\mathbf{N}} \tag{2.11}$$

$$= \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{b} - \sum_{j \in J_{\mathbf{N}}} \left(\mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{A}_{:j} - c_j\right) x_j \tag{2.12}$$

$$= z_0 - \sum_{j \in J_{\mathbf{N}}} \left(z_j - c_j\right) x_j, \tag{2.13}$$

where $z_j = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{A}_{:j}$ for each nonbasic variable, $z_0 = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{b}$ and $\mathbf{A}_{:j}$ is the j^{th} column of $\mathbf{A}$.

Since the problem (2.4) is the maximization problem and z_0 is a constant, the objective value will increase when there exists $z_j - c_j < 0$. Therefore, the optimal

solution is reached when the index set

$$J = \{z_j - c_j < 0 \mid j \in J_{\mathbf{N}}\} = \varnothing. \tag{2.14}$$

For the current basic feasible solution, since $x_j = 0$ for all $j \in J_{\mathbf{N}}$, $z = z_0$. When $z_j - c_j \geq 0$ for all $j \in J_{\mathbf{N}}$, $z \leq z_0$ for any feasible solution. Therefore, the current basic feasible solution is the optimal solution.

The simplex algorithm is an iterative method that moves from one basis to an adjacent basis by entering one variable from the nonbasic variable set into the basis, and remove one variable from the basic variable set from the basis. The variable which will be introduced to the basis is called the *entering variable* and the variable which will be removed from the basis is called the *leaving variable*. Each step is known as *iteration* or *pivot*. So the simplex algorithm can be summarized below:

The Simplex Algorithm (Maximization Problem)
INITIALIZATION STEP: Choose a starting basis $\mathbf{B}$.
MAIN STEP:

(i) Solve the system $\mathbf{B}\mathbf{x}_{\mathbf{B}} = \mathbf{b}$. Then, we have $\mathbf{x}_{\mathbf{B}} = \mathbf{B}^{-1}\mathbf{b} = \bar{\mathbf{b}}$.

(ii) Solve the system $\mathbf{w}^{\mathrm{T}}\mathbf{B} = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}$, $\mathbf{w}$ is called the vector of *simplex multipliers*.

(iii) Calculate $z_j = \mathbf{w}^{\mathrm{T}}\mathbf{A}_{:j}$ for all $j \in J_{\mathbf{N}}$.

(iv) Determine the entering variable k such that

$$z_k - c_k = \min\{z_j - c_j \mid j \in J_{\mathbf{N}}\}. \tag{2.15}$$

(v) If $z_k - c_k \geq 0$, then stop. *The optimal solution is the current basic feasible solution.*

Otherwise, solve the system $\mathbf{B}\mathbf{y}_k = \mathbf{A}_{:k}$.

(vi) If $\mathbf{y}_k \leq \mathbf{0}$, then stop. *The optimal solution is unbounded.*

Otherwise, determine the index r of the leaving variable $\mathbf{x}_{\mathbf{B}_r}$ by *minimum ratio test* such that

$$\frac{\bar{b}_r}{y_{rk}} = \min_{1 \le i \le m} \left\{ \frac{\bar{b}_i}{y_{ik}} \mid y_{ik} > 0 \right\}. \tag{2.16}$$

(vii) Update the basis $\mathbf{B}$ where $\mathbf{A}_{:k}$ replaces $\mathbf{A}_{:\mathbf{B}_r}$

(viii) Update the index set $J_{\mathbf{N}}$, and repeat the MAIN STEP (step (i) - step (viii)).

2.2.2 The Simplex Algorithm in Tableau Format

From the previous algorithm, the linear system of equations: $\mathbf{B}\mathbf{x}_{\mathbf{B}} = \mathbf{b}$, $\mathbf{w}^{\mathrm{T}}\mathbf{B} = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}$, and $\mathbf{B}\mathbf{y}_k = \mathbf{A}_k$ need to be solved. Solving and updating these systems can be handled easily if we use the tableau format to describe the simplex method.

Suppose that we have a starting basic feasible solution $\mathbf{x} = \begin{bmatrix} \mathbf{x}_{\mathbf{B}} \\ \mathbf{x}_{\mathbf{N}} \end{bmatrix}$ with basis $\mathbf{B}$. We can represent the problem (2.4) as follows:

$$\begin{aligned} \text{Maximize} \quad & z \\ \text{subject to} \quad & z - \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{x}_{\mathbf{B}} - \mathbf{c}_{\mathbf{N}}^{\mathrm{T}}\mathbf{x}_{\mathbf{N}} = 0 && (2.17) \\ & \mathbf{B}\mathbf{x}_{\mathbf{B}} + \mathbf{N}\mathbf{x}_{\mathbf{N}} = \mathbf{b} && (2.18) \\ & \mathbf{x}_{\mathbf{B}}, \quad \mathbf{x}_{\mathbf{N}} \ge \mathbf{0}. \end{aligned}$$

From the equation (2.18), we have

$$\mathbf{x}_{\mathbf{B}} + \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_{\mathbf{N}} = \mathbf{B}^{-1}\mathbf{b}. \tag{2.19}$$

Multiplying (2.19) by $\mathbf{c}_{\mathbf{B}}^{\mathrm{T}}$ and adding to the equation (2.17), we get

$$z + \mathbf{0}\mathbf{x}_{\mathbf{B}} + \left(\mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_{\mathbf{N}}^{\mathrm{T}}\right)\mathbf{x}_{\mathbf{N}} = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{b}. \tag{2.20}$$

Therefore, the equivalent system is as follows:

$$\begin{aligned} &\text{Maximize} && z \\ &\text{subject to} && z + \mathbf{0}\mathbf{x_B} + \left(\mathbf{c_B^T B^{-1} N} - \mathbf{c_N^T}\right)\mathbf{x_N} = \mathbf{c_B^T B^{-1} b} && (2.21) \\ & && \mathbf{x_B} + \mathbf{B^{-1} N x_N} = \mathbf{B^{-1} b} && (2.22) \\ & && \mathbf{x_B},\ \mathbf{x_N} \geq \mathbf{0}. \end{aligned}$$

Since $\mathbf{x_N} = \mathbf{0}$, we get $\mathbf{x_B} = \mathbf{B^{-1} b}$ and $z = \mathbf{c_B^T B^{-1} b}$. For convenience, we can represent the current basic feasible solution with basis $\mathbf{B}$ in the following tableau.

	z	$\mathbf{x_B}$	$\mathbf{x_N}$	**RHS**	
z	1	0	$\mathbf{c_B^T B^{-1} N} - \mathbf{c_N^T}$	$\mathbf{c_B^T B^{-1} b}$	Row 0
$\mathbf{x_B}$	0	$\mathbf{I}$	$\mathbf{B^{-1} N}$	$\mathbf{B^{-1} b}$	Row 1 through m

The objective row is referred to as row 0 and the remaining rows are rows 1 through m. The right-hand-side column (RHS) contains the value of the basic variables including the objective value.

Consider row 0, $\mathbf{c_B^T B^{-1} N} - \mathbf{c_N^T}$ consists of $z_j - c_j$ and $\mathbf{B^{-1} N}$ consists of $\mathbf{y}_j = \mathbf{B^{-1} A}_{:j}$ for all nonbasic variables. Therefore, we can determine the entering variable by considering row 0. If each $z_j - c_j \geq 0$, the current basic feasible solution is the optimal solution. Otherwise, nonbasic variables can be increased. If x_k is selected as an entering variable, then we can determine how much x_k can be increased by evaluating the minimum ratio between vector $\mathbf{y}_k$ which is stored in the tableau in rows 1 through m under the variable x_k and $\mathbf{B^{-1} b}$. If $\mathbf{y}_k \leq \mathbf{0}$, then the optimal objective value is unbounded. Otherwise, x_k will be blocked by the minimum ratio test. One of the current basic variables which blocks x_k will be the leaving variable and will be droped to zero. If x_k enters the basis and $x_{\mathbf{B}_r}$ leaves the basis, then the tableau will be updated by pivoting on y_{rk} that can be stated as follows:

(i) Divide row r by y_{rk}.

(ii) Update the i^{th} row by adding to it $-y_{ik}$ times the new r^{th} row for $i = 1, 2, ..., m$ and $i \neq r$.

(iii) Update the row zero by adding to it $c_k - z_k$ times the new r^{th} row.

So the simplex method in tableau format can be summarized below.

The Simplex Method (Maximization Problem)

INITIALIZATION STEP: Find an initial basic feasible solution with basis $\mathbf{B}$. Form the following initial tableau:

	z	$\mathbf{x_B}$	$\mathbf{x_N}$	**RHS**
z	1	0	$\mathbf{c_B^T B^{-1} N - c_N^T}$	$\mathbf{c_B^T B^{-1} b}$
$\mathbf{x_B}$	0	$\mathbf{I}$	$\mathbf{B^{-1} N}$	$\mathbf{B^{-1} b}$

MAIN STEP:

(i) Determine the entering variable k such that

$$z_k - c_k = \min\{z_j - c_j \mid j \in J_{\mathbf{N}}\}. \tag{2.23}$$

If $z_k - c_k \geq 0$, then stop. *The optimal solution is the current basic feasible solution.* Otherwise, examine $\mathbf{y}_k = \mathbf{B}^{-1}\mathbf{A}_{:k}$.

If $\mathbf{y}_k \leq \mathbf{0}$, then stop. *The optimal solution is unbounded.* Otherwise, determine the index r of the leaving variable $\mathbf{x}_{\mathbf{B}_r}$ by *minimum ratio test* such that

$$\frac{\bar{b}_r}{y_{rk}} = \min_{1 \leq i \leq m} \left\{ \frac{\bar{b}_i}{y_{ik}} \mid y_{ik} > 0 \right\}. \tag{2.24}$$

(ii) Update the tableau by pivoting at y_{rk}. Update the basic and nonbasic variables where x_k enters the basis and $x_{\mathbf{B}_r}$ leaves the basis, and repeat the MAIN STEP.

2.2.3 The Initial Basic Feasible Solution

Since the simplex method starts at a basic feasible solution with a basis $\mathbf{B}$ which $\mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$, the question arises how can we find the initial basis.

Consider an easy case where $\mathbf{b} \geq \mathbf{0}$, suppose that the constraints are of the following form:

$$\begin{aligned} \mathbf{Ax} &\leq \mathbf{b}, \\ \mathbf{x} &\geq \mathbf{0} \end{aligned} \tag{2.25}$$

where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{b}$ is a nonnegative m- dimensional vector. Recall that the simplex method is designed to deal with the standard form, so the slack vector $\mathbf{s}$ is added as follows:

$$\begin{aligned} \mathbf{Ax} \;+\; \mathbf{s} \;&= \mathbf{b}, \\ \mathbf{x},\; \mathbf{s} \;&\geq \mathbf{0} \end{aligned} \tag{2.26}$$

where $\mathbf{s}$ is a nonnegative m- dimensional vector. So the new constraint matrix is $[\mathbf{A}, \mathbf{I}]$ having rank m. Let $\mathbf{I}$ be the basis and $\mathbf{A}$ be the nonbasic matrix, then $\mathbf{s}$ is the basic vector. Therefore, the initial basic feasible solution is $(\mathbf{x}, \mathbf{s})^{\mathrm{T}} = (\mathbf{0}, \mathbf{b})^{\mathrm{T}} \geq \mathbf{0}$, that is $\mathbf{x} = \mathbf{0}$ is a feasible point, and the simplex method can be performed.

Example 2.17. Consider the following problem:

$$\begin{array}{lrcrcrcl} \text{Maximize} & x_1 & - & 3x_2 & + & 2x_3 & & \\ \text{subject to} & 5x_1 & - & 3x_2 & - & 2x_3 & \leq & 1 \\ - & 2x_1 & + & 4x_2 & + & x_3 & \leq & 2 \\ & x_1, & & x_2, & & x_3 & \geq & 0 \end{array} \tag{2.27}$$

Solution. Before starting the simplex algorithm, the problem must be in the standard form. By adding slack variables, we get

$$\begin{array}{lrcrcrcrcrcl} \text{Maximize} & x_1 & - & 3x_2 & + & 2x_3 & & & & & \\ \text{subject to} & 5x_1 & - & 3x_2 & - & 2x_3 & + & s_1 & & = & 1 \\ - & 2x_1 & + & 4x_2 & + & x_3 & & & + & s_2 & = & 2 \\ & x_1, & & x_2, & & x_3, & & s_1, & & s_2 & \geq & 0 \end{array} \tag{2.28}$$

Let the identity matrix be an initial basis. Then, s_1, s_2 are basic variables with the identity basis, x_1, x_2, x_3 are nonbasic variables, and $(x_1, x_2, x_3, s_1, s_2)^{\mathbf{T}} = (0, 0, 0, 1, 2)^{\mathrm{T}} \geq \mathbf{0}$ is the initial basic feasible solution. The simplex method can start. Put it in the initial tableau,

	z	x_1	x_2	x_3	s_1	s_2	RHS
z	1	-1	3	-2	0	0	0
s_1	0	5	-3	-2	1	0	1
s_2	0	-2	4	(1)	0	1	2

Determine the pivot column k by Dantzigs' pivot rule, $z_k - c_k = -2 = \min\{z_j - c_j \mid j \in \mathbf{J_N}\}$ which $k = 3$. Determine the pivot row r by the minimum ratio test, $\frac{\bar{b}_2}{y_{23}} = \min\{\frac{\bar{b}_2}{y_{23}} \mid y_{23} \geq 0\} = \min\{2\}$ which $r = 2$. Then, we pivot at y_{23}, x_3 enters the basis and s_2 leaves the basis. After pivoting, we get

	z	x_1	x_2	x_3	s_1	s_2	RHS
z	1	-5	11	0	0	2	4
s_1	0	(1)	1	0	1	2	5
x_3	0	-2	4	1	0	1	2

Similarly, we get the pivot element at y_{12}, that is, x_2 enters the basis and s_1 leaves the basis. After pivoting, we get

	z	x_1	x_2	x_3	s_1	s_2	RHS
z	1	0	16	0	5	12	29
x_1	0	1	1	0	1	2	5
x_3	0	0	6	1	2	5	12

Since $z_j - c_j \geq 0$ for all j, the optimal solution is found at $(x_1^*, x_2^*, x_3^*, s_1^*, s_2^*)^{\mathrm{T}} = (5, 0, 12, 0, 0)^{\mathrm{T}}$ with the objective value $z^* = 29$. □

However, in many problems, the initial basis can not easily be obtained from the identity matrix such as when the constraints are in the following form:

$$\begin{aligned} \mathbf{Ax} &\leq \mathbf{b}, \\ \mathbf{x} &\geq \mathbf{0}, \end{aligned} \tag{2.29}$$

where the vector $\mathbf{b}$ is negative. By adding the slack vector $\mathbf{s}$, we have the standard form as $\mathbf{Ax}+\mathbf{s} = \mathbf{b}$, $\mathbf{x}$, $\mathbf{s} \geq \mathbf{0}$, that is, the new constraint matrix is $[\mathbf{A},\ \mathbf{I}]$. If let $\mathbf{I}$ be the basis and $\mathbf{A}$ be the nonbasic matrix, then $(\mathbf{x}, \mathbf{s})^{\mathrm{T}} = (\mathbf{0}, \mathbf{b})^{\mathrm{T}} \not\geq \mathbf{0}$ violates the nonnegativity constraints.

Another problems occur when the constraints are of the following form:

$$\begin{aligned} \mathbf{Ax} &\geq \mathbf{b}, \\ \mathbf{x} &\geq \mathbf{0}, \end{aligned} \tag{2.30}$$

where $\mathbf{b} \not\leq \mathbf{0}$. To get the standard form, the surplus vector $\mathbf{s}$ is subtracted and we get as follows:

$$\begin{aligned} \mathbf{Ax} \quad - \quad \mathbf{s} \quad &= \mathbf{b}, \\ \mathbf{x}, \quad \mathbf{s} \quad &\geq \mathbf{0}. \end{aligned} \tag{2.31}$$

The new constraints matrix is $[\mathbf{A}, \ -\mathbf{I}]$ which is difficult to pick a basis $\mathbf{B}$ with $\mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$. We can handle this problem by introducing artificial variables to be an initial basis. We describe the artificial variable techniques in the following section.

2.3 Artificial Variable Techniques

For some cases that we can not pick a basis $\mathbf{B}$ from the standard form $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$, where $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{b} \geq \mathbf{0}$, we will introduce *artificial variables* to the system to get a starting basic feasible solution as follows:

$$\begin{aligned} \mathbf{Ax} \quad + \quad \mathbf{x}_a \quad &= \mathbf{b}, \\ \mathbf{x}, \quad \mathbf{x}_a \quad &\geq \mathbf{0}, \end{aligned} \tag{2.32}$$

where $\mathbf{x}_a$ is a vector of artificial variables. The new constraint matrix is $[\mathbf{A}, \ \mathbf{I}]$, then this gives a basic feasible solution with $\mathbf{x}_a = \mathbf{b} \geq \mathbf{0}$ and $\mathbf{x} = \mathbf{0}$ and the simplex method can be applied.

Example 2.18. Consider the following constrains:

$$\begin{aligned} x_1 \quad + \quad x_2 \quad &\leq \quad 3 \\ x_1 \quad + \quad 4x_2 \quad &\geq \quad 4 \\ 5x_1 \quad + \quad x_2 \quad &\geq \quad 5 \\ x_1, \qquad x_2, \quad &\geq \quad 0 \end{aligned} \tag{2.33}$$

Manipulating the problem to the standard form, we get

$$\begin{aligned} x_1 \quad + \quad x_2 \quad + \quad s_1 \qquad\qquad\qquad &= \quad 3 \\ x_1 \quad + \quad 4x_2 \qquad - \quad s_2 \qquad &= \quad 4 \\ 5x_1 \quad + \quad x_2 \qquad\qquad - s_3 \quad &= \quad 5 \\ x_1, \qquad x_2, \qquad s_1, \qquad s_2, \qquad s_3 \quad &\geq \quad 0 \end{aligned} \tag{2.34}$$

This constraint matrix has no identity matrix. We can introduce three artificial variables to get an initial basic feasible solution. However, since s_1 has the coefficient of 1, we need to add only two artificial variables x_{a_1} and x_{a_2}, then we get the following system.

$$\begin{array}{rcrcrcrcrcrcl} x_1 & + & x_2 & + & s_1 & & & & & & & = & 3 \\ x_1 & + & 4x_2 & & & - & s_2 & & + & x_{a_1} & & = & 4 \\ 5x_1 & + & x_2 & & & & & - & s_3 & & + & x_{a_2} & = & 5 \\ x_1, & & x_2, & & s_1, & & s_2, & & s_3, & x_{a_1}, & & x_{a_2} & \geq & 0 \end{array} \tag{2.35}$$

Here, we have the identity matrix with the initial basic feasible solution, $s_1 = 3, x_{a_1} = 4, x_{a_2} = 5$. Remaining variables are nonbasic variables having values equal to zeroes.

Although we have the basic feasible solution, the problem have been changed from adding artificial variables. To get back to the original problem, we need to force these artificial variables to zero, because $\mathbf{Ax} = \mathbf{b}$ if and only if $\mathbf{Ax}+\mathbf{x}_a = \mathbf{b}$ with $\mathbf{x}_a = \mathbf{0}$. In other words, adding artificial variables is only a tool to get a basic feasible solution for starting the simplex method. The two well-known techniques for eliminating artificial variables are the two-phase method and the big-M method.

2.3.1 Two-Phase Method

Two-Phase method is a method to find an initial basic feasible solution of the linear programming problem. The algorithm is separated into two phases. Phase I: finds a basic feasible solution and phase II starts the simplex method from the current basic feasible solution from phase I.

Phase I obtains a basic feasible solution of the original problem. The new linear programming problem which is minimized the sum of artificial variables subject to $\mathbf{Ax}+\mathbf{x}_a = \mathbf{b}$, $\mathbf{x}_a \geq \mathbf{0}$ is solved. If an original problem has a feasible solution, then the objective value of this problem is zero, that is, values of all variables drop to zero. Then, they leave the basis, and the basis consists of legitimate variables (if

all artificial variables are out of the basis). So we get a basic feasible solution for the original problem $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$, and the simplex method can start with the original objective function. If the objective value of phase I is not zero, then the original problem is infeasible. The two-phase method is summarized as follows:

Phase I:

Solve the following linear programming problem with the starting basic feasible solution $\mathbf{x} = \mathbf{0}$ and $\mathbf{x}_a = \mathbf{b}$:

$$\begin{array}{lrcrl} \text{Minimize} & x_0 & = & \mathbf{1}^{\mathrm{T}}\mathbf{x}_a & \\ \text{subject to} & \mathbf{Ax} & + & \mathbf{x}_a & = \mathbf{b} \\ & \mathbf{x}, & & \mathbf{x}_a & \geq \mathbf{0}. \end{array} \tag{2.36}$$

At optimality, if $\mathbf{x}_a \neq \mathbf{0}$, then stop; the original problem is infeasible. Otherwise, let basic and nonbasic legitimate variables be $\mathbf{x_B}$ and $\mathbf{x_N}$. (We are assuming that all artificial variables left the basis.) Proceed to Phase II.

Phase II:

Solve the following linear programming problem with the starting basic feasible solution $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b}$ and $\mathbf{x_N} = \mathbf{0}$.

$$\begin{array}{lrcrl} \text{Maximize} & z = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{x_B} & + & \mathbf{c}_{\mathbf{N}}^{\mathrm{T}}\mathbf{x_N} & \\ \text{subject to} & \mathbf{x_B} & + & \mathbf{B}^{-1}\mathbf{N}\mathbf{x_N} & = \mathbf{B}^{-1}\mathbf{b} \\ & \mathbf{x_B}, & & \mathbf{x_N} & \geq \mathbf{0}. \end{array} \tag{2.37}$$

This problem is equivalent to the original problem and it can be solved by the simplex method.

Example 2.19. Consider the following linear programming problem:

$$\begin{array}{lrcrcrl} \text{Maximize} & 2x_1 & + & x_2 & & & \\ \text{Subject to} & x_1 & + & x_2 & \leq & 3 & \\ & x_1 & + & 2x_2 & \geq & 4 & \\ & 3x_1 & + & x_2 & \geq & 5 & \\ & x_1, & & x_2, & \geq & 0 & \end{array} \tag{2.38}$$

Manipulating the problem to the standard form, we get

$$\begin{aligned}
&\text{Maximize } 2x_1 + x_2 \\
&\text{Subject to } x_1 + x_2 + s_1 = 3 \\
&\qquad x_1 + 2x_2 - s_2 = 4 \\
&\qquad 3x_1 + x_2 - s_3 = 5 \\
&\qquad x_1, \quad x_2, \quad s_1, \quad s_2, \quad s_3 \geq 0
\end{aligned} \tag{2.39}$$

This constraint matrix has no identity submatrix. We need to add two artificial variables x_{a_1} and x_{a_2}, then we get the following system.

$$\begin{aligned}
&\text{Minimize } 2x_1 + x_2 \\
&\text{Subject to } x_1 + x_2 + s_1 = 3 \\
&\qquad x_1 + 2x_2 - s_2 + x_{a_1} = 4 \\
&\qquad 3x_1 + x_2 - s_3 + x_{a_2} = 5 \\
&\qquad x_1, \quad x_2, \quad s_1, \quad s_2, \quad s_3, \quad x_{a_1}, \quad x_{a_2} \geq 0
\end{aligned} \tag{2.40}$$

Therefore, phase I is written as follows:

Phase I:

$$\begin{aligned}
&\text{Minimize } x_{a_1} + x_{a_2} \\
&\text{Subject to } x_1 + x_2 + s_1 = 3 \\
&\qquad x_1 + 2x_2 - s_2 + x_{a_1} = 4 \\
&\qquad 3x_1 + x_2 - s_3 + x_{a_2} = 5 \\
&\qquad x_1, \quad x_2, \quad s_1, \quad s_2, \quad s_3, \quad x_{a_1}, \quad x_{a_2} \geq 0
\end{aligned} \tag{2.41}$$

Here, we have s_1, x_{a_1} and x_{a_2} being the basic variables, so the tableau can be written below:

	x_0	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
x_0	1	0	0	0	0	0	-1	-1	0
s_1	0	1	1	1	0	0	0	0	3
x_{a_1}	0	1	2	0	-1	0	1	0	4
x_{a_2}	0	3	1	0	0	-1	0	1	5

Since the reduced cost of basic variables will be zero, add row 2 and row 3 to row 0, we have

	x_0	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
x_0	1	4	3	0	-1	-1	0	0	9
s_1	0	1	1	1	0	0	0	0	3
x_{a_1}	0	1	2	0	-1	0	1	0	4
x_{a_2}	0	3	1	0	0	-1	0	1	5

For the minimization problem, the entering variable k can be chosen by $z_k - c_k = \max_{j \in \mathbf{N}} \{z_j - c_j\}$. All iterations in phase I are shown below.

	x_0	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
x_0	1	4	3	0	-1	-1	0	0	9
s_1	0	1	1	1	0	0	0	0	3
x_{a_1}	0	1	2	0	-1	0	1	0	4
x_{a_2}	0	(3)	1	0	0	-1	0	1	5

	x_0	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
x_0	1	0	5/3	0	-1	1/3	0	-4/3	7/3
s_1	0	0	2/3	1	0	1/3	0	- 1/3	4/3
x_{a_1}	0	0	(5/3)	0	-1	1/3	1	-1/3	7/3
x_1	0	1	1/3	0	0	-1/3	0	1/3	5/3

	x_0	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
x_0	1	0	0	0	0	0	-1	-1	0
s_1	0	0	0	1	2/5	1/5	-2/5	-1/5	2/5
x_2	0	0	1	0	-3/5	1/5	3/5	-1/5	7/5
x_1	0	1	0	0	1/5	-2/5	-1/5	2/5	6/5

The last iteration of phase I has no artificial variables in the basis the starting basic feasible solution $(x_1, x_2, s_1)^{\mathrm{T}} = (6/5, 7/5, 2/5)^{\mathrm{T}}$. Phase II can be started at

this basic feasible solution which the original objective function is maximized and all artificial variables are removed.

Phase II:

	z	x_1	x_2	s_1	s_2	s_3	RHS
z	1	-2	-1	0	0	0	0
s_1	0	0	0	1	2/5	1/5	2/5
x_2	0	0	1	0	-3/5	1/5	7/5
x_1	0	1	0	0	1/5	-2/5	6/5

Multiply row 2 and row 3 by 1 and 2, respectively, and add to row 0, producing $z_1 - c_1 = z_2 - c_2 = 0$. For the maximization problem, the entering variable k is chosen by $z_k - c_k = \min_{j \in \mathbf{N}} \{z_j - c_j\}$. So each iteration in phase II can be performed below.

	z	x_1	x_2	s_1	s_2	s_3	RHS
z	1	0	0	0	-1/5	-3/5	19/5
s_1	0	0	0	1	2/5	(1/5)	2/5
x_2	0	0	1	0	-3/5	1/5	7/5
x_1	0	1	0	0	1/5	-2/5	6/5

	z	x_1	x_2	s_1	s_2	s_3	RHS
z	1	0	0	3	1	0	5
s_3	0	0	0	5	2	1	2
x_2	0	0	1	-1	-1	0	1
x_1	0	1	0	2	1	0	2

Since $z_j - c_j \geq 0$ for all nonbasic variables, the optimal solution is found at $(x_1, x_2)^{\mathrm{T}} = (2, 1)^{\mathrm{T}}$ with the objective value 5.

Note that Phase I moved from the infeasible point (0,0) to the point (0, 5/3), and finally to the feasible point (6/5, 7/5). From this feasible point, Phase II

moved to the feasible point (2, 1) and stopped at this point since it is the optimal

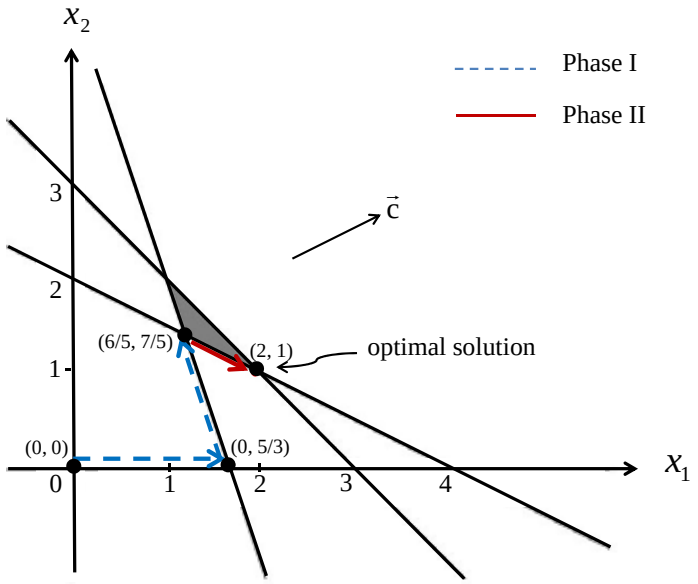


Figure 2.1: Example of Two-Phase method

2.3.2 The Big-M Method

Another technique for eliminating artificial variables is to assign very big coefficients for these variables in the original objective function. To illustrate, suppose that we want to solve the following linear programming problem, where $\mathbf{b} \geq \mathbf{0}$:

$$\begin{aligned} \text{P:} \quad \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.42}$$

If no convenient basis is known, we can introduce the artificial vector $\mathbf{x}_a$, which leads to the following system:

$$\begin{aligned} \mathbf{Ax} + \mathbf{x}_a &= \mathbf{b} \\ \mathbf{x}, \quad \mathbf{x}_a &\geq \mathbf{0}. \end{aligned} \tag{2.43}$$

The starting basic feasible solution is given by $\mathbf{x}_a = \mathbf{b}$. In order to reflect the undesirability of a nonzero artificial vector, the objective function is modified such

that a large penalty is assigned for any such solution. More specifically consider the following problem.

$$\begin{array}{lllllll} \text{P(M):} & \text{Maximize} & z_{\text{big-M}} = & \mathbf{c^T x} & - & M\mathbf{1}^{\mathrm{T}}\mathbf{x}_a & \\ & \text{subject to} & & \mathbf{Ax} & + & \mathbf{x}_a & = \mathbf{b} \\ & & & \mathbf{x}, & & \mathbf{x}_a & \geq \mathbf{0}, \end{array} \tag{2.44}$$

where M is a very large positive number. The term $-M\mathbf{1}^{\mathrm{T}}\mathbf{x}_a$ can be interpreted as a penalty to be assigned to a solution with $\mathbf{x}_a \neq \mathbf{0}$. Alternatively, the foregoing strategy can be interpreted as one that minimizes $\mathbf{1}^{\mathrm{T}}\mathbf{x}_a$ with priority one, and among all alternative optimal solutions for this objective, maximizes the secondary objective $\mathbf{c}^{\mathrm{T}}\mathbf{x}$. Hence, even though the starting solution $\mathbf{x} = \mathbf{0}, \mathbf{x}_a = \mathbf{b}$ is feasible to the new constraints, it has a very unattractive objective value, namely $M\mathbf{1}^{\mathrm{T}}\mathbf{b}$. Therefore, the simplex method itself will try to drop artificial variables out of the basis, and then continue to find the optimal solution to the original problem.

Since we are interested in the solution of the original problem, after solving it by the simplex method, one of the following two cases may occur:

(i) We found the optimal solution of P(M).

- *The artificial variables are all equal to zero.* In this case, the original problem is feasible and the optimal solution is found.
- *Some artificial variables are positive.* In this case, the original problem is infeasible.

(ii) We found that the problem P(M) has an unbounded solution. Then, the original problem has an unbounded solution.

The big-M method is illustrated by the following numerical example.

Example 2.20. Consider the following linear programming problem:

$$\begin{array}{lrcrcl} \text{Maximize} & 2x_1 & + & x_2 & & \\ \text{Subject to} & x_1 & + & x_2 & \leq & 3 \\ & x_1 & + & 2x_2 & \geq & 4 \\ & 3x_1 & + & x_2 & \geq & 5 \\ & x_1, & & x_2, & \geq & 0 \end{array} \tag{2.45}$$

This example is the same linear programming problem in example 2.19. The slack variable s_1 and the surplus variables s_2, s_3 are introduced and the artificial variables x_{a_1} and x_{a_2} are incorporated in the last two constraints. The modified objective function is $z_{\text{big-M}} = \mathbf{c}^{\mathrm{T}}\mathbf{x} - M\mathbf{1}^{\mathrm{T}}\mathbf{x}_a$, where M is a large positive number. This leads to the following sequences of tableau:

	$z_{\text{big-M}}$	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
$z_{\text{big-M}}$	1	-2	-1	0	0	0	M	M	0
s_1	0	1	1	1	0	0	0	0	3
x_{a_1}	0	1	2	0	-1	0	1	0	4
x_{a_2}	0	3	1	0	0	-1	0	1	5

Since the reduced cost of basic variables will be zero, multiply row 2 and row 3 by $-M$ and add to row 0, we have

	$z_{\text{big-M}}$	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
$z_{\text{big-M}}$	1	$-2-4M$	$-1-3M$	0	M	M	0	0	$-9M$
s_1	0	1	1	1	0	0	0	0	3
x_{a_1}	0	1	2	0	-1	0	1	0	4
x_{a_2}	0	(3)	1	0	0	-1	0	1	5

	$z_{\text{big-M}}$	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
$z_{\text{big-M}}$	1	0	$-\frac{1}{3}-\frac{5M}{3}$	0	M	$-\frac{2}{3}-\frac{M}{3}$	0	$\frac{2}{3}+\frac{4M}{3}$	$\frac{10}{3}-\frac{7M}{3}$
s_1	0	0	2/3	1	0	1/3	0	- 1/3	4/3
x_{a_1}	0	0	(5/3)	0	-1	1/3	1	-1/3	7/3
x_1	0	1	1/3	0	0	-1/3	0	1/3	5/3

	$z_{\text{big-M}}$	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
$z_{\text{big-M}}$	1	0	0	0	$-1/5$	$-3/5$	$\frac{1}{5}+M$	$\frac{3}{5}+M$	19/5
s_1	0	0	0	1	2/5	(1/5)	-2/5	-1/5	2/5
x_2	0	0	1	0	-3/5	1/5	3/5	-1/5	7/5
x_1	0	1	0	0	1/5	-2/5	-1/5	2/5	6/5

	$z_{\text{big-M}}$	x_1	x_2	s_1	s_2	s_3	x_{a_1}	x_{a_2}	RHS
$z_{\text{big-M}}$	1	0	0	3	1	0	$-1+M$	M	5
s_3	0	0	0	5	2	1	-2	-1	2
x_2	0	0	1	-1	-1	0	1	0	1
x_1	0	1	0	2	1	0	-1	0	2

Since $z_j - c_j \geq 0$ for each nonbasic variable, the last tableau gives the optimal solution with the same sequences of points as in Figure 2.1.

Summary of the Initial Basic Feasible Solution

Before picking a basis $\mathbf{B}$, a general linear programming problem needs to be transformed into the following standard form:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.46}$$

where $\mathbf{b} \geq \mathbf{0}$ (if $b_i < 0$, we can multiply the i^{th} row by -1). An initial basis can be picked as follows:

- if $\mathbf{A}$ contains an identity matrix, then an initial basis $\mathbf{B}=\mathbf{I}$ and since $\mathbf{b} \geq \mathbf{0}$, $\mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$, then the simplex method can start.
- Otherwise, artificial variables are introduced with the associated identity matrix and letting $\mathbf{B}=\mathbf{I}$, then we need to force these artificial variables to zero by Two-Phase method or Big-M method.

2.4 Duality

Each linear programming problem called the primal has the associated problem called the dual which maintains all coefficients of the primal problem with different objective function. The number of variables in the primal is equal to the number of constraints in the dual and the number of variables in the dual is equal to the number of constraints in the primal. Moreover, coefficients of objective function in the primal will be on the right-hand side values of the dual and the constraints matrix $\mathbf{A}$ of the primal will be transposed for the dual. The dual linear programming problem possesses many important properties related to the original primal linear programming problem. There are two important forms of duality: the canonical form and the standard form.

Canonical Form of Duality

Suppose that the primal linear programming problem is given in the canonical form:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.47}$$

Then the dual linear programming problem is defined by:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{b^T w} \\ \text{subject to} \quad & \mathbf{A^T w} \geq \mathbf{c} \\ & \mathbf{w} \geq \mathbf{0}. \end{aligned} \tag{2.48}$$

Standard Form of Duality

The primal linear programming problem in the standard form is given below:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.49}$$

Then the dual linear programming problem is defined by:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{b^T w} \\ \text{subject to} \quad & \mathbf{A^T w} \leq \mathbf{c} \\ & \mathbf{w} \quad \text{unrestricted.} \end{aligned} \tag{2.50}$$

In practice, many linear programming problems contain some constraints of the "$\leq$" type, "$\geq$" type or "$=$" type. Additionally, variables may be "≤ 0," "≥ 0," or "unrestricted." From table 1.2, we can convert the primal problem to the dual problem or use the transformation between primal and dual problem as below:

Table 2.1: Relationships Between Primal and Dual Problems

	Maximization Problem		Minimization Problem	
Variables	≥ 0	$\Longleftrightarrow$	$\geq$	Constraints
	≤ 0	$\Longleftrightarrow$	$\leq$	
	Unrestricted	$\Longleftrightarrow$	$=$	
Constraints	$\geq$	$\Longleftrightarrow$	≤ 0	Variables
	$\leq$	$\Longleftrightarrow$	≥ 0	
	$=$	$\Longleftrightarrow$	Unrestricted	

Note that there is exactly one dual variable for each primal constraint and exactly one dual constraint for each primary variable.

Example 2.21. Consider the following linear programming problem:

$$\begin{aligned} \text{Maximize} \quad & 2x_1 - 3x_2 + x_3 \\ \text{subject to} \quad & -2x_1 + x_2 + 5x_3 \leq 3 \\ & 3x_1 + 6x_2 - x_3 \geq 7 \\ & -x_1 + 2x_2 + 4x_3 = -10 \\ & x_1 \geq 0, \quad x_2 \leq 0, \quad x_3 \quad \text{unrestricted.} \end{aligned} \tag{2.51}$$

Then, the dual problem can be written as below:

$$\begin{array}{lrcrcrcrl} \text{Minimize} & & 3w_1 & + & 7w_2 & - & 10w_3 & & \\ \text{subject to} & - & 2w_1 & + & 3w_2 & - & w_3 & \geq & 2 \\ & & w_1 & + & 6w_2 & + & 2w_3 & \leq & -3 \\ & & 5w_1 & - & w_2 & + & 4w_3 & = & 1 \\ & & w_1 & \geq & 0, & w_2 & \leq & 0, & w_3 \quad \text{unrestricted.} \end{array} \tag{2.52}$$

Karush-Kuhn-Tucker (KKT) Optimality Conditions

The primal linear programming problem in the standard form is given below:

$$\begin{array}{lll} \text{(P):} & \text{Maximize} & \mathbf{c^T x} \\ & \text{subject to} & \mathbf{Ax} = \mathbf{b} \\ & & \mathbf{x} \geq \mathbf{0}. \end{array} \tag{2.53}$$

Then the dual linear programming problem can be written by:

$$\begin{array}{lll} \text{(D):} & \text{Minimize} & \mathbf{b^T w} \\ & \text{subject to} & \mathbf{A^T w} \geq \mathbf{c} \end{array} \tag{2.54}$$

The decision variables $\mathbf{w}$ of the dual problem (2.54) are unrestricted in sign. The optimality conditions for a linear programming problem state that a necessary and sufficient condition for $\mathbf{x}^*$ to be the optimal solution is that there exists a vector $\mathbf{w}^*$ such that

$$\begin{array}{ll} 1. & \mathbf{Ax}^* = \mathbf{b}, \mathbf{x}^* \geq \mathbf{0}, \\ 2. & \mathbf{A}^{\mathrm{T}}\mathbf{w}^* \geq \mathbf{c}, \\ 3. & \mathbf{w}^{*\mathrm{T}}(\mathbf{Ax}^* - \mathbf{b}) = \mathbf{0}, \\ & \mathbf{x}^{*\mathrm{T}}(\mathbf{A}^{\mathrm{T}}\mathbf{w}^* - \mathbf{c}) = \mathbf{0}. \end{array} \tag{2.55}$$

Condition 1 requires that the optimal solution $\mathbf{x}^*$ must be feasible to the primal problem while condition 2 requires that the optimal solution $\mathbf{w}^*$ must be feasible to the dual problem. From condition 3 called *complimentary slackness*, we find that $\mathbf{w}^{*\mathrm{T}}\mathbf{Ax}^* = \mathbf{w}^{*\mathrm{T}}\mathbf{b}$ and $\mathbf{x}^{*\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{w}^* = \mathbf{x}^{*\mathrm{T}}\mathbf{c}$, that is $\mathbf{c}^{\mathrm{T}}\mathbf{x}^* = \mathbf{b}^{\mathrm{T}}\mathbf{w}^*$, the optimal objective values of the primal problem is equal to the optimal objective value of the dual problem.

The Fundamental Theorem of Duality

Theorem 2.22. *With regard to the primal and dual linear programming problems, exactly one of the following statements is true:*

(i) *Both possess optimal solutions* $\mathbf{x}^*$ *and* $\mathbf{w}^*$ *with* $\mathbf{c^T x}^* = \mathbf{b^T w}^*$.

(ii) *One problem has an unbounded optimal objective value, in which case another problem must be infeasible.*

(iii) *Both problems are infeasible.*

From (ii) and (iii) of this theorem, if one problem is unbounded then another problem must be infeasible. While one problem is infeasible, another problem can be unbounded or infeasible.

2.5 The Dual Simplex Method

The dual simplex method is a method which solves a dual linear programming problem by using the primal simplex tableau directly. Consider a primal problem (2.53) at any basic feasible solution, it can be written in the following tableau:

	z	$\mathbf{x_B}$	$\mathbf{x_N}$	**RHS**
z	1	0	$\mathbf{c_B^T B^{-1} N - c_N^T}$	$\mathbf{c_B^T B^{-1} b}$
$\mathbf{x_B}$	0	$\mathbf{I}$	$\mathbf{B^{-1} N}$	$\mathbf{B^{-1} b}$

The tableau shows the primal feasible solution if $\mathbf{B^{-1}b} \geq \mathbf{0}$ for $i = 1, 2, ..., m$. If $z_j - c_j \geq 0$ for all $j = 1, 2, ..., n$, then the tableau is optimal. Consider the row zero, $z_j - c_j = \mathbf{c_B^T B^{-1} A}_{:j} - c_j$ for all $j = 1, 2, ..., n$. Define $\mathbf{w^T} = \mathbf{c_B^T B^{-1}}$. Then, we have

$$z_j - c_j = \mathbf{c_B^T B^{-1} A}_{:j} - c_j = \mathbf{w^T A}_{:j} - c_j. \tag{2.56}$$

At the optimal tableau, we have $z_j - c_j \geq 0$ for all $j = 1, 2, ..., n$ that is $\mathbf{w^T A}_{:j} - c_j \geq 0$ for all $j = 1, 2, ..., n$ which implies that $\mathbf{w^T A} \geq \mathbf{c^T}$ or $\mathbf{A^T w} \geq \mathbf{c}$. Therefore, $\mathbf{w^T} = \mathbf{c_B^T B^{-1}}$ is a dual feasible point.

Lemma 2.23. *At optimality of the primal maximization problem in the canonical form(that is, $z_j - c_j \geq 0$ for all j), $\boldsymbol{w}^* = \boldsymbol{c}_{\boldsymbol{b}}^T \boldsymbol{B}^{-1}$ is an optimal solution to the dual problem. Furthermore, $w_i^* = -(z_{n+i} - c_{n+i}) = -z_{n+i}$ for $i = 1, 2, ..., m$.*

Consider the objective value, $z = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}} \mathbf{B}^{-1} \mathbf{b} = \mathbf{w}^{\mathrm{T}} \mathbf{b}$, that is, the primal objective value and the dual objective value are equal. By KKT conditions, this primal basic feasible solution and this dual basic feasible solution are optimal to the primal and dual problem, respectively.

Similarly, if there is $z_j - c_j < 0$ for some j which implies that there is $\mathbf{w}^{\mathrm{T}} \mathbf{A}_{:j} - c_j < 0$, this dual point is infeasible. The primal simplex method then performs until $z_j - c_j \geq 0$ or $\mathbf{w}^{\mathrm{T}} \mathbf{A}_{:j} - c_j \geq 0$ for all $j = 1, 2, ..., n$, that is, it will perform until the dual solution is feasible.

For the dual simplex method, it starts when the dual is feasible that is $z_j - c_j \geq 0$ for all $j = 1, 2, ..., n$ while there exists $\mathbf{B}^{-1} b_i < 0$ for some $i \in 1, 2, ..., m$, that is the primal is infeasible. The dual simplex method is useful when the dual feasible point is found easier than the primal feasible point. On the other hand, some new constraints are added to the optimal tableau. These constraints may be violated by the current optimal solution which cause the infeasibility of primal then the dual simplex can handle this problem. So the dual simplex method is summarized as follows:

Summary of the Dual Simplex Method (Maximization Problem)

INITIALIZATION STEP: Find a basis $\mathbf{B}$ of the primal such that $z_j - c_j = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}} \mathbf{B}^{-1} \mathbf{A}_{:j} - c_j \geq 0$ for all j.
MAIN STEP:

(i) If $\bar{\mathbf{b}} = \mathbf{B}^{-1} \mathbf{b} \geq 0$, then stop; the current solution is optimal. Otherwise, select a pivot row r with $\bar{b}_r < 0$;

$$\bar{b}_r = \min_{1 \leq i \leq m} \{\bar{b}_i\}. \tag{2.57}$$

(ii) If $y_{rk} \geq 0$ for all j, then stop; the dual is unbounded and *the primal is infeasible. The optimal solution is the current basic feasible solution.*

Otherwise, select the pivot column k by the following minimum ratio test:

$$\frac{z_k - c_k}{|y_{rk}|} = \min_{1 \le j \le n} \left\{ \frac{z_j - c_j}{|y_{jk}|} \mid y_{jk} < 0 \right\}. \tag{2.58}$$

(iii) Pivot at y_{rk} and repeat the MAIN STEP.

Example 2.24. Consider the following problem:

$$\begin{array}{lrrrrrrrrr} \text{Maximize} & - & x_1 & - & 3x_2 & - & 2x_3 & & & \\ \text{subject to} & & 1x_1 & - & 2x_2 & + & x_3 & \le & -2 \\ & - & 3x_1 & + & 3x_2 & - & 2x_3 & \le & -3 \\ & & x_1, & & x_2, & & x_3 & \ge & 0 \end{array} \tag{2.59}$$

Solution. Before starting the simplex algorithm, the problem must be in the standard form. By adding slack variables, we get

$$\begin{array}{lrrrrrrrrrrrr} \text{Maximize} & - & x_1 & - & 3x_2 & - & 2x_3 & & & & & & \\ \text{subject to} & & 1x_1 & - & 2x_2 & + & x_3 & + & s_1 & & & = & -2 \\ & - & 3x_1 & + & 3x_2 & - & 2x_3 & & & + & s_2 & = & -3 \\ & & x_1, & & x_2, & & x_3, & & s_1, & & s_2 & \ge & 0 \end{array} \tag{2.60}$$

Put it in the initial tableau, then

	z	x_1	x_2	x_3	s_1	s_2	RHS
z	1	1	3	2	0	0	0
s_1	0	1	-2	1	1	0	-2
s_2	0	(-3)	3	-2	0	1	-3

With this basis, the primal is infeasible while the dual is feasible. Applying the dual simplex method, select a pivot row r with $\bar{b}_r < 0$, we get $\bar{b}_2 = \min\{\bar{b}_1, \bar{b}_2\} = \min\{-2, -3\}$, and the minimum ratio test is computed with $y_{2j} < 0$ for all $j \in \{1, 2, ..., n\}$ and we get $\frac{1}{3} = \min\{\frac{1}{3}, \frac{2}{2}\}$. Then, we pivot at y_{21}

	z	x_1	x_2	x_3	s_1	s_2	RHS
z	1	0	4	4/3	0	1/3	-1
s_1	0	0	(-1)	1/3	1	1/3	-3
x_1	0	1	-1	2/3	0	-1/3	1

In a similar fashion, we pivot at y_{12} then we get,

	z	x_1	x_2	x_3	s_1	s_2	RHS
z	1	0	0	8/3	4	5/3	-13
x_2	0	0	1	-1/3	-1	-1/3	3
x_1	0	1	0	1/3	-1	-2/3	4

Since $\bar{\mathbf{b}} \geq \mathbf{0}$ and $z_j - c_j \geq 0$ for all j, the optimal solution is found with $(x_1^*, x_2^*, x_3^*, s_1^*, s_2^*) = (4, 3, 0, 0, 0)$. Moreover, the optimal solution of the dual is found with $(w_1^*, w_2^*) = (-4, -5/3)$.

2.6 Sensitivity Analysis

Suppose a linear programming problem is solved with the optimal tableau. If some coefficients are changed, we can determine the effect on the new change with respect to the current optimal solution without resolving the problem from the beginning. In this dissertation, the following variations in the problem will be applied in our algorithm.

- Change in the cost vector $\mathbf{c}$.
- Addition of a new constraint.

Change in the Cost Vector

Given the optimal basic feasible solution, if some objective coefficients are changed, the effect of this change will occur in the cost row on the final tableau, that is, the dual problem may be infeasible.

In our algorithm, the objective cost will be changed from c_k to c_k' for some k which x_k is a nonbasic variable. So $\mathbf{c_B}$ is not affected, and $z_j = \mathbf{c_B^T B^{-1} A}_{:j}$ is not changed for all $j \in \mathbf{N}$. Since the current basic feasible solution is the optimal solution of the original problem, $z_k - c_k \geq 0$. We would like to know the sign of $z_k - c_k'$. We can calculate $z_k - c_k'$ by

$$z_k - c_k' = z_k - c_k + c_k - c_k' = (z_k - c_k) + (c_k - c_k') \tag{2.61}$$

That is, it can compute easily by adding $c_k - c'_k$ to the known value $z_k - c_k$. If $z_k - c'_k \geq 0$, then the old solution is still optimal for the new problem. Otherwise, the primal simplex method will be continued by introducing x_k into the basis and performed the standard simplex method.

Adding a New Constraint

Consider the following optimal tableau with a basis $\mathbf{B}$.

	z	$\mathbf{x_B}$	$\mathbf{x_N}$	**RHS**
z	1	0	$\mathbf{c_B^T B^{-1} N - c_N^T}$	$\mathbf{c_B^T B^{-1} b}$
$\mathbf{x_B}$	0	$\mathbf{I}$	$\mathbf{B^{-1} N}$	$\mathbf{B^{-1} b}$

Suppose a new constraint $\mathbf{a}_{m+1}\mathbf{x} \geq b_{m+1}$ is added to the problem. Before adding the new constraint to the tableau, $\mathbf{a}_{m+1}$ is decomposed into $[\mathbf{a}_{m+1_\mathbf{B}}, \mathbf{a}_{m+1_\mathbf{N}}]$ and it is rewritten as

$$\mathbf{a}_{m+1_\mathbf{B}}\mathbf{x_B} + \mathbf{a}_{m+1_\mathbf{N}}\mathbf{x_N} + s_{n+1} = b_{m+1} \tag{2.62}$$

where s_{n+1} is a nonnegative slack variable. Then, add it into the optimal tableau, we get

	z	$\mathbf{x_B}$	$\mathbf{x_N}$	s_{n+1}	**RHS**
z	1	0	$\mathbf{c_B^T B^{-1} N - c_N^T}$	0	$\mathbf{c_B^T B^{-1} b}$
$\mathbf{x_B}$	0	$\mathbf{I}$	$\mathbf{B^{-1} N}$	0	$\mathbf{B^{-1} b}$
s_{n+1}	0	$\mathbf{a}_{m+1_\mathbf{B}}$	$\mathbf{a}_{m+1_\mathbf{N}}$	1	b_{m+1}

We can eliminate $\mathbf{a}_{m+1_\mathbf{B}}$ by multiplying row 1 by $\mathbf{a}_{m+1_\mathbf{B}}$ and subtracting from row 2 gives the following tableau:

	z	$\mathbf{x_B}$	$\mathbf{x_N}$	s_{n+1}	**RHS**
z	1	0	$\mathbf{c_B^T B^{-1} N - c_N^T}$	0	$\mathbf{c_B^T B^{-1} b}$
$\mathbf{x_B}$	0	$\mathbf{I}$	$\mathbf{B^{-1} N}$	0	$\mathbf{B^{-1} b}$
s_{n+1}	0	0	$\mathbf{a}_{m+1_\mathbf{N}} - \mathbf{a}_{m+1_\mathbf{B}}\mathbf{B^{-1} N}$	1	$b_{m+1} - \mathbf{a}_{m+1_\mathbf{B}}\mathbf{B^{-1} b}$

We can obtain the optimal solution by considering the sign of the right hand side in the s_{n+1}. If $b_{m+1} - \mathbf{a}_{m+1_\mathbf{B}}\mathbf{B^{-1} b} \geq 0$, then the current solution is optimal. Otherwise, the dual simplex method is needed to find the optimal solution.

2.7 The Artificial-Variable-Free Techniques

2.7.1 The Criss-Cross Method

To solve a linear programming problem using the standard simplex method, it requires a primal feasible solution to start the algorithm. Similarly, the dual simplex method requires a dual feasible solution to start the algorithm. However, for a linear programming problem, neither a primal nor a dual feasible basic solution could be found easily. So the modified problem with artificial variables is set up to start the algorithm. In 1969, Zionts [10] proposed an algorithm which needed not maintain feasibility and no artificial variables required called the criss-cross algorithm. The algorithm starts by partitioning the problem as follows:

The Partitioned Problem:*

$$\begin{array}{lrcrcr} \text{Maximize} & -\mathbf{c_1^T x_1} & + & \mathbf{c_2^T x_2} & & \\ \text{subject to} & \mathbf{A_{11} x_1} & + & \mathbf{A_{12} x_2} & \leq & -\mathbf{b}_1 \\ & \mathbf{A_{21} x_1} & + & \mathbf{A_{22} x_2} & \leq & \mathbf{b}_2 \\ & \mathbf{x_1}, & & \mathbf{x_2} & \geq & \mathbf{0}, \end{array} \tag{2.63}$$

The Primal Portion of the Problem:

$$\begin{array}{lrcrcr} \text{Maximize} & -\mathbf{c_1^T x_1} & + & \mathbf{c_2^T x_2} & & \\ \text{subject to} & \mathbf{A_{21} x_1} & + & \mathbf{A_{22} x_2} & \leq & \mathbf{b}_2 \\ & \mathbf{x_1}, & & \mathbf{x_2} & \geq & \mathbf{0}, \end{array} \tag{2.64}$$

The Dual Portion of the Problem:

$$\begin{array}{lrcr} \text{Maximize} & -\mathbf{c_1^T x_1} & & \\ \text{subject to} & \mathbf{A_{11} x_1} & \leq & -\mathbf{b}_1 \\ & \mathbf{A_{21} x_1} & \leq & \mathbf{b}_2 \\ & \mathbf{x_1} & \geq & \mathbf{0}, \end{array} \tag{2.65}$$

where $\mathbf{c}_1$ and $\mathbf{x}_1$ are n_1-dimensional column vectors, $\mathbf{c}_2$ and $\mathbf{x}_2$ are n_2-dimensional column vectors, $\mathbf{b}_1$ and $\mathbf{b}_2$ are $m_1$and m_2-dimensional column vectors, respectively, $\mathbf{A}_{11}$, $\mathbf{A}_{12}$, $\mathbf{A}_{21}$ and $\mathbf{A}_{22}$ are $m_1 \times n_1, m_1 \times n_2, m_2 \times n_1$ and $m_2 \times n_2$ matrix; m_1

is the number of primal infeasibilities, n_2 is the number of dual infeasibilities; $m = m_1 + m_2, n = n_1 + n_2$; b_2 and c_2 are non-positive; b_1 and c_1 are strictly positive.

The original problem is partitioned to a primal feasible problem as the problem (2.64) and a dual feasible problem as (2.65). Then, the dual portion of problem is considered, for pivoting selection, dual infeasible constraints are ignored for a dual iteration, but the entire problem is performed for each pivoting. Then, it alternates to consider the primal portion of problem and ignores primal infeasible constraints, then performs the pivoting on the entire problem. The algorithm performs alternating during primal and dual iterations until a primal or dual feasible solution is obtained. Only primal or dual iterations are performed to reach an optimal solution.

Example 2.25. Consider the following problem:

$$\begin{array}{lrcrcrr} \text{Maximize} & & 3x_1 & - & 4x_2 & & \\ \text{subject to} & - & x_1 & - & 2x_2 & \leq & -2 \\ & - & 3x_1 & - & x_2 & \leq & -4 \\ & & x_1 & - & x_2 & \leq & 1 \\ & & x_1 & + & x_2 & \leq & 3 \\ & & x_1, & & x_2 & \geq & 0. \end{array} \tag{2.66}$$

The primal portion of the problem can be expressed as follows:

$$\begin{array}{lrcrcr} \text{Maximize} & 3x_1 & - & 4x_2 & & \\ \text{subject to} & x_1 & - & x_2 & \leq & 1 \\ & x_1 & + & x_2 & \leq & 3 \\ & x_1, & & x_2 & \geq & 0. \end{array} \tag{2.67}$$

Then, the dual portion of the problem can be written as follows:

$$\begin{array}{lrrcr} \text{Maximize} & - & 4x_2 & & \\ \text{subject to} & - & 2x_2 & \leq & -2 \\ & - & x_2 & \leq & -4 \\ & - & x_2 & \leq & 1 \\ & & x_2 & \leq & 3 \\ & & x_2 & \geq & 0. \end{array} \tag{2.68}$$

However, since the entire problem will be performed for each pivoting, it will be put to the following tableau.

	z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
z	1	-3	4	0	0	0	0	0
s_1	0	-1	-2	1	0	0	0	-2
s_2	0	-3	(-1)	0	1	0	0	-4
s_3	0	1	-1	0	0	1	0	1
s_4	0	1	1	0	0	0	1	3

Start with a dual iteration by ignoring the column x_1. So s_2 leaves the basis and x_2 enters the basis. Then, we get

	z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
z	1	-15	0	0	4	0	0	-16
s_1	0	(5)	0	1	-2	0	0	6
x_2	0	3	1	0	-1	0	0	4
s_3	0	4	0	0	-1	1	0	5
s_4	0	-2	0	0	1	0	1	-1

Then, the primal iteration is alternated by ignoring row 4 which x_1 enters the basis and s_1 leaves the basis. After pivoting, we get

	z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
z	1	0	0	3	-2	0	0	2
x_1	0	1	0	1/5	- 2/5	0	0	6/5
x_2	0	0	1	- 3/5	1/5	0	0	2/5
s_3	0	0	0	- 4/5	3/5	1	0	1/5
s_4	0	0	0	2/5	1/5	0	1	7/5

Here, the primal is feasible, so we can perform the standard simplex method. s_3 leaves the basis and s_2 enters the basis. Then, we get

	z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
z	1	0	0	1/3	0	10/3	0	8/3
x_1	0	1	0	- 1/3	0	2/3	0	4/3
x_2	0	0	1	- 1/3	0	- 1/3	0	1/3
s_2	0	0	0	-1 1/3	1	5/3	0	1/3
s_3	0	0	0	2/3	0	- 1/3	1	4/3

The optimal solution is found, and the algorithm stops at the point $(x_1^*, x_2^*) = (4/3, 1/3)$ with the optimal value $z = 8/3$.

Advantage of this method is no initial requirements as to whether a primal or dual feasible solution is available. No artificial variables are required. Moreover, the convergence of this method is proved. However, the criss-cross algorithm does not have the polynomial time complexity and it is not efficient in practice.

2.7.2 Primal Perturbation Simplex Algorithm

In 2000, Pan [14] proposed the algorithm for solving a linear programming problem without introducing artificial variables. If the initial basis was neither primal nor dual feasible, then the cost of the objective function in primal will be perturbed for the dual feasibility and the dual simplex method will be performed until the dual solution is found. Then, the original cost of the objective function in primal will be restored and the primal simplex will be performed.

Consider the following linear programming problem in the standard form:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.69}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $rank(A) = k \leq m < n, \mathbf{b} \in \mathbb{R}^m$, $\mathbf{c}$ and $\mathbf{x}$ are n-dimensional column vectors.

Let an initial basis

$$\mathbf{B} = (\mathbf{A}_{:j_1}, \mathbf{A}_{:j_2}, ..., \mathbf{A}_{:j_k}), \tag{2.70}$$

where $\mathbf{A}_{:j_i}$ is the column of $\mathbf{A}$ corresponding to the basic variable x_i $(i = 1, 2, ..., k)$. Let J_B be the set of indices of basic variables and $\bar{J}_B$ be the remaining set by

$$\bar{J}_B = \{1, 2, ..., n\} - J_B. \tag{2.71}$$

Let $\mathbf{B}^+$ be the Moore - Penrose pseudoinverse [26, 27] of $\mathbf{B}$ and let $\mathbf{c}_B$ be the cost of the objective function corresponding to basic variables. Then, the following partially revised simplex tableau will be used:

$$\begin{array}{c|c} \bar{\mathbf{c}} & \bar{z} \\ \hline \mathbf{B}^+\mathbf{A} & \bar{\mathbf{b}} \end{array} \tag{2.72}$$

where $\bar{z}, \bar{\mathbf{b}}$ and $\bar{\mathbf{c}}$ are determined by

$$\begin{aligned} \bar{z} &= \mathbf{c}_\mathbf{B}^{\mathrm{T}} \mathbf{B}^+ \mathbf{b} \\ \bar{\mathbf{c}} &= \mathbf{c}_\mathbf{B}^{\mathrm{T}} \mathbf{B}^+ \mathbf{A} - \mathbf{c} \\ \bar{\mathbf{b}} &= \mathbf{B}^+ \mathbf{b}. \end{aligned} \tag{2.73}$$

In the tableau (2.72), if both primal and dual are feasible, the following two sets

$$I = \{i \mid \bar{b}_i < 0, i = 1, ..., k\}, \tag{2.74}$$

$$J = \{j \mid \bar{c}_j < 0, j \in \bar{J}_B\}, \tag{2.75}$$

are empty, and the optimal solution is found.

Suppose that the tableau (2.72) is neither primal nor dual feasible. $\bar{c}_j$, for all $j \in J$, will be perturbed to some predetermined number $\delta_j > 0$, then the tableau (2.72) is turned into the following:

$$\begin{array}{c|c} \bar{\mathbf{c}}' & \bar{z} \\ \hline \mathbf{B}^{+}\mathbf{A} & \bar{\mathbf{b}} \end{array} \tag{2.76}$$

where

$$\bar{c}'_j = \begin{cases} \delta_j, & \forall j \in J, \\ \bar{c}_j, & \forall j \in \{1, 2, ..., n\} - J. \end{cases} \tag{2.77}$$

The following perturbation of the problem (2.69) can be written as follows:

$$\begin{aligned} \text{Maximize} \quad & \hat{\mathbf{c}}^{\mathbf{T}}\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.78}$$

where

$$\hat{c}'_j = \begin{cases} \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\mathbf{B}^{+}\mathbf{A}_{:j} - \delta_j, & \forall j \in J, \\ c_j, & \forall j \in \{1, 2, ..., n\} - J. \end{cases} \tag{2.79}$$

Since $c_j > 0$ for all $j \in \{1, 2, ..., n\}$, the tableau (2.76) is dual feasible. So the dual simplex method can start to solve the problem (2.78).

Suppose the following tableau is the optimal tableau of the problem (2.78):

$$\begin{array}{c|c} \tilde{\mathbf{c}}' & \tilde{z}' \\ \hline \tilde{\mathbf{B}}^{+}\mathbf{A} & \tilde{\mathbf{b}} \end{array} \tag{2.80}$$

It means that $\tilde{\mathbf{c}}' \geq \mathbf{0}, \tilde{\mathbf{b}} \geq \mathbf{0}$. Then, a feasible tableau for the original problem (2.69) can be restored from (2.80) by replacing $\tilde{z}$ and $\tilde{c}'$ with

$$\tilde{z} = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\tilde{\mathbf{B}}^{+}\mathbf{b} \tag{2.81}$$

$$\tilde{\mathbf{c}}'_j = \mathbf{c}_{\mathbf{B}}^{\mathrm{T}}\tilde{\mathbf{B}}^{+}\mathbf{A} - \mathbf{c}. \tag{2.82}$$

If $\tilde{\mathbf{c}} \geq \mathbf{0}$, the restored tableau is already optimal to the original problem (2.69). Otherwise, it can be performed by the standard simplex method.

The following theorems and lemmas can guarantee that the solution from the restored tableau (2.80) is an optimal or a feasible point of the original problem.

Lemma 2.26. *Suppose that the problem (2.69) has an optimal solution which is also optimal to (2.78). If any optimal tableau (2.80) of (2.78) is dually non-degenerate, the tableau restored from (2.80) gives the optimal solution to the problem (2.69).*

Lemma 2.27. *Suppose that the problem (2.69) has an optimal basis which is also optimal to (2.78). If any optimal tableau (2.80) of (2.78) is primally and dually non-degenerate, the tableau restored from the (2.80) is optimal to (2.69).*

Lemma 2.28. *Let $\boldsymbol{B}_*$ be an optimal basis of the original problem (2.69) and let $J \cap J_B$ be empty. Then, $\boldsymbol{B}_*$ is also and optimal basis for the problem (2.78).*

Theorem 2.29. *Under the same assumption of Lemma 2.28, the tableau restored from (2.80) gives the optimal solution to the original problem if (2.80) is dually non-degenerate, and even the optimal tableau of it if, in addition, (2.80) is also primally non-degenerate.*

Theorem 2.30. *Under the same assumption of Lemma 2.28, the tableau restored from (2.80) gives an optimal solution to the original program (2.69) if only those components of the relative cost row of (2.72) are changed which correspond to non-basic variables of (2.80), i.e., $J \cap J_B$ is empty.*

Primal Perturbation Simplex Algorithm

The algorithm is separated into two Phases: the dual Phase-1 consists of Steps 2 through 8, the primal Phase-2 consist of Steps 8 through 14. Symbols ia and ic are defined as follows:

$ia = 1$ or 2: proceeding with Phase-1 or Phase-2;

$ic = 0$ or 1: proceeding with relative price row unchanged or changed.

Algorithm A. Let $\mathbf{B}^+ \in \mathbb{R}^{k\times m}$ be the Moore-Penrose pseudoinverse of an initial basis. Given constants $\delta_j = 10^{-6}, j = 1, ..., n$, and a tolerance $\epsilon = 10^{-6}$.

Step 1. Set $ia = 1$ and $ic = 0$.

Step 2. Compute $\bar{z}, \bar{c}$ and $\bar{b}$ by (2.73).

Step 3. If the index set, define by (2.75), is nonempty, set $\bar{c}_j = \delta_j, j \in J$, and $ic = 1$.

Step 4. Determine the row index r such that

$$\bar{b}_r = \min\{\bar{b}_i \mid i = 1, ..., k\} < 0. \tag{2.83}$$

Step 5. If $\bar{b}_r \geq -\epsilon$, then:

(a) stop if $ic = 0$;

(b) restore $\bar{z}$ and $\bar{c}$ by (2.73);

(c) set $ia = 2$, and then go to Step 10.

Step 6. Stop if index set J', defined by

$$J' = \{j \mid \mathbf{B}^+(\mathbf{A}_{:j})_r < 0, j \in \bar{J}_B\} \tag{2.84}$$

is empty.

Step 7. Determine the column index s by

$$\bar{c}'_s/(\mathbf{B}^+\mathbf{A}_{:s})_r = \max\{\bar{c}'_j/(\mathbf{B}^+\mathbf{A}_{:j})_r, j \in J'\}. \tag{2.85}$$

Step 8. Update entire tableau.

Step 9. Go to Step 4 if $ia = 1$.

Step 10. Determine s such that

$$\bar{c}_s = \min\{\bar{c}_j \mid j \in \bar{J}_B\} \tag{2.86}$$

where $\bar{J}_B$ defined as (2.71).

Step 11. Stop if $\bar{c}_s \geq -\epsilon$.

Step 12. Stop if the row index set

$$I' = \{i \mid (\mathbf{B}^+\mathbf{A}_{:s})_i > 0, i = 1, ..., k\} \tag{2.87}$$

is empty.

Step 13. Determine r such that

$$\bar{b}'_r/(\mathbf{B}^+\mathbf{A}_{:s})_r = \max\{\bar{b}'_i/(\mathbf{B}^+\mathbf{A}_{:s})_i, i \in I'\}. \tag{2.88}$$

Step 14. Go to Step 8.

Based on properties of the dual and primal pivoting rules, the following theorem can be stated as:

Theorem 2.31. *Assuming dual non-degeneracy for Phase-1 and primal non-degeneracy for Phase-2, Algorithm A terminates at either*

(i) *Step 5(a) or 11, with the optimal solution of (2.69) reached; or*

(ii) *Step 6, indicating infeasibility of the problem; or*

(iii) *Step 12, indicating upper unboundedness of the problem.*

The author showed the efficiency of the algorithm by computational results of small problems.

Advantages of this method is that no artificial variables are required. Moreover, if we can choose the initial basis which closes to the optimal basis, we may reduce the computational time to solve the problem. However, in this research, the appropriate initial basis was not suggested and computational results were shown to be superior for small problems.

2.7.3 Big-M Free Solution Algorithm

In 2006, Arsham [15, 16] presented the new solution algorithm for solving a general linear programming problem without using artificial variables by relaxing constraints. The algorithm starts with the following linear programming problem:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} \leq \mathbf{a} \\ & \mathbf{Bx} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.89}$$

where $\mathbf{b} \geq \mathbf{0}$ and $\mathbf{a} > \mathbf{0}$, $\mathbf{A}$ and $\mathbf{B}$ are the respective matrices of constraint coefficients, $\mathbf{a}$ and $\mathbf{b}$ are the corresponding RHS vectors (all with appropriate dimensions). If any linear programming problem is not in this form, we can convert

it according to the problem manipulation in Chapter I. Then, the algorithm can be summarized as follows:

Big-M Free Solution Algorithm

Phase I Relax the greater-than ($\geq$) constraints and solve the relaxed problem by the standard simplex at the origin point.

Phase II If the solution satisfies all relaxed constraints. Then stop. Otherwise, if the relaxed problem is unbounded, then the optimal solution is modified for dual feasibility and the most "violated" constraint is appended into the tableau and the dual simplex is used to restore feasibility.

Phase III Restore the original objective function (if needed). Then, the standard simplex is used until the solution is found.

Some numerical examples illustrate the efficiency of the big-M free solution algorithm for small problems.

Example 2.32. Consider the following problem:

$$\begin{array}{lrcrcrcl} \text{Maximize} & 3x_1 & + & x_2 & - & 4x_3 & & \\ \text{subject to} & x_1 & + & x_2 & - & x_3 & = & 1 \\ & & & x_2 & & & \geq & 2 \\ & x_1, & & x_2 & & & \geq & 0. \end{array} \tag{2.90}$$

Since the problem (2.90) is not in form of the system (2.89), we need to convert it as follows:

$$\begin{array}{lrcrcrcl} \text{Maximize} & 3x_1 & + & x_2 & - & 4x_3 & & \\ \text{subject to} & x_1 & + & x_2 & - & x_3 & \leq & 1 \\ & x_1 & + & x_2 & - & x_3 & \geq & 1 \\ & & & x_2 & & & \geq & 2 \\ & x_1 & & x_2 & & & \geq & 0. \end{array} \tag{2.91}$$

Phase I, two of $\geq$ constraints are relaxed as follows:

$$\begin{array}{lrcrcrcl} \text{Maximize} & 3x_1 & + & x_2 & - & 4x_3 & & \\ \text{subject to} & x_1 & + & x_2 & - & x_3 & \leq & 1 \\ & x_1 & & x_2 & & & \geq & 0. \end{array} \tag{2.92}$$

The simplex algorithm can start by adding a slack variable as the following tableau:

	z	x_1	x_2	x_3	s_1	RHS
z	1	-3	-1	4	0	0
s_1	0	(1)	1	-1	1	1

	z	x_1	x_2	x_3	s_1	RHS
z	1	0	2	1	3	3
x_1	0	1	1	-1	1	1

The optimal solution is $(x_1, x_2, x_3) = (1, 0, 0)$. This solution is violated the constraint $x_2 \geq 2$, so this constraint is appended to the tableau as follows:

	z	x_1	x_2	x_3	s_1	s_2	RHS
z	1	0	2	1	3	0	3
x_1	0	1	1	-1	1	0	1
s_2	0	0	(-1)	0	0	1	-2

Perform the dual simplex by entering x_2 into the basis and s_2 leaves the basis. After pivoting, we have

	z	x_1	x_2	x_3	s_1	s_2	RHS
z	1	0	0	1	3	2	-1
x_1	0	1	0	(-1)	1	1	-1
x_2	0	0	1	0	0	-1	2

Enter x_3 into the basis and x_1 leaves the basis as follows:

	z	x_1	x_2	x_3	s_1	s_2	RHS
z	1	1	0	0	4	3	-2
x_3	0	-1	0	1	-1	-1	1
x_2	0	0	1	0	0	-1	2

The solution $(x_1, x_2, x_3) = (0, 2, 1)$ satisfies all constraints, so this solution is the optimal solution of the original problem.

The strength of this algorithm is that no artificial variables are required and it deals with a small fraction of the original constraints.

However, Arsham's paper showed only small examples and lacked a computational result. Moreover, if all constraints which form the optimal solution are "$\geq$" constraints, they will be relaxed. So it will converge very slowly since the algorithm wastes time to solve the relaxed problem which far from the solution. This is the obvious weakness of that algorithm. Moreover, if there are no "$\leq$" constraints, the original positive costs are changed to zero for the dual feasibility and performed the dual simplex which converges slowly.

2.7.4 The Cosine Simplex Method

The simplex algorithm without using artificial variables which is called *the Cosine Simplex Method* was repeatedly proposed in the same year by Corley et. al. [17]. It is similar to Big-M free solution algorithm by Arsham which solved a sequence of relaxed linear programming problems until the optimal solution of the original problem was found. They used the cosine criterion to select the constraint. The algorithm starts by considering the following linear programming problem:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.93}$$

where $\mathbf{x}$ and $\mathbf{c}$ are n-dimensional vectors, $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{b}$ is an m-dimensional vector. The problem (2.93) is assumed that it has an optimal

solution.

Denote $\mathbf{a}_i$ the row i of the matrix $\mathbf{A}$ of the problem (2.93), so constraint i is written as follows:

$$\mathbf{a}_i\mathbf{x} \leq b_i, i = 1, ..., m. \tag{2.94}$$

To guarantee that the problem (2.93) has the optimal solution, a constraint r in (2.94) is assumed for which $a_{rj} > 0, j = l, ..., n$ and $b_r > 0$. Therefore, the problem is bounded. Then, the initial relaxed problem can be written as below:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{a}_r\mathbf{x} \leq b_r \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.95}$$

Obviously, this problem has the optimal solution. Then, define

$$\cos\theta_i = \frac{\mathbf{a}_i{}^{\mathrm{T}} \cdot \mathbf{c}}{\|\mathbf{a}_i\|\|\mathbf{c}\|} \tag{2.96}$$

as the cosine of the angle θ_i between the normal vectors $\mathbf{a}_i$ for constraint i and $\mathbf{c}$ for the objective function. If this constraint is a part of the current relaxed problem, then it is called *a constraint operative*. Otherwise, it is called *a constraint inoperative*. The algorithm can be stated as follows:

The Cosine Simplex Algorithm

Step 0 Compute $\cos\theta_i = \dfrac{\mathbf{a}_i{}^{\mathrm{T}} \cdot \mathbf{c}}{\|\mathbf{a}_i\|\|\mathbf{c}\|}, i = 1, ..., m, i \neq r$ and order the constraints according to decrease $\cos\theta_i$, where ties are broken arbitrarily.

Step 1 Solve the problem (2.95) to obtain $\mathbf{x}^1$. Set $k = 1$.

Step 2 Check the inoperative constraints in decreasing order of $\cos\theta_i$. If the first one is violated by $\mathbf{x}_k$, then go to Step 3. Otherwise, $\mathbf{x}_k$ is the optimal solution of the problem (2.93). Then, stop.

step 3 Set $k = k + 1$. Append the violated constraint to the final tableau of relaxed problem k to obtain relaxed problem $k + 1$. Apply the dual simplex algorithm to obtain a solution $\mathbf{x}_k$. Go to Step 2.

Small problems were shown the efficiency of the cosine simplex algorithm as the following example.

Example 2.33. Consider the problem

$$\begin{array}{lrcrcrcrcl} \text{Maximize} & 4x_1 & + & 5x_2 & + & 9x_3 & + & 11x_4 & & \\ \text{subject to} & 3x_1 & + & 5x_2 & + & 10x_3 & + & 15x_4 & \leq & 100 \\ & x_1 & + & x_2 & + & x_3 & + & x_4 & \leq & 15 \\ & 7x_1 & + & 5x_2 & + & 3x_3 & + & 2x_4 & \leq & 120 \\ & x_1, & & x_2, & & x_3, & & x_4 & \geq & 0. \end{array} \tag{2.97}$$

The following table shows the values of $\cos\theta_i$ up to two decimal digits in Step 0.

Constraint i	$\cos\theta_i$
1	0.99
2	0.93
3	0.70

Each of these constraints bounds the problem. Since $\cos\theta_1$ which associated with constraint 1 is the maximum, the first relaxed problem to be solved is

$$\begin{array}{lrcrcrcrcl} \text{Maximize} & 4x_1 & + & 5x_2 & + & 9x_3 & + & 11x_4 & & \\ \text{subject to} & 3x_1 & + & 5x_2 & + & 10x_3 & + & 15x_4 & \leq & 100 \\ & x_1, & & x_2, & & x_3, & & x_4 & \geq & 0. \end{array} \tag{2.98}$$

Solving this relaxed problem gives the following sequences of tableaus.

	z	x_1	x_2	x_3	x_4	s_1	RHS
z	1	-4	-5	-9	-11	0	0
s_1	0	3	5	10	(15)	1	100

	z	x_1	x_2	x_3	x_4	s_1	RHS
z	1	-9/5	-4/3	-5/3	0	3/4	220/3
x_4	0	(1/5)	1/3	2/3	1	0	20/3

	z	x_1	x_2	x_3	x_4	s_1	RHS
z	1	0	5/3	13/3	9	4/3	400/3
x_1	0	1	5/3	10/3	5	1/3	100/3

The solution of the relaxed problem is obtained at the second iteration with $\mathbf{x}^1 = [100/3, 0, 0, 0]$. Check this point with the inoperative constraint 2 and 3. The constraint 2 is violated by $\mathbf{x}^1$, then it is appended to the last tableau and is solved by the dual simplex method. So we get the following sequences of tableaus:

	z	x_1	x_2	x_3	x_4	s_1	s_2	RHS
z	1	0	5/3	13/3	9	4/3	0	400/3
x_1	0	1	5/3	10/3	5	1/3	0	100/3
s_2	0	0	-2/3	(-7/3)	-4	- 1/3	1	-55/3

	z	x_1	x_2	x_3	x_4	s_1	s_2	RHS
z	1	0	3/7	0	11/7	5/7	13/7	695/7
x_1	0	1	5/7	0	- 5/7	- 1/7	1 3/7	50/7
x_3	0	0	2/7	1	12/7	1/7	- 3/7	55/7

The optimal solution is found with $\mathbf{x}^2 = [50/7, 0, 55/7, 0]$. Check this point with the inoperative constraint 3. The constraint 3 is not violated by $\mathbf{x}^2$. So this solution is optimal to the original problem.

The standard simplex algorithm solves this problem with four tableaus including the initial one.

The computational efficiency of their algorithm was compared with the standard simplex method by counting the constraints which is operated from the first relaxed problem until the optimal solution is found. Therefore, the computation of the cosine simplex is

$$C = \sum_{k=1}^{p} (\#\text{ of tableaus with k constraints}) * k \tag{2.99}$$

where p is the number of constraints in the optimal tableau. For the standard simplex, they computed

$$S = (\#\text{ of tableaus required by the standard simplex for a solution}) * m. \quad (2.100)$$

Therefore, in the example 2.33, $C = 6$, while $S = 12$.

The strength of this algorithm is, at each iteration, the cosine algorithm deal with a small fraction of the original constraints in the simplex computation. In addition, relatively few constraints by the cosine criterion may active at optimality. Moreover, artificial variables are not needed to start the algorithm.

However, the efficiency of this research were shown by small examples and lacked a computational result. In addition, this algorithm could deal with only a feasible and bounded linear programming problem.

According to literature reviews, some theorems are used in our proof of our theorem. Moreover, the simplex method, the dual simplex method and sensitivity analysis will be applied in our algorithms. Efficiency of our algorithms is shown by comparing with Two-Phase method and Arsham's method.

CHAPTER III
ARTIFICIAL-VARIABLE-FREE SIMPLEX METHOD

In this chapter, the algorithm for solving a linear programming problem by the simplex method without using artificial variables is proposed. Our algorithm consists of four important parts, i.e., the classification of constraints, the non-acute constraint relaxation problem (NAR), the transformed NAR problem and the reinsertion of relaxed constraints. These four parts are composed to one algorithm called *Simplex method based on Non-Acute constraint Relaxation (SNAR)* which is proposed to solve either a primal or a dual linear programming problem.

3.1 Preliminaries

Consider a linear programming problem in the following form:

$$\begin{aligned} &\text{Maximize} \quad \mathbf{c^T x} \\ &\text{subject to} \quad \mathbf{Ax} \leq \mathbf{b} \end{aligned} \tag{3.1}$$

where $\mathbf{c}$ is a nonzero vector and $\mathbf{x}$ is an n-dimensional column vector, $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{b}$ is an m-dimensional column vector. Recall the problem manipulation, since any linear programming problem can be converted from one form to another form, all linear programming problems can be converted to this form.

3.1.1 The Classification of Constraints

Since we know that a constraint making the acute angle between its gradient vector to the gradient vector of the objective function may form the extreme point close to the optimal solution, we will separate constraints by their angles between gradient vectors and the gradient vector of the objective function into two collections, i.e., the collection of acute constraints and the collection of non-acute constraints.

Let $\mathbf{A}_{i:}$ be the gradient vector of the constraint i and $\mathbf{c}$ be the gradient vector of the objective function. Let θ_i be the angle between $\mathbf{A}_{i:}$ and $\mathbf{c}$ which can be computed by

$$\theta_i = \arccos \frac{\mathbf{A}_{i:} \cdot \mathbf{c}}{\|\mathbf{A}_{i:}\|\|\mathbf{c}\|}. \tag{3.2}$$

Since all constraints of the problem (3.1) are in the form "$\leq$", the feasible region is on opposite side of the direction of gradient vectors of constraints, see fig

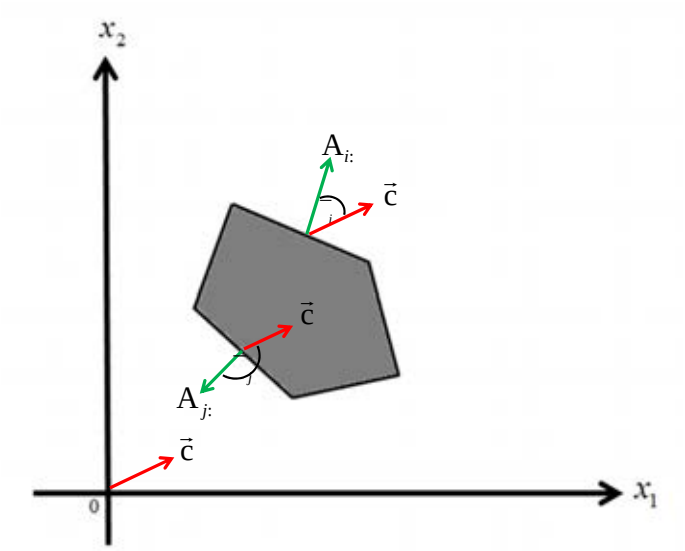


Figure 3.1: Angle between gradient vectors of constraints and the gradient vector of the objective function in $\mathbb{R}^2$

The computation of arccosines appearing in the computation of angles can be avoided by using their cosine values instead. For positive cosine, the constraint will be put in the collection of acute constraints and for negative or zero cosine, the constraint will be put in the collection of non-acute constraints. Since $\|\mathbf{A}_{i:}\|\|\mathbf{c}\| > 0$, we can use the sign of $\mathbf{A}_{i:} \cdot \mathbf{c}$ value to separate the constraint i. Let

$$P = \{i \mid \mathbf{A}_{i:} \cdot \mathbf{c} > 0\} \quad \text{and} \quad N = \{j \mid \mathbf{A}_{j:} \cdot \mathbf{c} \leq 0\} \tag{3.3}$$

where $|P| = m_1, |N| = m_2$ and $m_1 + m_2 = m$, so P is *the collection of index of*

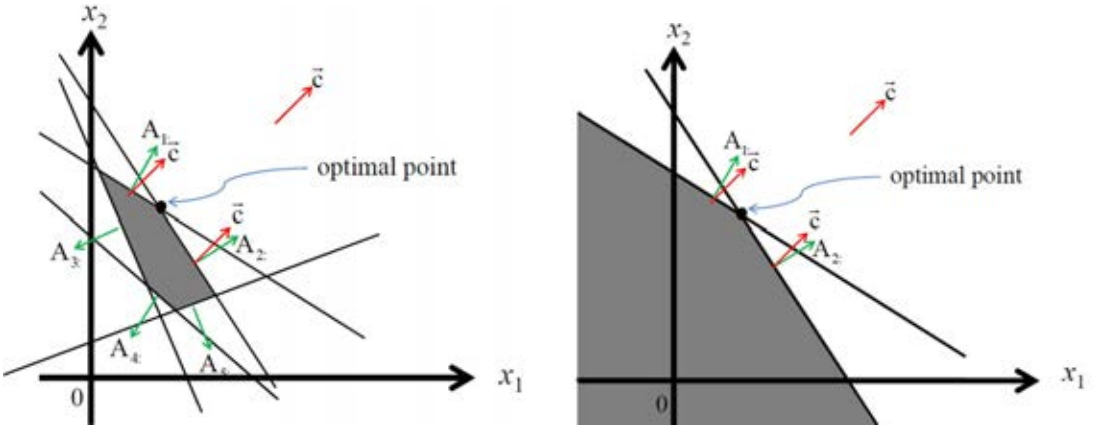
acute constraints and N is *the collection of index of non-acute constraints.*

3.1.2 The Non-Acute Constraint Relaxation Problem

After we separate two collections of constraints, we will relax the original problem by removing constraints from N and it is called *the non-acute constraint relaxation problem (NAR).* So NAR can be written as follows:

$$\begin{aligned} &\text{Maximize} && \mathbf{c^T x} \\ &\text{subject to} && \mathbf{A}_P\mathbf{x} \leq \mathbf{b}_P \end{aligned} \tag{3.4}$$

where $\mathbf{A}_P$ is an $m_1 \times n$ submatrix of constraints corresponding to the collection P, $\mathbf{b}_P$ is m_1-dimensional column vector and $\mathbf{A}_P\mathbf{c} > \mathbf{0}$. The number of constraints of the problem (3.4) is less than or equal to the original problem (3.1) and we can show that the problem (3.4) is always feasible.



(a) The original problem. (b) NAR.

Figure 3.2: Example of the original problem and NAR

Theorem 3.1. *Let* $P^- = \{i \in P \mid b_i < 0\}$ *and* $P^+ = \{i \in P \mid b_i \geq 0\}$. *If* $P^- \neq \varnothing$ *and* $\lambda = \max_{i \in P^-}\{\dfrac{b_i}{-\boldsymbol{A}_{i:} \cdot \boldsymbol{c}}\}$. *Then* $\boldsymbol{x}_0 = -\lambda \boldsymbol{c}$ *is a feasible point of the problem (3.4).*

Proof. Suppose $P^- \neq \varnothing$. Then there is $b_i < 0$ where $i \in P$. Consider

$$\lambda = \max_{i \in P^-}\{\frac{b_i}{-\mathbf{A}_{i:} \cdot \mathbf{c}}\}. \tag{3.5}$$

Since $b_i < 0$ and $\mathbf{A}_{i:} \cdot \mathbf{c} > 0$ for all $i \in P^-$, $-\mathbf{A}_{i:} \cdot \mathbf{c} < 0$, then $\lambda > 0$. For all $k \in P^-$, $\lambda \geq \dfrac{b_k}{-\mathbf{A}_{k:} \cdot \mathbf{c}}$. Therefore $-\lambda \mathbf{A}_{k:} \cdot \mathbf{c} \leq b_k$.

Choose $\mathbf{x}_0 = -\lambda \mathbf{c}$, we get $\mathbf{A}_{k:}\mathbf{x}_0 \leq b_k$ for all $k \in P^-$. For all $l \in P^+$, $-\mathbf{A}_{l:} \cdot \mathbf{c} < 0$ and $b_l \geq 0$. So $\mathbf{A}_{l:}\mathbf{x}_0 = -\lambda \mathbf{A}_{l:} \cdot \mathbf{c} < 0 \leq b_l$ for all $l \in P^+$.

Then $\mathbf{A}_{P:}\mathbf{x}_0 \leq b_P$, i.e., $\mathbf{x}_0$ is a feasible point.

□

Corollary 3.2. *If $P \neq \varnothing$ then the problem (3.4) is always feasible.*

Proof. If $\mathbf{b}_P \geq \mathbf{0}$ then $\mathbf{x}_0 = \mathbf{0}$ is a feasible point. Otherwise, by theorem 3.1, $\mathbf{x}_0 = -\lambda \mathbf{c}$ is always a feasible point.

□

If $P = \varnothing$, the original problem (3.1) can be rewritten as

$$\begin{aligned} &\text{Maximize} && \mathbf{c^T x} \\ &\text{subject to} && \mathbf{A}_N \mathbf{x} \leq \mathbf{b}_N \end{aligned} \tag{3.6}$$

where $\mathbf{A}_N$ is an $m_2 \times n$ submatrix of the constraints corresponding to the collection N, $\mathbf{b}_N$ is m_2-dimensional column vector and $\mathbf{A}_N \mathbf{c} \leq \mathbf{0}$. If $\mathbf{A}_N \mathbf{c} < \mathbf{0}$, then we can show that the original problem has an unbounded optimal solution.

Theorem 3.3. *Let $\boldsymbol{c} \neq \mathbf{0}$, $N_e = \{i \in N \mid \boldsymbol{A}_{i:} \cdot \boldsymbol{c} = 0\}$ and $N_l = \{i \in N \mid \boldsymbol{A}_{i:} \cdot \boldsymbol{c} < 0\}$. If $P = \varnothing$ and $N_e = \varnothing$ then the problem (3.1) is unbounded.*

Proof. Assume $P = \varnothing$ and $N_e = \varnothing$, we get $N = N_l$ and $\mathbf{Ac} = \mathbf{A}_N \mathbf{c} < \mathbf{0}$. Let $X = \{\mathbf{x} \mid \mathbf{A}_N \mathbf{x} \leq \mathbf{b}_N\}$. Show that X is not empty.

Case 1: If $\mathbf{b}_N \geq 0$. Then $\mathbf{x}_0 = 0$ is a feasible point.

Case 2: If there is $b_i < 0$ where $i \in N$. Let $N^- = \{i \in N \mid b_i < 0\} \neq \varnothing$, $N^+ = \{i \in N \mid b_i \geq 0\}$ and

$$\lambda = \max_{i \in N^-} \{\frac{b_i}{\mathbf{A}_{i:} \cdot \mathbf{c}}\}. \tag{3.7}$$

Since $b_i < 0$ and $\mathbf{A}_{i:} \cdot \mathbf{c} < 0$, $\lambda > 0$. For all $k \in N^-$, $\lambda \geq \dfrac{b_k}{-\mathbf{A}_{k:} \cdot \mathbf{c}}$.

By choosing $\mathbf{x}_0 = \lambda\mathbf{c}$, we get $\mathbf{A}_{k:}\mathbf{x}_0 = \lambda\mathbf{A}_{k:}\mathbf{c} \leq b_k$. For all $l \in N^+$, $\mathbf{A}_{l:}\mathbf{x}_0 = \lambda\mathbf{A}_{l:} \cdot \mathbf{c} \leq 0 \leq b_l$. Then $\mathbf{A}_{N:}\mathbf{x}_0 \leq b_N$, i.e., $\mathbf{x}_0$ is a feasible point. So X is not empty.

We will show that $\mathbf{c}$ is a recession direction [25]. For all $\alpha > 0$ and $\mathbf{x}_0 + \alpha\mathbf{c} \in X$, $\mathbf{A}_N(\mathbf{x}_0 + \alpha\mathbf{c}) = \mathbf{A}_N\mathbf{x}_0 + \alpha\mathbf{A}_N\mathbf{c} \leq \mathbf{A}_N\mathbf{x}_0 \leq \mathbf{b}_N$. Therefore $\mathbf{c}$ is a recession direction of X.

For a nonzero vector $\mathbf{c}$, $\mathbf{c^T}(\mathbf{x_0} + \alpha\mathbf{c}) = \mathbf{c^T}\mathbf{x_0} + \alpha\mathbf{c^T}\mathbf{c}$. Since $\mathbf{c^T}\mathbf{c} > \mathbf{0}$, $\mathbf{c^T}\mathbf{x_0} + \alpha\mathbf{c^T}\mathbf{c} \to \infty$ as $\alpha \to \infty$. Therefore the problem (3.1) is unbounded.

□

Consequently, if all constraints make obtuse angles between its gradient vector to the gradient vector of the objective function, then the problem is unbounded.

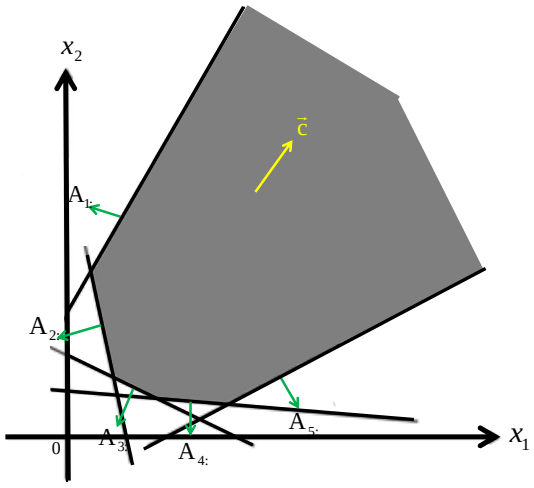


Figure 3.3: Example of the unbounded problem

It should be noted that, Pan proposed the similar theorem [14] to theorem 3.3 where $\mathbf{c} > 0$.

3.1.3 The Transformed NAR Problem

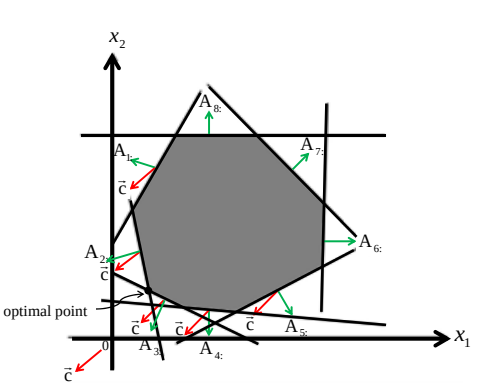
After non-acute constraints are relaxed, we will use the simplex algorithm to solve NAR (3.4). From corollary 3.2, NAR is always feasible. If $\mathbf{b}_P \geq 0$ then $\mathbf{x_0} = \mathbf{0}$ is a feasible point and we can start the simplex algorithm by adding slack variables. Otherwise, $\mathbf{x}_0 = -\lambda\mathbf{c}$ is a feasible point. We will relocate the problem so that $\mathbf{0}$ is the initial basic feasible solution using $\mathbf{x}' = \mathbf{x} - \mathbf{x}_0$.

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x'} + \mathbf{c^T x_0} \\ \text{subject to} \quad & \mathbf{A}_P \mathbf{x'} \leq \mathbf{b}_P - \mathbf{A}_P \mathbf{x}_0 \end{aligned} \tag{3.8}$$

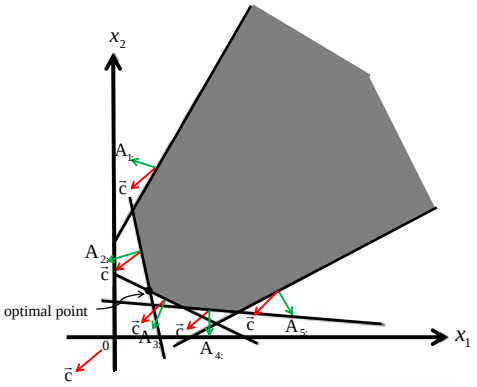
Constraints of the problem (3.6) will be transformed also to $\mathbf{A}_N \mathbf{x'} \leq \mathbf{b}_N - \mathbf{A}_N \mathbf{x}_0$. So the transformed problem can be written as

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x'} + \mathbf{c^T x_0} \\ \text{subject to} \quad & \mathbf{A x'} \leq \mathbf{b} - \mathbf{A x}_0 \end{aligned} \tag{3.9}$$

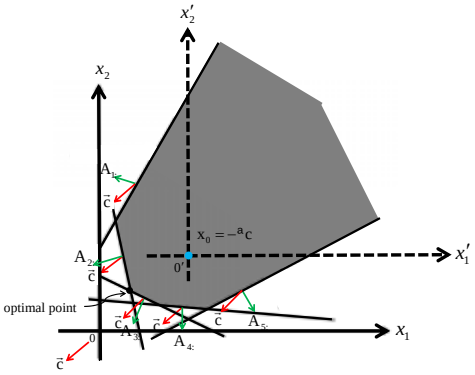
This problem is called "*the transformed NAR problem*". If the transformed NAR problem is infeasible or unbounded then the original problem will be infeasible or unbounded, respectively. If the optimal solution of the transformed problem ($\mathbf{x'}^*$) is found then the optimal solution ($\mathbf{x}^*$) of the original problem will be found by computing $\mathbf{x}^* = \mathbf{x'}^* + \mathbf{x_0}$.



(a) Original problem.



(b) NAR.



(c) Transformed NAR problem.

Figure 3.4: Example of the original, NAR and transformed NAR problems

3.1.4 The Reinsertion of Relaxed Constraints

After transformed NAR problem was solved, relaxed constraints will be reinserted to transformed NAR. Recall the sensitivity analysis, we can add a new constraint to the optimal tableau and can analyse the effect on the problem case by case, without having to resolve the problem from the beginning.

Consider the problem (3.8), $\mathbf{x}_0$ is a feasible point. So $\mathbf{A}_P\mathbf{x}_0 \leq \mathbf{b}_P$ and $\mathbf{b}_P - \mathbf{A}_P\mathbf{x}_0 \geq \mathbf{0}$. Then slack variables are added to transform inequality constraints to equality constraints. So we have

$$\begin{array}{lrcll} \text{Maximize} & \mathbf{c^T x'} & + & \mathbf{c^T x_0} & \\ \text{subject to} & \mathbf{A}_P\mathbf{x'} & + & \mathbf{s} & = \mathbf{b}_P - \mathbf{A}_P\mathbf{x}_0 \\ & & & \mathbf{s} & \geq 0 \end{array} \tag{3.10}$$

Then the simplex algorithm can start to solve the problem without using artificial variables. However, if unrestricted variables exist, they will be transformed by letting $\mathbf{x'} = \mathbf{x}^+ - \mathbf{x}^-$ to get

$$\begin{array}{ll} \text{Maximize} & \mathbf{c^T x}^+ - \mathbf{c^T x}^- + \mathbf{c^T x_0} \\ \text{subject to} & \mathbf{A}_P\mathbf{x}^+ - \mathbf{A}_P\mathbf{x}^- + \mathbf{s} = \mathbf{b}_P - \mathbf{A}_P\mathbf{x}_0 \\ & \mathbf{s}, \mathbf{x}^+, \mathbf{x}^- \geq 0 \end{array} \tag{3.11}$$

Let $z = \mathbf{c^T x}^+ - \mathbf{c^T x}^- + \mathbf{c^T x_0}$. So the initial tableau of the problem (3.11) can be shown as follow:

	z	$\mathbf{x}^+$	$\mathbf{x}^-$	$\mathbf{s}$	**RHS**
z	1	$-\mathbf{c}^{\mathrm{T}}$	$\mathbf{c}^{\mathrm{T}}$	0	$\mathbf{c^T x_0}$
$\mathbf{s}$	0	$\mathbf{A}_P$	$-\mathbf{A}_P$	$\mathbf{I}_{m_1}$	$\hat{\mathbf{b}}_P = \mathbf{b}_P - \mathbf{A}_P\mathbf{x}_0$

where $\mathbf{I}_{m_1}$ is an $m_1 \times m_1$ identity matrix. Since the problem (3.10) is always feasible, the solution can be one of two cases: optimal or unbounded solution.

The Optimal Solution Case

After we found the optimal solution of the problem (3.11), then we will add constraints from N into the problem (3.11). For the transformed problem, the right hand side of N will be changed as $\hat{\mathbf{b}}_N = \mathbf{b}_N - \mathbf{A}_N\mathbf{x}_0$. Then we add the slack $\mathbf{s}_N$ to the transformed problem as $\mathbf{A}_N\mathbf{x}^+ - \mathbf{A}_N\mathbf{x}^- + \mathbf{s}_N = \hat{\mathbf{b}}_N$.

Let $\mathbf{B}_{P^*}$ and $\mathbf{N}_{P^*}$ are the optimal basis and the associated nonbasic matrix of NAR (3.11), respectively. The corresponding tableau of the non-acute constraint relaxation is as follows:

	z	$\mathbf{x}_{\mathbf{B}_{P^*}}$	$\mathbf{x}_{\mathbf{N}_{P^*}}$	**RHS**
z	1	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_{P^*}}\mathbf{B}^{-1}_{P^*}\mathbf{N}_{P^*} - \mathbf{c}^{\mathrm{T}}_{\mathbf{N}_{P^*}}$	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_{P^*}}\mathbf{B}^{-1}_{P^*}\hat{\mathbf{b}}_P + \mathbf{c}^{\mathbf{T}}\mathbf{x_0}$
$\mathbf{x}_{\mathbf{B}_{P^*}}$	0	$\mathbf{I}_{m_1}$	$\mathbf{B}^{-1}_{P^*}\mathbf{N}_{P^*}$	$\mathbf{B}^{-1}_{P^*}\hat{\mathbf{b}}_P$

Let $\hat{\mathbf{A}}_N = [\mathbf{A}_N, -\mathbf{A}_N]$. The constraint $\mathbf{A}_N\mathbf{x}^+ - \mathbf{A}_N\mathbf{x}^- + \mathbf{s}_N = \hat{\mathbf{b}}_N$ is rewritten by:

$$\mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{x}_{\mathbf{B}_{P^*}} + \mathbf{A}_{N\mathbf{N}_{P^*}}\mathbf{x}_{\mathbf{N}_{P^*}} + \mathbf{s}_N = \hat{\mathbf{b}}_N \tag{3.12}$$

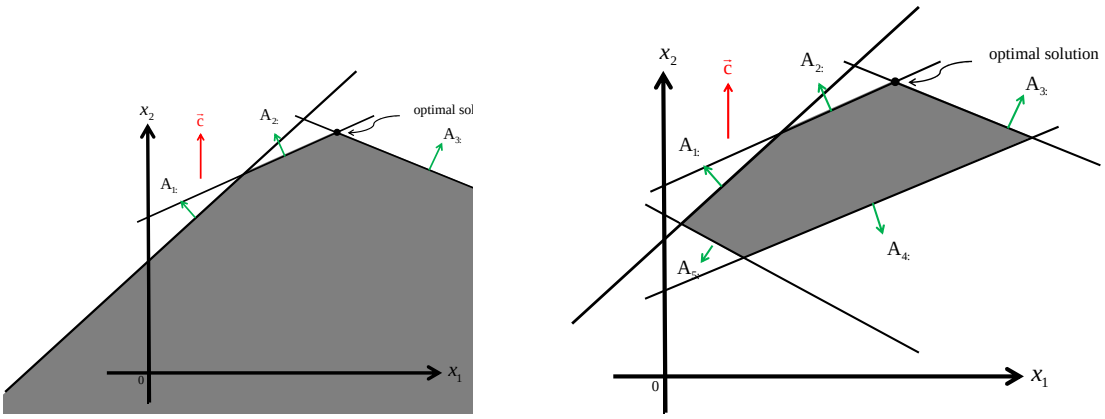
where $\hat{\mathbf{A}}_N = [\mathbf{A}_{N\mathbf{B}_{P^*}}, \mathbf{A}_{N\mathbf{N}_{P^*}}]$ are rearranged by basic and nonbasic columns. After adding constraints (3.12) into tableau, we get

	z	$\mathbf{x}_{\mathbf{B}_{P^*}}$	$\mathbf{x}_{\mathbf{N}_{P^*}}$	$\mathbf{s}_N$	**RHS**
z	1	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_{P^*}}\mathbf{B}^{-1}_{P^*}\mathbf{N}_{P^*} - \mathbf{c}^{\mathrm{T}}_{\mathbf{N}_{P^*}}$	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_{P^*}}\mathbf{B}^{-1}_{P^*}\hat{\mathbf{b}}_P + \mathbf{c}^{\mathbf{T}}\mathbf{x_0}$
$\mathbf{x}_{\mathbf{B}_{P^*}}$	0	$\mathbf{I}_{m_1}$	$\mathbf{B}^{-1}_{P^*}\mathbf{N}_{P^*}$	0	$\mathbf{B}_{P^*}{}^{-1}\hat{\mathbf{b}}_P$
$\mathbf{s}_N$	0	$\mathbf{A}_{N\mathbf{B}_{P^*}}$	$\mathbf{A}_{N\mathbf{N}_{P^*}}$	$\mathbf{I}_{m_2}$	$\hat{\mathbf{b}}_N$

where $\mathbf{I}_{m_2}$ is an $m_2 \times m_2$ identity matrix. We can eliminate $\mathbf{A}_{N\mathbf{B}_{P^*}}$ by multiplying row 1 by $\mathbf{A}_{N\mathbf{B}_{P^*}}$ and subtracting from the row 2 gives the following tableau:

	z	$\mathbf{x}_{\mathbf{B}_{P^*}}$	$\mathbf{x}_{\mathbf{N}_{P^*}}$	$\mathbf{s}_N$	**RHS**
z	1	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_{P^*}}\mathbf{B}^{-1}_{P^*}\mathbf{N}_{P^*} - \mathbf{c}^{\mathrm{T}}_{\mathbf{N}_{P^*}}$	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_{P^*}}\mathbf{B}^{-1}_{P^*}\hat{\mathbf{b}}_P + \mathbf{c}^{\mathbf{T}}\mathbf{x_0}$
$\mathbf{x}_{\hat{\mathbf{B}}_{P^*}}$	0	$\mathbf{I}_{m_1}$	$\mathbf{B}^{-1}_{P^*}\mathbf{N}_{P^*}$	0	$\mathbf{B}^{-1}_{P^*}\hat{\mathbf{b}}_P$
$\mathbf{s}_N$	0	0	$\mathbf{A}_{N\mathbf{N}_{P^*}} - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}^{-1}_{P^*}\mathbf{N}_{P^*}$	$\mathbf{I}_{m_2}$	$\hat{\mathbf{b}}_N - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}^{-1}_{P^*}\hat{\mathbf{b}}_P$

Since the optimal solution is found, the dual problem is feasible. We can obtain the optimal solution by considering the feasibility of primal feasible after the non-acute constraints are reinserted. That is considering the sign of the right hand side in the $\mathbf{s}_N$. If $\hat{\mathbf{b}}_N - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}_{P^*}^{-1}\hat{\mathbf{b}}_P \geq \mathbf{0}$, then the current solution is optimal.

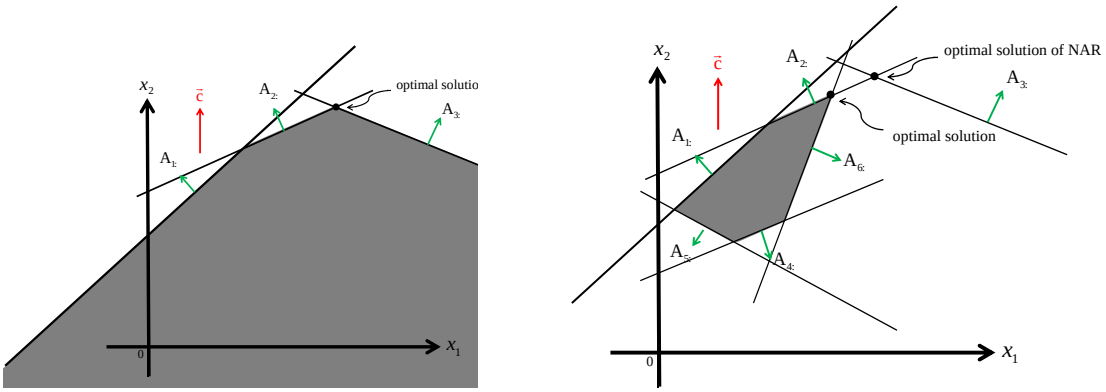


(a) NAR is optimal. (b) Non-acute constraints are added.

Figure 3.5: Example of the optimal solution found from NAR

From figure 3.5b, after the non-acute constraints are reinserted, the optimal solution from NAR satisfies all constraint, that is, primal is feasible. Therefore, this optimal solution is the optimal solution of the original problem.

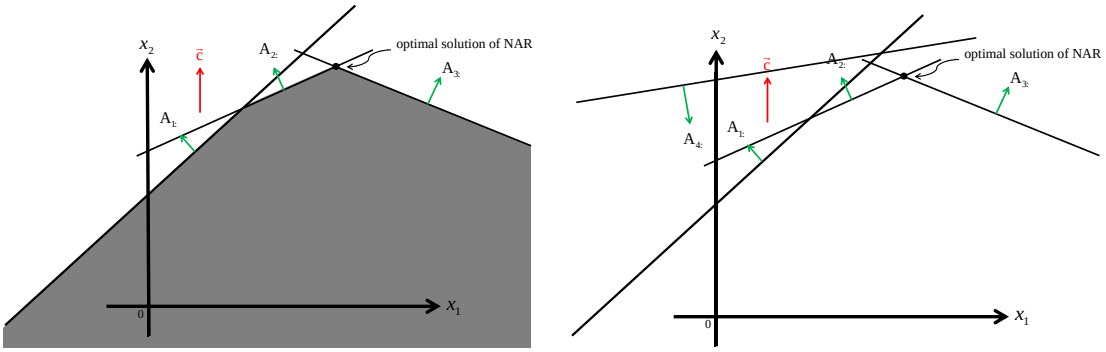
Otherwise, if $\hat{\mathbf{b}}_N - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}_{P^*}^{-1}\hat{\mathbf{b}}_P \ngeq \mathbf{0}$, that is, primal is infeasible, then the dual simplex method is needed to find the optimal solution. Then we can conclude that if we find the optimal solution by the dual simplex method using Dantzig's rule, the value of the right hand side in the optimal tableau is the optimal solution of the transformed NAR problem. Otherwise, if the dual is unbounded, we can conclude that the original problem is infeasible.



(a) NAR is optimal. (b) Non-acute constraints are added.

Figure 3.6: Example of the optimal solution from NAR is infeasible

From figure 3.6b, after non-acute constraints are reinserted, the optimal solution from NAR does not satisfy the sixth constraint, that is, primal is infeasible at the sixth constraint. We will use the dual simplex method to move to the optimal solution.



(a) NAR is optimal. (b) Non-acute constraints are added.

Figure 3.7: Example of the original problem is infeasible

From figure 3.7b, after non-acute constraints are reinserted, the optimal solution from NAR does not satisfy the fourth constraint. We will use the dual simplex method which can detect the primal infeasibility by the dual unbounded.

The Unbounded Case

Let $\mathbf{B}_P$ be the basis and $\mathbf{N}_P$ be the associated nonbasic matrix of NAR. The corresponding tableau is as follows:

	z	$\mathbf{x_{B}}_P$	$\mathbf{x_{N}}_P$	**RHS**
z	1	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_P}\mathbf{B}_P^{-1}\mathbf{N}_P - \mathbf{c}^{\mathrm{T}}_{\mathbf{N}_P}$	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_P}\mathbf{B}_P^{-1}\hat{\mathbf{b}}_P + \mathbf{c^T x_0}$
$\mathbf{x}_{\mathbf{B}_P}$	0	$\mathbf{I}_{m_1}$	$\mathbf{B}_P^{-1}\mathbf{N}_P$	$\mathbf{B}_P^{-1}\hat{\mathbf{b}}_P$

The unbounded solution will occur when some components of $\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_P}\mathbf{B}_P^{-1}\mathbf{N}_P - \mathbf{c}^{\mathrm{T}}_{\mathbf{N}_P}$ are negative. If the non-acute constraint relaxation problem is unbounded, it means that there is $z_j - c_j < 0$ and $\mathbf{y}_j \leq \mathbf{0}$ where $\mathbf{y}_j^{\mathrm{T}} = [a'_{1j}, a'_{2j}, ..., a'_{m_1 j}]^{\mathrm{T}} = \mathbf{B}_P^{-1}\mathbf{N}_{P:j}$. We will find the solution of the transformed problem by adding each constraint from N into the current relaxed tableau.

Let $\mathbf{A}^{\mathrm{T}}_{l:} = [a_{l_{\mathbf{B}_P}}, a_{l_{\mathbf{N}_P}}]^{\mathrm{T}}$ be the coefficient of the constraint l which is rearranged by basic and nonbasic columns where $l \in N$. We will add the constraint $a_{l_{\mathbf{B}_P}}\mathbf{x}_{\mathbf{B}_P} + a_{l_{\mathbf{N}_P}}\mathbf{x}_{\mathbf{N}_P} + s_l = b_l - \mathbf{A}_{l:}\mathbf{x}_0$ into the tableau as follows:

	z	$\mathbf{x_{B}}_P$	$\mathbf{x_{N}}_P$	s_l	**RHS**
z	1	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_P}\mathbf{B}_P^{-1}\mathbf{N}_P - \mathbf{c}^{\mathrm{T}}_{\mathbf{N}_P}$	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_P}\mathbf{B}_P^{-1}\hat{\mathbf{b}}_P + \mathbf{c^T x_0}$
$\mathbf{x}_{\mathbf{B}_P}$	0	$\mathbf{I}_{m_1}$	$\mathbf{B}_P^{-1}\mathbf{N}_P$	**0**	$\mathbf{B}_P^{-1}\hat{\mathbf{b}}_P$
s_l	0	$a_{l_{\mathbf{B}_P}}$	$a_{l_{\mathbf{N}_P}}$	1	$b_l - \mathbf{A}_{l:}\mathbf{x}_0$

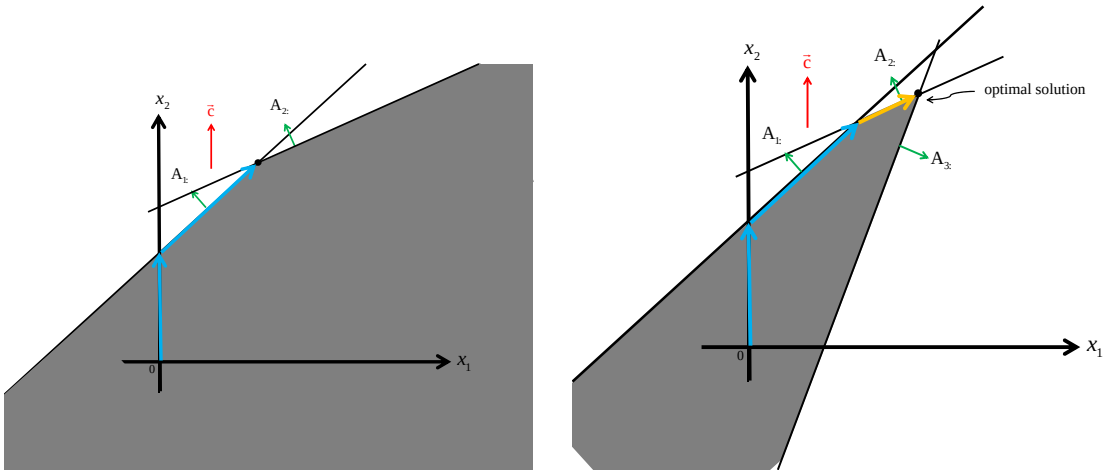
We can eliminate $a_{l_{\mathbf{B}_P}}$ by multiplying the second row by $a_{l_{\mathbf{B}_P}}$ and subtracting from the third row gives the following tableau:

	z	$\mathbf{x_{B}}_P$	$\mathbf{x_{N}}_P$	s_l	**RHS**
z	1	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_P}\mathbf{B}_P^{-1}\mathbf{N}_P - \mathbf{c}^{\mathrm{T}}_{\mathbf{N}_P}$	0	$\mathbf{c}^{\mathrm{T}}_{\mathbf{B}_P}\mathbf{B}_P^{-1}\hat{\mathbf{b}}_P + \mathbf{c^T x_0}$
$\mathbf{x}_{\mathbf{B}_P}$	0	$\mathbf{I}_{m_1}$	$\mathbf{B}_P^{-1}\mathbf{N}_P$	**0**	$\mathbf{B}_P^{-1}\hat{\mathbf{b}}_P$
s_l	0	0	$a_{l_{\mathbf{N}_P}} - a_{l_{\mathbf{B}_P}}\mathbf{B}_P^{-1}\mathbf{N}_P$	1	$b'_l = (b_l - \mathbf{A}_{l:}\mathbf{x}_0) - a_{l_{\mathbf{B}_P}}\mathbf{B}_P^{-1}\hat{\mathbf{b}}_P$

Let a'_{lj} be the coefficient of the constraint l in column j after the elimination. Then the solution can be in one of these two cases.

(i) If $b'_l \geq 0$, that is, the primal is still feasible, then we will consider the value of a'_{lj}.

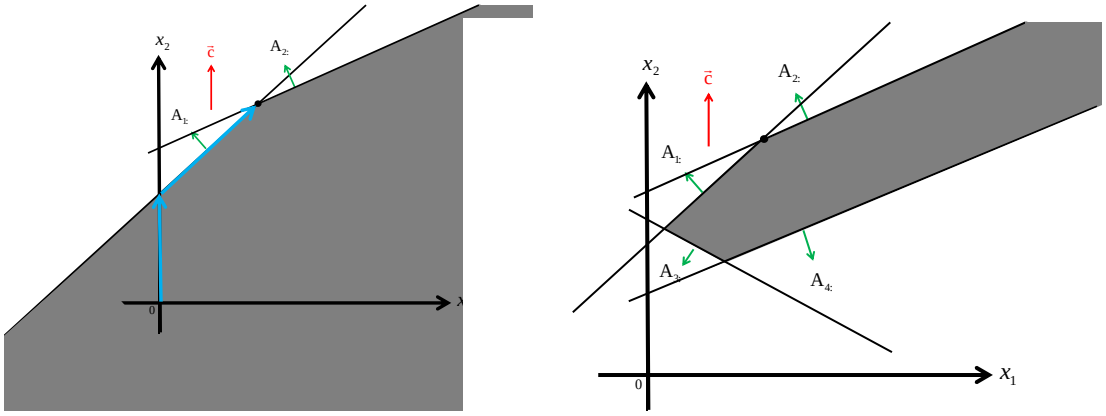
- If $a'_{lj} > 0$, we apply the primal simplex pivoting at a'_{lj}.
- If $a'_{lj} \leq 0$, the transformed problem is unbounded.



(a) NAR is unbounded. (b) Non-acute constraint is added.

Figure 3.8: Example of the original problem is optimal

From the figure 3.8b, the third constraint is valid and then we can pivot at a_{3j}.



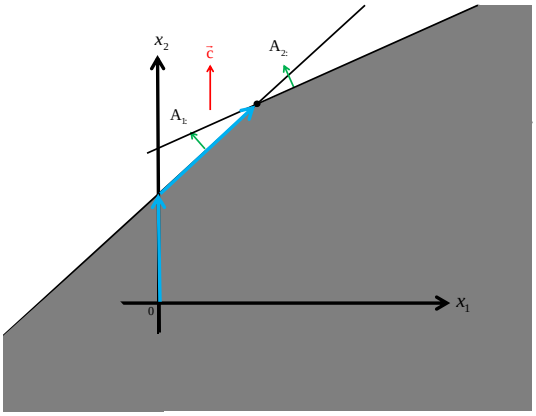
(a) NAR is unbounded. (b) Non-acute constraints are added.

Figure 3.9: Example of the original problem is unbounded

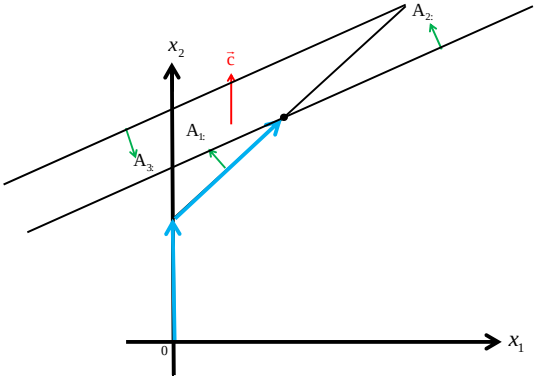
From the figure 3.9b, after non-acute constraints were added, the primal is still feasible and can not pivot at a_{3j} and a_{4j}. So the original problem is unbounded.

(ii) If $b'_l < 0$, then both primal and dual solutions are infeasible at the current iteration because of $z_j - c_j < 0$. We use the technique from Pan [14]. Pan's

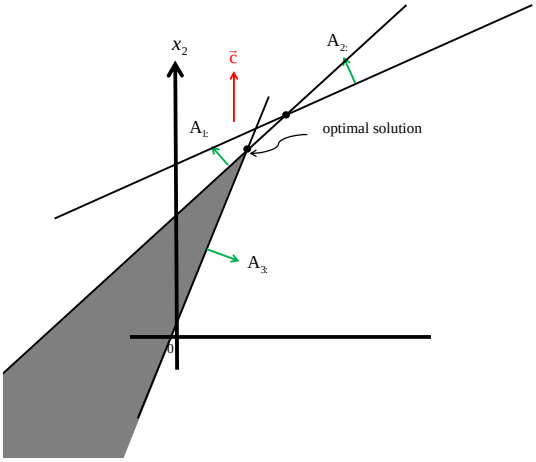
method perturbs $z_j - c_j < 0$ to a positive value to obtain the dual feasible and then perform the dual simplex. After the optimal solution is found, the original $z_j - c_j$ will be restored and the primal simplex is used. However, if the dual problem is unbounded, then the original problem is infeasible.



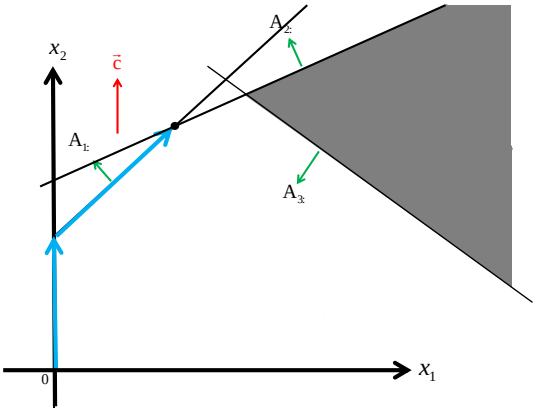
(a) NAR is unbounded.



(b) The reinserted problem is infeasible.



(c) The reinserted problem is o



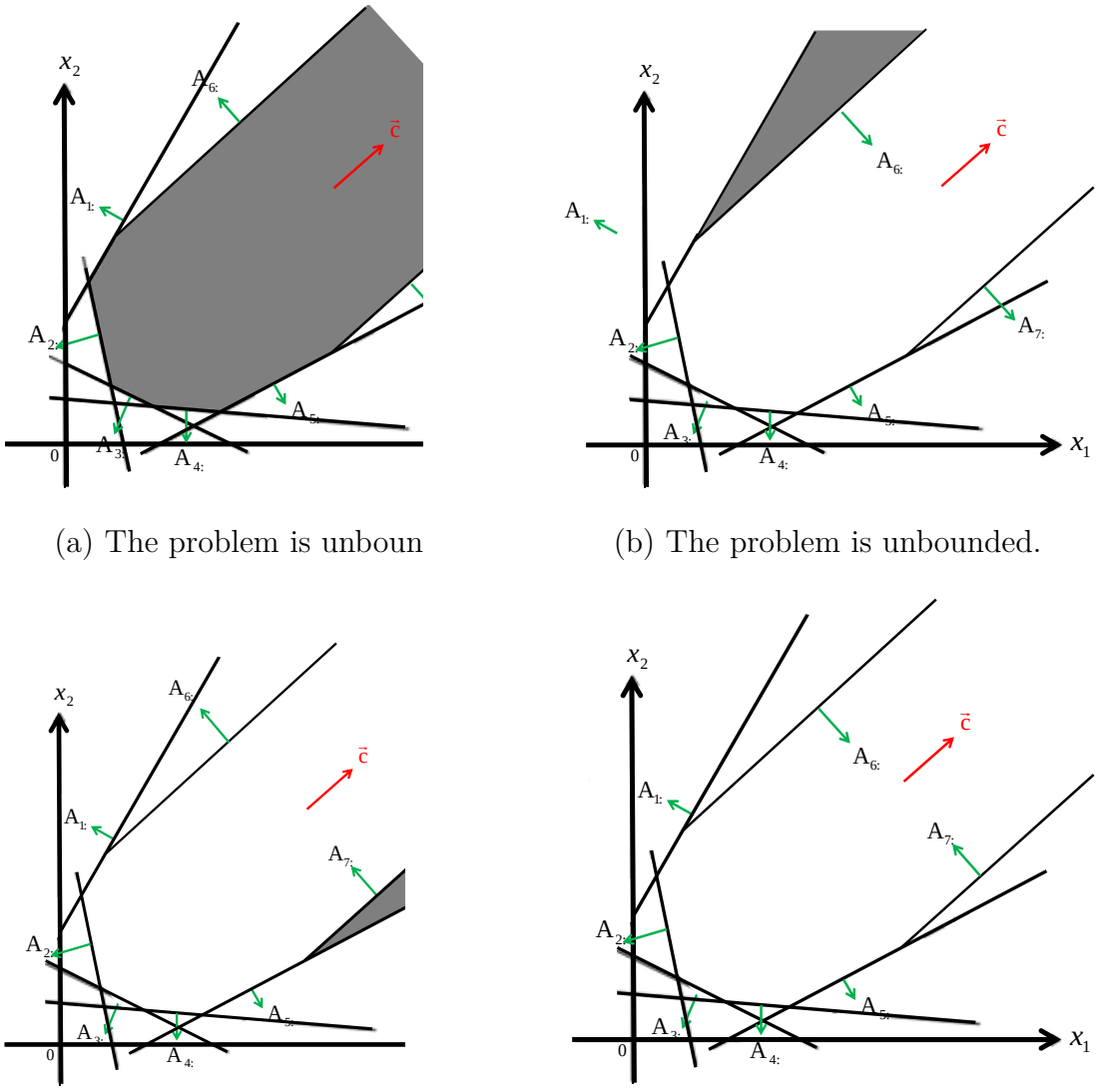
(d) The reinserted problem is unbounded.

Figure 3.10: Some cases of the unbounded of NAR after adding

The Empty Set of P

If $P \neq \varnothing$, then NAR will be solved and the solution can be as the above explanation. On the other hands, if $P = \varnothing$ and $N_e = \varnothing$, then the problem is unbounded as theorem (3.3).

In other cases, such as, if $P = \varnothing$, $N_l \neq \varnothing$ and $N_e \neq \varnothing$, then some possible cases can occur as figure 3.11.



(a) The problem is unboun

(b) The problem is unbounded.

(c) The problem is unbou

(d) The problem is infeasible.

Figure 3.11: Some cases of $P = \varnothing$, $N_l \neq \varnothing$ and $N_e \neq \varnothing$ in $\mathbb{R}^2$

The solution can be either infeasible or unbounded. Therefore, we will relax constraints from N_e and then the relaxed problem can be rewritten as follows:

$$\begin{aligned} &\text{Maximize} && \mathbf{c^T x} \\ &\text{subject to} && \mathbf{A}_{N_l}\mathbf{x} \leq \mathbf{b}_{N_l} \end{aligned} \tag{3.13}$$

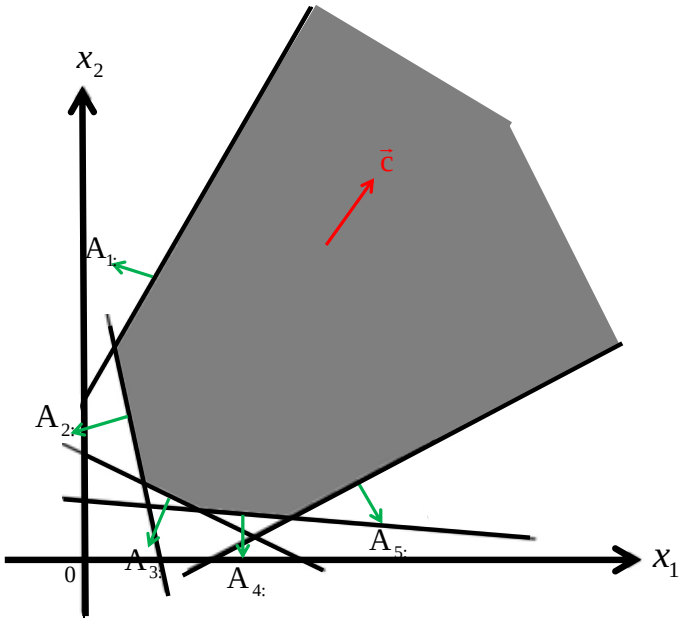


Figure 3.12: Example of the N_e constraint relaxation for $P = \varnothing$ and $N_l \neq \varnothing$

We can choose

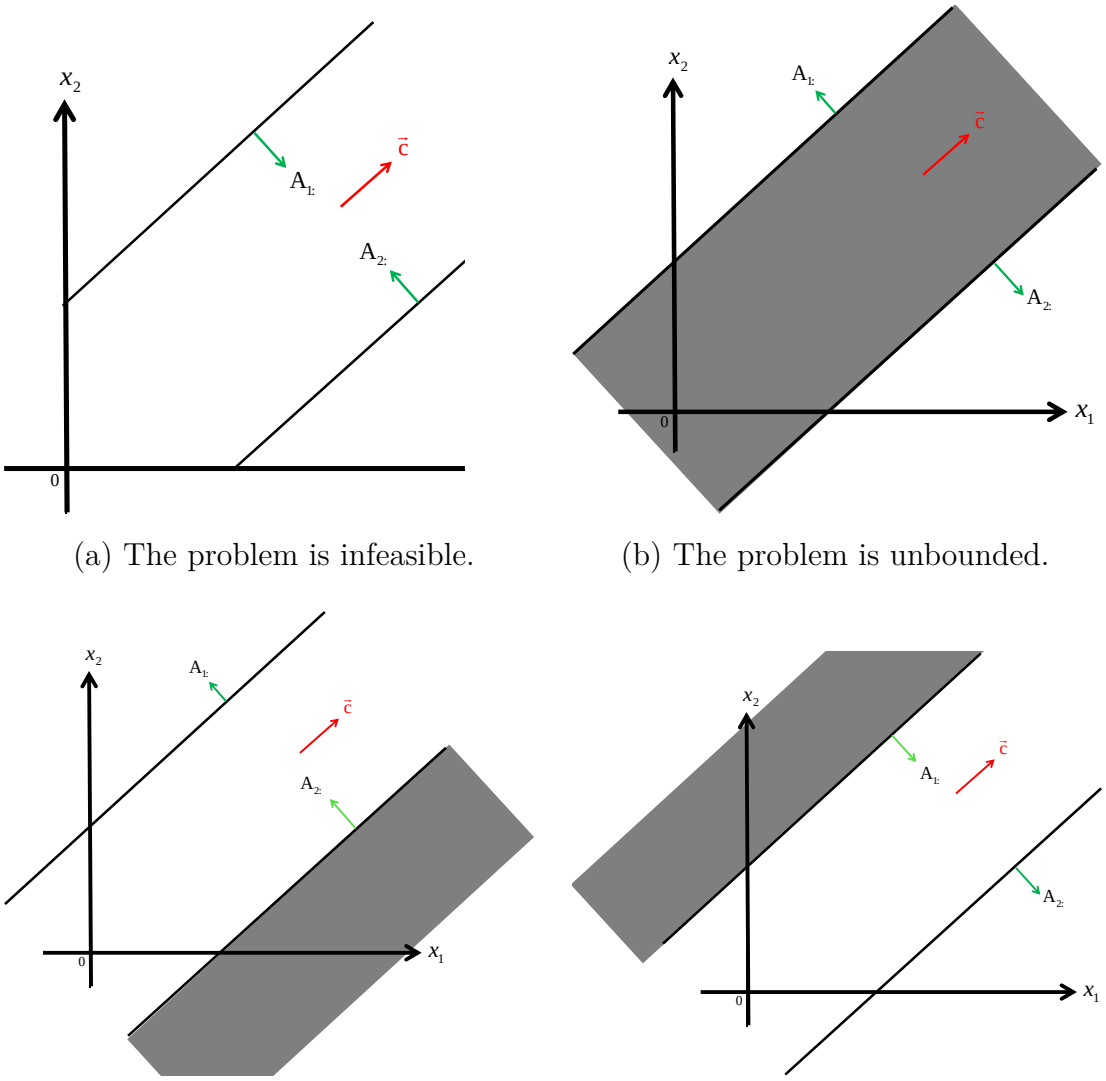
$$\mathbf{x}_0 = \lambda \mathbf{c} \quad \text{where} \quad \lambda = \max_{i \in N^-} \{ \frac{b_i}{\mathbf{A}_{i:} \cdot \mathbf{c}} \}. \tag{3.14}$$

Then, the relaxed problem will be transformed as follows:

$$\begin{aligned} &\text{Maximize} \quad \mathbf{c^T x'} + \mathbf{c}^{\mathrm{T}} \mathbf{x}_0 \\ &\text{subject to} \quad \mathbf{A}_{N_l} \mathbf{x'} \leq \mathbf{b}_{N_l} - \mathbf{A}_{N_l} \mathbf{x}'_0 \end{aligned} \tag{3.15}$$

The simplex can start by adding slack variables without using artificial variables and this problem is unbounded by theorem 3.3. Then, the unbounded case will be applied and constraints from N_e will be reinserted one by one.

In the last case, if $P = \varnothing$ and $N_l = \varnothing$, then some possible cases can occur as in figure 3.13.



(a) The problem is infeasible.

(b) The problem is unbounded.

(c) The problem is unbounded.

(d) The problem is unbounded.

Figure 3.13: Some possible cases of $P = \varnothing$, $N_l = \varnothing$ and $N_e \neq \varnothing$

Then, we will relax all constraints except the first constraint. We can pick a feasible point by choosing one variable with its coefficient is nonzero and set other variables equal to zero. Then the value of the selected variable is the right hand side value divided by its coefficient. The relaxed problem will has a single constraint as

$$\begin{aligned} &\text{Maximize} && \mathbf{c^T x} \\ &\text{subject to} && \mathbf{A}_{1:}\mathbf{x} \leq b_1 \end{aligned} \tag{3.16}$$

We can choose

$$\mathbf{x}_0^{\mathrm{T}} = (0, ..., 0, x_i, 0, ..., 0)^{\mathrm{T}} \quad \text{where} \quad x_i = \frac{b_1}{a_{1i}}, a_{1i} \neq 0. \tag{3.17}$$

Therefore, the relaxed problem can be transformed as follows:

$$\begin{aligned} &\text{Maximize} \quad \mathbf{c^T x'} + \mathbf{c^T x_0} \\ &\text{subject to} \quad \mathbf{A}_{1:}\mathbf{x'} \leq b_1 - \mathbf{A}_{1:}\mathbf{x}_0 \end{aligned} \tag{3.18}$$

The simplex can start by adding slack variables without using an artificial variable and our algorithm can proceed.

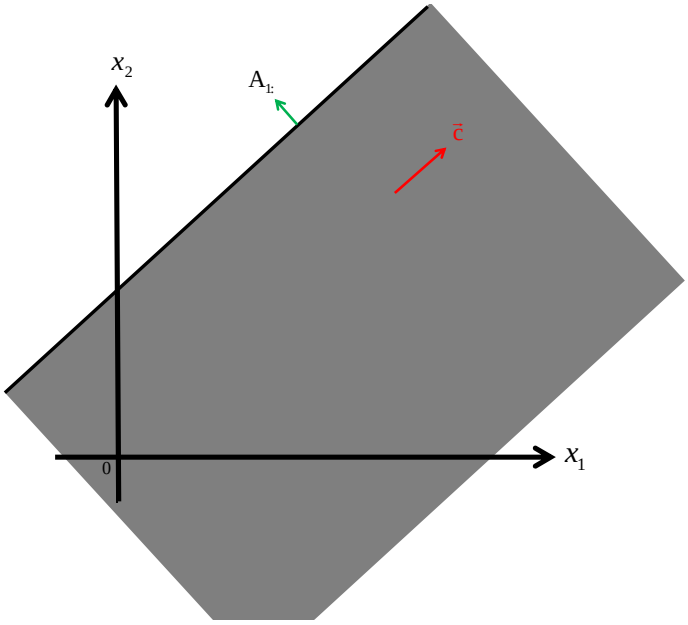


Figure 3.14: Example of the N_e constraint relaxation for $P = \varnothing$ and $N_l = \varnothing$

Summary of Our Algorithm

Accordingly, our algorithm starts from the classification of constraints then non-acute constraints are relaxed. If the origin point of NAR is feasible, then the simplex method will be applied. Otherwise, NAR will be transformed to the new origin point which is identified by theorem 3.1. After that, non-acute constraints will be reinserted to obtain the optimal solution of the original problem.

Since our algorithm solves the non-acute constraint relaxation problem with

the simplex algorithm, we will call our algorithm as "*Simplex method based on Non-Acute constraint Relaxation (SNAR)*".

3.2 SNAR

A general linear programming problem may not be in the form of the problem (3.1). Then we will manipulate the problem to this form as the problem manipulation table in Chapter I. SNAR can be summarized as the following flow chart:

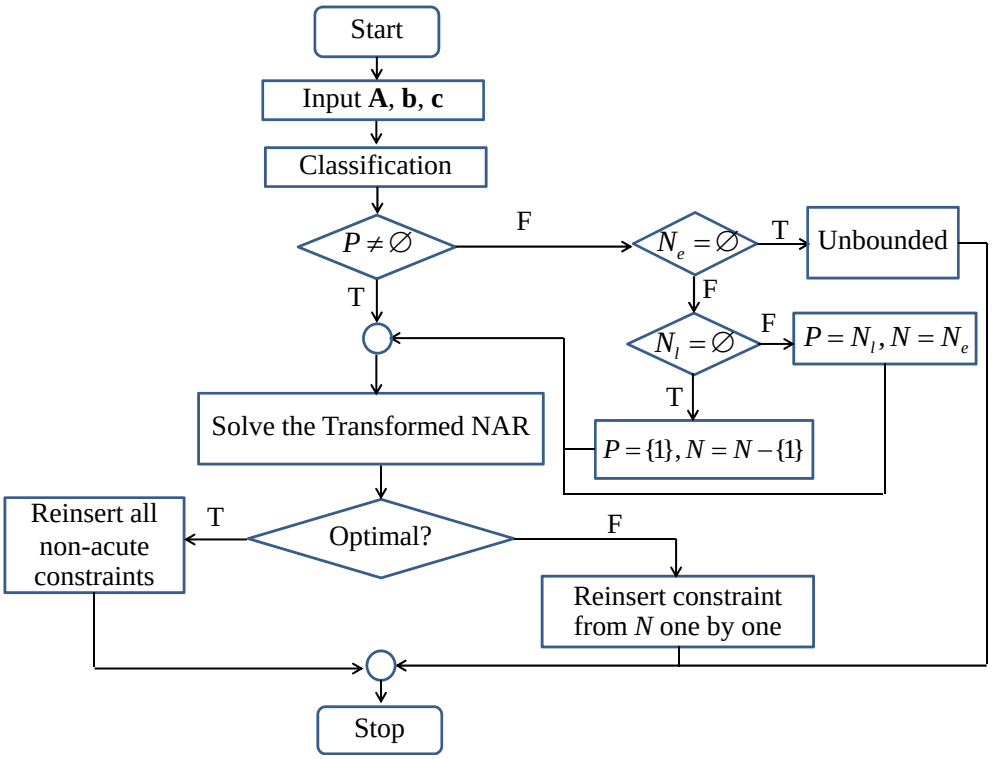


Figure 3.15: Flow chart of SNAR

Each partial algorithm can be stated in detail below. Then, some examples show the algorithm solving step by step.

Algorithm 1 Classification

$P = \varnothing, P^- = \varnothing, N_e = \varnothing, N_l = \varnothing, N_l^- = \varnothing, \delta = 10^{-6}$
for $i = 1 \to m$ **do** compute $\beta_i = A_{i:} \cdot c$
    if $\beta_i > 0$ **then**
        Put i in P
        if $b_i < 0$ **then**
            Put i in P^-
        end if
    else
        Put i in N
        if $\beta = 0$ **then**
            Put i in N_e
        else
            Put i in N_l
            if $b_i < 0$ **then**
                Put i in N_l^-
            end if
        end if
    end if
end for

Algorithm 2 Transform NAR

if $P \neq \varnothing$ **then**

  if $P^- = \varnothing$ **then**

    $\mathbf{x}_0 = \mathbf{0}$

  else

    $\mathbf{x}_0 = -\lambda \mathbf{c}$, where $\lambda = \max_{i \in P^-} \{ \frac{b_i}{-\mathbf{A}_{i:} \cdot \mathbf{c}} \}$

  end if

else

  if $P = \varnothing$ and $N_e = \varnothing$ **then**

    The problem is unbounded. Stop

  else

    if $N_l \neq \varnothing$ **then**

      if $N_l^- = \varnothing$ **then**

        $\mathbf{x}_0 = \mathbf{0}$

      else

        $\mathbf{x}_0 = \lambda \mathbf{c}$, where $\lambda = \max_{i \in N_l^-} \{ \frac{b_i}{\mathbf{A}_{i:} \cdot \mathbf{c}} \}$, $P = N_l, N = N_e$.

      end if

    else

      $P = \{1\}, N = N - \{1\}$ and $x_i = \frac{b_i}{a_{1i}}, a_{1i} \neq 0, x_j = 0,$

$j = 1, ..., n$ and $j \neq i$.

    end if

  end if

end if

Compute $\hat{\mathbf{b}} = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

Algorithm 3 Solve Transformed NAR

46: Solve the following problem:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{A}_P\mathbf{x} \leq \hat{\mathbf{b}}_P \end{aligned}$$

Algorithm 4 Check Optimality

47: **if** $z_j - c_j \geq 0$ for all $j \in R$ **then**

48: go to Algorithm 5.

49: **else**

50: go to Algorithm 6.

51: **end if**

Algorithm 5 Optimal Case

52: **if** $\hat{\mathbf{b}}_N - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}_{P^*}^{-1}\hat{\mathbf{b}}_P \geq \mathbf{0}$ **then**

53: The current solution ($\mathbf{x}'^*$) is the optimal solution and $\mathbf{x}^* = \mathbf{x}'^* + \mathbf{x}_0$ is the optimal solution of the original problem.

54: **else**

55: Add $[\mathbf{0}, \mathbf{A}_{N\mathbf{N}_{P^*}} - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}_{P^*}^{-1}\mathbf{N}_{P^*}, \mathbf{I}_{m_2}, \hat{\mathbf{b}}_N - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}_{P^*}^{-1}\hat{\mathbf{b}}_P]$ into the current tableau. Then perform the dual simplex algorithm.

56: **if** the dual optimal solution is found **then**

57: The current solution ($\mathbf{x}'^*$) is the optimal solution and $\mathbf{x}^* = \mathbf{x}'^* + \mathbf{x}_0$ is the optimal solution of the original problem.

58: **else**

59: The original problem is infeasible. Stop

60: **end if**

61: **end if**

Algorithm 6 Unbounded Case

for $l \in N$ **do**

compute $b'_l = (b_l - \mathbf{A}_{l:}\mathbf{x}_0) - a_{l_{\mathbf{B}_P}}\mathbf{B}_P^{-1}\hat{\mathbf{b}}_P$ and add [0, $a_{l_{\mathbf{N}_P}} - a_{l_{\mathbf{B}_P}}\mathbf{B}_P^{-1}\mathbf{N}_P$, 1, b'_l] into the current tableau.

if $b'_l > 0$ **then**

if $a'_{lj} > 0$ **then**

Apply the primal simplex pivoting at a'_{lj}, $N = N - l, P = P \cup \{l\}$ and perform primal simplex algorithm. Go to Algorithm 3.

else

The problem is unbounded.

end if

else

Replace all $z_j - c_j < 0$ where $j \in J_{\mathbf{N}}$ by δ and perform the dual simplex.

if The optimal dual solution is found **then**

Restore the original $z_j - c_j < 0$ where $j \in J_{\mathbf{N}}$ and perform the primal simplex algorithm.

else

The original problem is infeasible. Then stop.

end if

end if

end for

if the optimal solution is found **then**

The current solution ($\mathbf{x}'^*$) is the optimal solution and $\mathbf{x}^* = \mathbf{x}'^* + \mathbf{x}_0$ is the optimal solution of the original problem. Then stop.

else

The original problem is unbounded. Then stop.

end if

Example 3.4. Consider the linear programming problem:

$$\begin{array}{lrcrcr} \text{Maximize} & x_1 & + & 2x_2 & & \\ \text{subject to} & 2x_1 & + & x_2 & \geq & 4 \\ & 3x_1 & + & 3x_2 & \geq & 9 \\ & x_1 & + & 2x_2 & \geq & 4 \\ & -3x_1 & + & x_2 & \leq & 6 \\ & x_1 & - & 3x_2 & \leq & 6 \\ & 2x_1 & - & 3x_2 & \leq & 12 \\ & 3x_1 & + & 5x_2 & \leq & 30 \\ & & & x_2 & \leq & 5 \\ & x_1 & + & x_2 & \geq & 2 \\ & 4x_1 & + & x_2 & \geq & 4 \end{array} \tag{3.19}$$

Solution. Since some constraints are $\geq$, we multiply the $\geq$ constraints by -1. Then all constraints are $\leq$ as follows:

$$\begin{array}{rcrcr} -2x_1 & - & x_2 & \leq & -4 \\ -3x_1 & - & 3x_2 & \leq & -9 \\ -x_1 & - & 2x_2 & \leq & -4 \\ -3x_1 & + & x_2 & \leq & 6 \\ x_1 & - & 3x_2 & \leq & 6 \\ 2x_1 & - & 3x_2 & \leq & 12 \\ 3x_1 & + & 5x_2 & \leq & 30 \\ & & x_2 & \leq & 5 \\ -x_1 & - & x_2 & \leq & -2 \\ -4x_1 & - & x_2 & \leq & -4 \end{array} \tag{3.20}$$

Then we compute $\mathbf{A}_{i:} \cdot \mathbf{c}$

Constraint No. (i)	$\mathbf{A}_{i:}$	$\mathbf{A}_{i:} \cdot \mathbf{c}$
1	$[-2,-1]^{\mathrm{T}}$	-4
2	$[-3,-3]^{\mathrm{T}}$	-9
3	$[-1,-2]^{\mathrm{T}}$	-5
4	$[-3,1]^{\mathrm{T}}$	-1
5	$[1,-3]^{\mathrm{T}}$	-5
6	$[2,-3]^{\mathrm{T}}$	-4
7	$[3,5]^{\mathrm{T}}$	13
8	$[0,1]^{\mathrm{T}}$	2
9	$[-1,-1]^{\mathrm{T}}$	-3
10	$[-4,-1]^{\mathrm{T}}$	-6

So $P = \{7, 8\}$ and $N = \{1, 2, 3, 4, 5, 6, 9, 10\}$, then the non-acute constraint relaxation is written as:

$$\begin{array}{lrcrcl} \text{Maximize} & x_1 & + & 2x_2 & & \\ \text{subject to} & 3x_1 & + & 5x_2 & \leq & 30 \\ & & & x_2 & \leq & 5 \end{array} \tag{3.21}$$

Since $\mathbf{b}_P = [30, 5]^{\mathrm{T}} \geq \mathbf{0}$, $\mathbf{x}_0 = \mathbf{0}$ is a feasible point. Slack variables will be added into the problem (3.21) to get

$$\begin{array}{lrcrcrcl} \text{Maximize} & x_1 & + & 2x_2 & & & & \\ \text{subject to} & 3x_1 & + & 5x_2 & + & s_1 & = & 30 \\ & & & x_2 & + & s_2 & = & 5 \\ & & & & & s_1, \quad s_2 & \geq & 0. \end{array} \tag{3.22}$$

The relaxed problem is now written in the following standard form.

$$\begin{array}{l} \text{Maximize} \quad {x_1}^+ - {x_1}^- + 2{x_2}^+ - 2{x_2}^- \\ \text{subject to} \quad 3{x_1}^+ - 3{x_1}^- + 5{x_2}^+ - 5{x_2}^- + s_1 = 30 \\ \qquad\qquad {x_2}^+ - {x_2}^- + s_2 = 5 \\ {x_1}^+, \quad {x_1}^-, \quad {x_2}^+, \quad {x_2}^-, \quad s_1, \quad s_2 \geq 0 \end{array} \tag{3.23}$$

and we get the following initial tableau

	z	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	RHS
z	1	-1	1	-2	2	0	0	0
s_1	0	3	-3	5	-5	1	0	30
s_2	0	0	0	(1)	-1	0	1	5

By Dantzig's pivot rule, x_2^+ enters the basis and s_2 leaves the basis in the first iteration. After pivoting, we get

	z	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	RHS
z	1	-1	1	0	0	0	2	10
s_1	0	(3)	-3	0	0	1	-5	5
x_2^+	0	0	0	1	-1	0	1	5

Then x_1^+ enters the basis and s_1 leaves the basis in the second iteration. After pivoting, we get

	z	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	RHS
z	1	0	0	0	0	1/3	1/3	35/3
x_1^+	0	1	-1	0	0	1/3	-5/3	5/3
x_2^+	0	0	0	1	-1	0	1	5

The optimal solution of the problem (3.21) is found at the second iteration. Compute $\hat{\mathbf{b}}_N - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}_{P^*}^{-1}\hat{\mathbf{b}}_P = [13/3, 11, 23/3, 6, 58/3, 71/3, 14/3,\ 23/3]^{\mathrm{T}} > 0$. Then the optimal solution is $x_1^+ = 5/3, x_1^- = 0, x_2^+ = 5, x_2^- = 0$. So the optimal solution of the original problem is $x_1 = x_1^+ - x_1^- = 5/3$, $x_2 = x_2^+ - x_2^- = 5$ and the optimal value is 35/3. □

The simplex algorithm solves this problem by adding 5 artificial variables using 5 iterations in Phase-I and 5 iterations in Phase-II, while SNAR uses 2 iterations without artificial variables.

For the next example, it has a nonempty collection of acute constraints and SNAR algorithm finds the optimal solution from the relaxed problem but it is not the optimal solution from the original problem.

Example 3.5. Consider the following problem:

$$\begin{array}{lrcl} \text{Maximize} & x_2 & & \\ \text{subject to} & x_1 - 2x_2 & \leq & 4 \\ & 3x_1 - 2x_2 & \leq & 6 \\ & x_1 + x_2 & \leq & -4 \\ & -2x_1 + x_2 & \leq & 4 \\ & x_1 & \leq & -3 \end{array} \tag{3.24}$$

Solution. All constraints are $\leq$ and $\mathbf{c}^{\mathrm{T}} = [0, 1]^{\mathrm{T}}$. We compute $\mathbf{A}_{i:} \cdot \mathbf{c}$.

Constraint No. (i)	$\mathbf{A}_{i:}$	$\mathbf{A}_{i:} \cdot \mathbf{c}$
1	$[1, -2]^{\mathrm{T}}$	-2
2	$[3, -2]^{\mathrm{T}}$	-2
3	$[1, 1]^{\mathrm{T}}$	1
4	$[-2, 1]^{\mathrm{T}}$	1
5	$[1, 0]^{\mathrm{T}}$	0

So $P = \{3, 4\}$ and $N = \{1, 2, 5\}$, then the non-acute constraint relaxation is written as:

$$\begin{array}{lrcl} \text{Maximize} & x_2 & & \\ \text{Subjec to} & x_1 + x_2 & \leq & -4 \\ & -2x_1 + x_2 & \leq & 4 \end{array} \tag{3.25}$$

Since $b_3 = -4 < 0$, choose $\mathbf{x}_0 = -\lambda \mathbf{c}$ where $\lambda = \max\{\frac{-4}{-1}\} = 4$. So ${\mathbf{x}_0}^{\mathrm{T}} = [0, -4]^{\mathrm{T}}$, $\mathbf{c}^{\mathrm{T}}\mathbf{x}_0 = [0, 1]^{\mathrm{T}} \begin{bmatrix} 0 \\ -4 \end{bmatrix} = -4$ and $\hat{\mathbf{b}}_P = \mathbf{b}_P - \mathbf{A}_P\mathbf{x}_0 = \begin{bmatrix} -4 \\ 4 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -4 \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \end{bmatrix}$. Then the transformed problem is rewritten as:

$$\begin{array}{lrcl} \text{Maximize} & {x'}_2 - 4 & & \\ \text{Subjec to} & {x'}_1 + {x'}_2 & \leq & 0 \\ & -2{x'}_1 + {x'}_2 & \leq & 8 \end{array} \tag{3.26}$$

where $\mathbf{x}' = \mathbf{x} - \mathbf{x}_0$. The transformed problem is written in the following standard form.

$$\begin{array}{lrcl} \text{Maximize} & {x_2}'^{+} - {x_2}'^{-} - 4 & & \\ \text{Subjec to} & {x_1}'^{+} - {x_1}'^{-} + {x_2}'^{+} - {x_2}'^{-} + s_1 & = & 0 \\ & -2{x_1}'^{+} + 2{x_1}'^{-} + {x_2}'^{+} - {x_2}'^{-} + s_2 & = & 8 \\ & {x_1}'^{+}, {x_1}'^{-}, {x_2}'^{+}, {x_2}'^{-}, s_1, s_2 & \geq & 0 \end{array} \tag{3.27}$$

Start with the simplex algorithm, and the initial tableau is

	z	${x_1}'^{+}$	${x_1}'^{-}$	${x_2}'^{+}$	${x_2}'^{-}$	s_1	s_2	RHS
z	1	0	0	-1	1	0	0	-4
s_1	0	1	-1	(1)	-1	1	0	0
s_2	0	-2	2	1	-1	0	1	8

By Dantzig's pivot rule, ${x_2}'^{+}$ enters the basis and s_1 leaves the basis in the first iteration. After pivoting, we get

	z	${x_1}'^{+}$	${x_1}'^{-}$	${x_2}'^{+}$	${x_2}'^{-}$	s_1	s_2	RHS
z	1	1	-1	0	0	1	0	-4
${x_2}'^{+}$	0	1	-1	1	-1	1	0	0
s_2	0	-3	(3)	0	0	-1	1	8

In the second iteration, ${x_1}'^{-}$ enters the basis and s_2 leaves the basis. After pivoting, we get

	z	${x_1}'^{+}$	${x_1}'^{-}$	${x_2}'^{+}$	${x_2}'^{-}$	s_1	s_2	RHS
z	1	0	0	0	0	2/3	1/3	-4/3
${x_2}'^{+}$	0	0	0	1	-1	2/3	1/3	8/3
${x_1}'^{-}$	0	-1	1	0	0	-1/3	1/3	8/3

The optimal solution of the relaxed problem is found at the second iteration. Compute $\hat{\mathbf{b}}_N - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}_{P^*}^{-1}\hat{\mathbf{b}}_P = [4, 34/3, -1/3]^{\mathrm{T}} \not\geq 0$. Add $[\mathbf{0},\ \mathbf{A}_{N\mathbf{N}_{P^*}} - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}_{P^*}^{-1}\mathbf{N}_{P^*},\ \mathbf{I}_{m_2},\ \hat{\mathbf{b}}_N - \mathbf{A}_{N\mathbf{B}_{P^*}}\mathbf{B}_{P^*}^{-1}\hat{\mathbf{b}}_P]$ into the current tableau. We get

	z	${x_1}'^{+}$	${x_1}'^{-}$	${x_2}'^{+}$	${x_2}'^{-}$	s_1	s_2	s_3	s_4	s_5	RHS
z	1	0	0	0	0	2/3	1/3	0	0	0	-4/3
${x_2}'^{+}$	0	0	0	1	-1	2/3	1/3	0	0	0	8/3
${x_1}'^{-}$	0	-1	1	0	0	-1/3	1/3	0	0	0	8/3
s_3	0	0	0	0	0	1	1	1	0	0	4
s_4	0	0	0	0	0	1/3	5/3	0	1	0	34/3
s_5	0	0	0	0	0	-1/3	1/3	0	0	1	-1/3

The dual simplex method can be applied by choosing s_1 to enter the basis and s_5 to leave the basis. The result can be shown in the following tableau.

	z	${x_1}'^{+}$	${x_1}'^{-}$	${x_2}'^{+}$	${x_2}'^{-}$	s_1	s_2	s_3	s_4	s_5	RHS
z	1	0	0	0	0	0	1	0	0	2	-2
${x_2}'^{+}$	0	0	0	1	-1	0	1	0	0	2	2
${x_1}'^{-}$	0	-1	1	0	0	0	0	0	0	-1	3
s_3	0	0	0	0	0	0	2	1	0	3	3
s_4	0	0	0	0	0	0	2	0	1	1	11
s_1	0	0	0	0	0	1	-1	0	0	-3	1

The optimal solution is found and ${x_1}'^{-} = 3, {x_1}'^{+} = 0, {x_2}'^{+} = 2, {x_2}'^{-} = 0, s_1 = 1, s_2 = 0, s_3 = 3, s_4 = 11$ and $s_5 = 0$. So the optimal solution for the transformed problem is $x_1' = {x_1}'^{+} - {x_1}'^{-} = -3$, $x_2' = {x_2}'^{+} - {x_2}'^{-} = 2$. Then the optimal solution of the original problem is $\mathbf{x} = \mathbf{x}' + \mathbf{x}_0 = \begin{bmatrix} -3 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ -4 \end{bmatrix} = \begin{bmatrix} -3 \\ -2 \end{bmatrix}$ and the optimal value is -2. □

For this problem, the simplex algorithm solves by adding 2 artificial variables in Phase-I and uses 3 iterations to obtain the optimal solution while SNAR uses 3 iterations without artificial variables.

Example 3.6. Consider the following problem:

$$\begin{array}{lrcrcr} \text{Maximize} & -x & & & & \\ \text{subject to} & -3x & + & 4y & \leq & 12 \\ & x & & & \leq & -1 \\ & 2x & - & y & \leq & -2 \\ & x & + & y & \leq & 1 \\ & & - & y & \leq & -2 \end{array} \tag{3.28}$$

Solution. All constraints are $\leq$ and $\mathbf{c}^{\mathrm{T}} = [-1, 0]^{\mathrm{T}}$. We compute $\mathbf{A}_{i:} \cdot \mathbf{c}$.

Constraint No. (i)	$\mathbf{A}_{i:}$	$\mathbf{A}_{i:} \cdot \mathbf{c}$
1	$[-3, 4]^{\mathrm{T}}$	3
2	$[1, 0]^{\mathrm{T}}$	-1
3	$[2, -1]^{\mathrm{T}}$	-2
4	$[1, 1]^{\mathrm{T}}$	-1
5	$[0, -1]^{\mathrm{T}}$	0

So $P = \{1\}$ and $N = \{2, 3, 4, 5\}$, then the non-acute constraint relaxation is written as:

$$\begin{array}{lrcr} \text{Maximize} & -x & & \\ \text{Subjec to} & -3x + 4y & \leq & 12 \end{array} \tag{3.29}$$

Since $b_1 = 12 > 0$, $\mathbf{x}_0 = \mathbf{0}$ is a feasible point. Let $x = x_1 - x_2$ and $y = x_3 - x_4$. The transformed problem is written in the following standard form.

$$\begin{array}{lrcr} \text{Maximize} & -x_1 + x_2 & & \\ \text{Subjec to} & -3x_1 + 3x_2 + 4x_3 - 4x_4 + s_1 & = & 12 \\ & x_1, x_2, x_3, x_4, s_1 & \geq & 0 \end{array} \tag{3.30}$$

Start with the simplex algorithm, and the initial tableau is in the following:

	z	x_1	x_2	x_3	x_4	s_1	RHS
z	1	1	-1	0	0	0	0
s_1	0	-3	(3)	4	-4	1	12

By Dantzig's pivot rule, x_2 enters the basis and s_1 leaves the basis in the first iteration. After pivoting, we get

	z	x_1	x_2	x_3	x_4	s_1	RHS
z	1	0	0	4/3	-4/3	1/3	4
x_2	0	-1	1	4/3	-4/3	1/3	4

and find that problem (3.30) is unbounded. So add a constraint from N into the current tableau one by one. The first constraint from N is added and we get the following tableau

	z	x_1	x_2	x_3	x_4	s_1	s_2	RHS
z	1	0	0	4/3	-4/3	1/3	0	4
x_2	0	-1	1	4/3	-4/3	1/3	0	4
s_2	0	0	0	4/3	-4/3	1/3	1	3

The problem is still unbounded. The second constraint from N is added, we get the following tableau.

	z	x_1	x_2	x_3	x_4	s_1	s_2	s_3	RHS
z	1	0	0	4/3	-4/3	1/3	0	0	4
x_2	0	-1	1	4/3	-4/3	1/3	0	0	4
s_2	0	0	0	4/3	-4/3	1/3	1	0	3
s_3	0	0	0	5/3	-5/3	2/3	0	1	6

The problem is still unbounded. The third constraint from N is added, we get the following tableau.

	z	x_1	x_2	x_3	x_4	s_1	s_2	s_3	s_4	RHS
z	1	0	0	4/3	-4/3	1/3	0	0	0	4
x_2	0	-1	1	4/3	-4/3	1/3	0	0	0	4
s_2	0	0	0	4/3	-4/3	1/3	1	0	0	3
s_3	0	0	0	5/3	-5/3	2/3	0	1	0	6
s_4	0	0	0	7/3	-7/3	1/3	0	0	1	5

The problem is still unbounded. The last constraint from N is added, we get the following tableau.

	z	x_1	x_2	x_3	x_4	s_1	s_2	s_3	s_4	s_5	RHS
z	1	0	0	4/3	-4/3	1/3	0	0	0	0	4
x_2	0	-1	1	4/3	-4/3	1/3	0	0	0	0	4
s_2	0	0	0	4/3	-4/3	1/3	1	0	0	0	3
s_3	0	0	0	5/3	-5/3	2/3	0	1	0	0	6
s_4	0	0	0	7/3	-7/3	1/3	0	0	1	0	5
s_5	0	0	0	-1	1	0	0	0	0	1	-2

Since $b'_5 < 0$, we perturb $z_4 - c_4$ to $\delta = 10^{-6}$ and add the $z_{perterb}$ row to the tableau as follows:

	z	x_1	x_2	x_3	x_4	s_1	s_2	s_3	s_4	s_5	RHS
z	1	0	0	4/3	-4/3	1/3	0	0	0	0	4
$z_{perterb}$	1	0	0	4/3	10^{-6}	1/3	0	0	0	0	4
x_2	0	-1	1	4/3	-4/3	1/3	0	0	0	0	4
s_2	0	0	0	4/3	-4/3	1/3	1	0	0	0	3
s_3	0	0	0	5/3	-5/3	2/3	0	1	0	0	6
s_4	0	0	0	7/3	-7/3	1/3	0	0	1	0	5
s_5	0	0	0	(-1)	1	0	0	0	0	1	-2

Perform the dual simplex, x_3 enters the basis and s_5 leaves the basis. We get

	z	x_1	x_2	x_3	x_4	s_1	s_2	s_3	s_4	s_5	RHS
z	1	0	0	0	0	1/3	0	0	0	4/3	4/3
$z_{perterb}$	1	0	0	0	4/3	1/3	0	0	0	4/3	4/3
x_2	0	-1	1	0	0	1/3	0	0	0	4/3	4/3
s_2	0	0	0	0	0	1/3	1	0	0	4/3	1/3
s_3	0	0	0	0	0	2/3	0	1	0	5/3	8/3
s_4	0	0	0	0	0	1/3	0	0	1	7/3	1/3
x_3	0	0	0	1	-1	0	0	0	0	-1	2

The right hand side is positive. So a primal solution is feasible. Consider the original row 0, the optimal solution is found at this iteration. The optimal solution is $x = x_1 - x_2 = 0 - 4/3 = -4/3$, $y = x_3 - x_4 = 2$ and the optimal value is $4/3$. $\square$

For this problem, the simplex algorithm needs by adding 3 artificial variables and used 4 iterations to find a feasible solution in Phase-I. Phase-II uses 1 iteration to obtain the optimal solution while SNAR uses 2 iterations without artificial variables.

Example 3.7. Consider the following problem:

$$\begin{array}{lrcrcr} \text{Maximize} & -x_1 & + & x_2 & & \\ \text{subject to} & 5x_1 & - & x_2 & \leq & -5 \\ & 2x_1 & - & x_2 & \leq & -4 \\ & 2x_1 & + & x_2 & \leq & -3 \\ & x_1 & - & 2x_2 & \leq & -4 \\ & x_1 & - & 5x_2 & \leq & -5 \\ & x_1 & & & \leq & 0 \\ & & - & x_2 & \leq & 0 \end{array} \tag{3.31}$$

Solution. All constraints are $\leq$ and $\mathbf{c}^{\mathrm{T}} = [-1, 1]^{\mathrm{T}}$. We compute $\mathbf{A}_{i:} \cdot \mathbf{c}$.

Constraint No. (i)	$\mathbf{A}_{i:}$	$\mathbf{A}_{i:} \cdot \mathbf{c}$
1	$[5, -1]^{\mathrm{T}}$	-6
2	$[2, -1]^{\mathrm{T}}$	-3
3	$[2, 1]^{\mathrm{T}}$	-1
4	$[1, -2]^{\mathrm{T}}$	-3
5	$[1, -5]^{\mathrm{T}}$	-6
6	$[1, 0]^{\mathrm{T}}$	-1
7	$[0, -1]^{\mathrm{T}}$	-1

So $P = \varnothing$, $N_e = \varnothing$ and $N = \{1, 2, 3, 4, 5, 6, 7\}$. From theorem 3.3, this problem is unbounded. □

For this problem, the simplex algorithm solves by adding 5 artificial variables and uses 7 iterations to obtain the unbounded solution while SNAR can conclude the unbounded optimal solution by our theorem.

Example 3.8. Consider the following problem:

$$\begin{array}{lrcrcrcr}
\text{Maximize} & -5x_1 & - & 4x_2 & - & 3x_3 & & \\
\text{subject to} & x_1 & & - & & x_3 & = & -1 \\
& x_1 & + & x_2 & - & 2x_3 & = & -1 \\
& -x_1 & + & x_2 & - & x_3 & = & -1 \\
& & - & 2x_2 & + & 3x_3 & = & 2 \\
& 3x_1 & & & & & = & 3 \\
& & - & 2x_2 & + & x_3 & = & -2 \\
& 3x_1 & - & 2x_2 & + & 2x_3 & = & 3 \\
& 3x_1 & & & + & 3x_3 & = & 9 \\
& x_1, & & x_2, & & x_3 & \geq & 0
\end{array} \tag{3.32}$$

Solution. Since this problem is in standard form which is not in form as the problem (3.1), it is transformed into:

$$\begin{array}{lrcrcrcr}
\text{Maximize} & -5x_1 & - & 4x_2 & - & 3x_3 & & \\
\text{subject to} & x_1 & & & - & x_3 & = & -1 \\
& x_1 & + & x_2 & - & 2x_3 & \leq & -1 \\
& -x_1 & -x_2 & + & 2x_3 & \leq & 1 & \\
& -x_1 & + & x_2 & - & x_3 & \leq & -1 \\
& x_1 & - & x_2 & + & x_3 & \leq & 1 \\
& & - & 2x_2 & + & 3x_3 & \leq & 2 \\
& & + & 2x_2 & - & 3x_3 & \leq & -2 \\
& 3x_1 & & & & & \leq & 3 \\
& -3x_1 & & & & & \leq & -3 \\
& & - & 2x_2 & + & x_3 & \leq & -2 \\
& & & 2x_2 & - & x_3 & \leq & 2 \\
& 3x_1 & - & 2x_2 & + & 2x_3 & \leq & 3 \\
& -3x_1 & + & 2x_2 & - & 2x_3 & \leq & -3
\end{array} \tag{3.33}$$

$$\begin{aligned} 3x_1 \quad &+ \quad 3x_3 \leq \quad 9 \\ -3x_1 \quad &- \quad 3x_3 \leq -9 \\ -x_1 \quad & \qquad \leq \quad 0 \\ -\ x_2 \quad & \qquad \leq \quad 0 \\ -\ x_3 \quad &\leq \quad 0 \end{aligned}$$

Before SNAR starts, the number of constraints is double. Moreover, the number of variables will increase twice for the standard form while the simplex method solves this problem by adding eight artificial variables. Therefore, the number of dimensions of parameter by SNAR solved is larger than the number of dimensions of parameter by the simplex method solved. So we will consider the dual of the standard form. □

3.3 Dual SNAR

Consider the primal linear programming problem in the standard form which is given below:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{3.34}$$

where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{c}$ is an n-dimensional vector and $\mathbf{b}$ is an m-dimensional vector.

Solving this by SNAR, the dimension of parameter will be expanded as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & -\mathbf{Ax} \leq -\mathbf{b} \\ & -\mathbf{x} \leq \mathbf{0}. \end{aligned} \tag{3.35}$$

Consequently, we will consider the dual of this standard form which is defined by:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{b^T w} \\ \text{subject to} \quad & \mathbf{A^T w} \geq \mathbf{c}. \end{aligned} \tag{3.36}$$

After the problem is transformed, it can be rewritten as follows:

$$\begin{aligned} -\text{Maximize} \quad & -\mathbf{b^T w} \\ \text{subject to} \quad & -\mathbf{A^T w} \leq -\mathbf{c}. \end{aligned} \tag{3.37}$$

This form is the same as the problem (3.1) which have the objective vector $-\mathbf{b}$, the coefficient matrix $-\mathbf{A}^{\mathrm{T}}$, and the right-hand side vector is $\mathbf{c}$. We can use the objective vector $-\mathbf{b}$ to separate the collections of constraints similar to SNAR. Since $(-\mathbf{A}_{i:}^{\mathrm{T}}) \cdot (-\mathbf{b}) = \mathbf{A}_{i:}^{\mathrm{T}} \cdot \mathbf{b}$, we can separate the collections of acute constraints and non-acute constraints as follows:

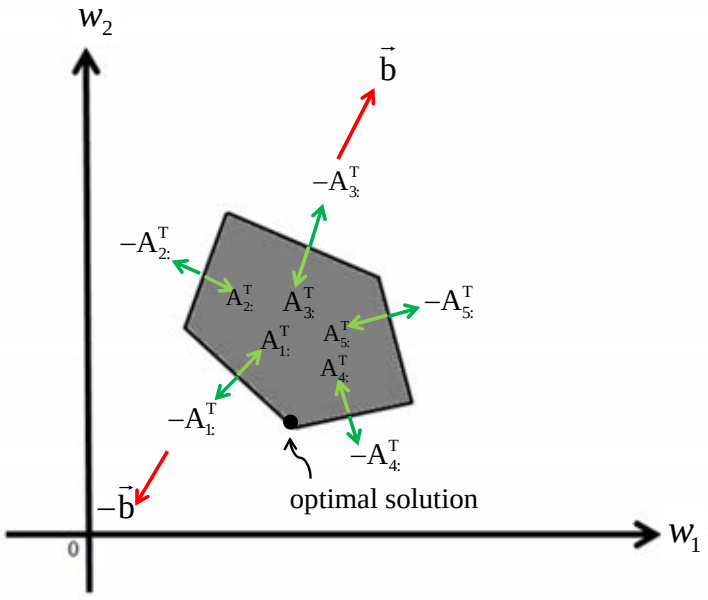


Figure 3.16: Example of the feasible region of the dual problem

Therefore, we can use SNAR by using the matrix $-\mathbf{A}^{\mathrm{T}}$ instead of $\mathbf{A}$, using the vector $-\mathbf{b}$ instead of $\mathbf{c}$, the vector -$\mathbf{c}$ is the right-hand side of this algorithm and the objective value is $-z$. Since we use SNAR in the dual form, we call this algorithm *the Dual SNAR*. From lemma 2.23, the optimal solution is $x_i^* = -z_{m+i}$ for all $i = 1, ..., n$. If SNAR reports unbounded optimal solution, then the original

problem is infeasible. If SNAR reports the infeasiblity, the original problem is unbounded.

The Dual SNAR can be summarized as the following flow chart.

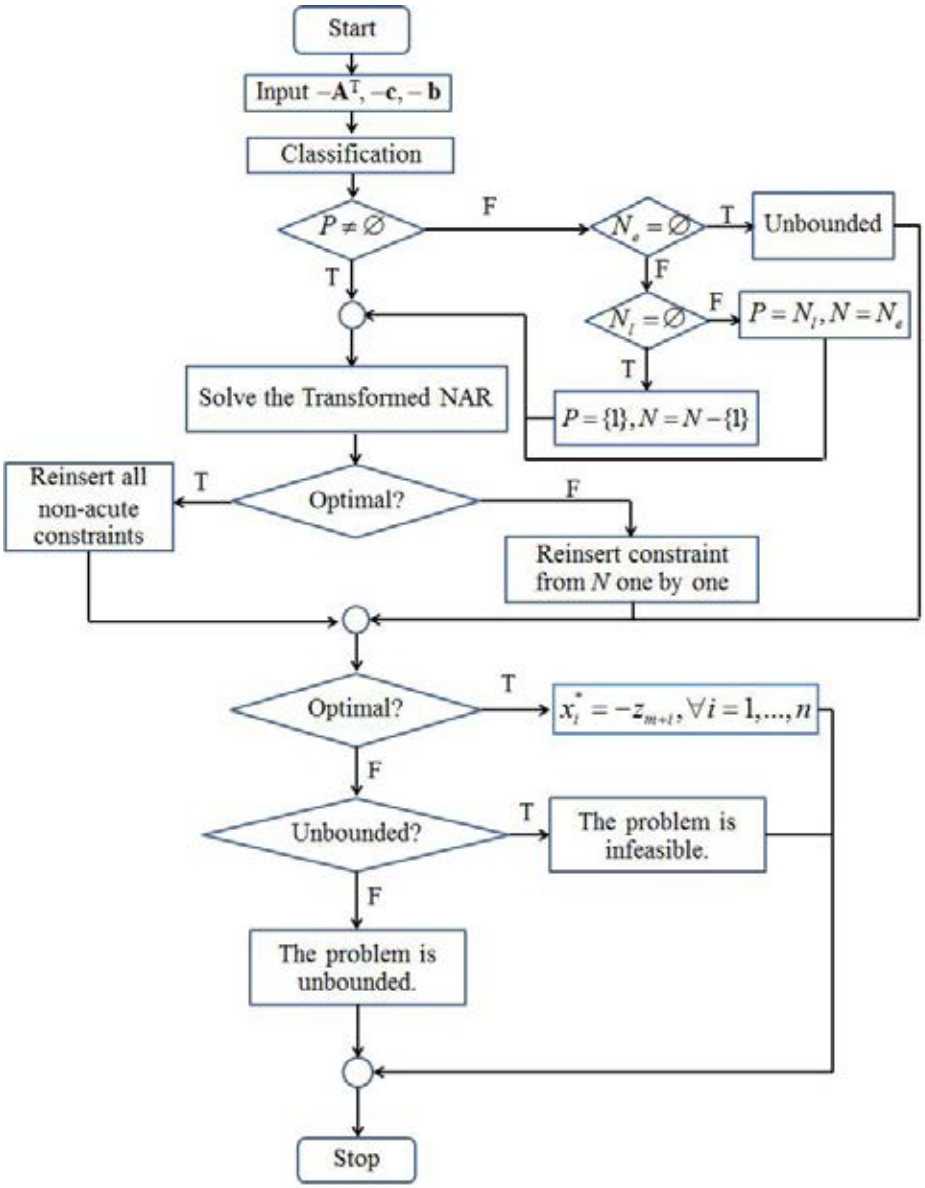


Figure 3.17: Flow chart of Dual SNAR

Example 3.9. Consider the following problem:

$$\begin{aligned}
\text{Maximize } & -5x_1 - 4x_2 - 3x_3 \\
\text{subject to } & -x_1 + x_2 - x_3 = -1 \\
& -2x_2 + 3x_3 = 2 \\
& -2x_2 + x_3 = -2 \\
& 3x_1 - 2x_2 + 2x_3 = 3 \\
& 3x_1 + 3x_3 = 9 \\
& x_1, \quad x_2, \quad x_3 \geq 0
\end{aligned} \tag{3.39}$$

Solution. The dual form of this problem can be written as follows:

$$\begin{aligned}
-\text{Maximize } & w_1 - 2w_2 + 2w_3 - 3w_4 - 9w_5 \\
\text{subject to } & w_1 - 3w_4 - 3w_5 \leq 5 \\
& -w_1 + 2w_2 + 2w_3 + 2w_4 \leq 4 \\
& w_1 - 3w_2 - w_3 - 2w_4 - 3w_5 \leq 3
\end{aligned} \tag{3.40}$$

Then we compute $\mathbf{A}_{i:}^{\mathrm{T}} \cdot \mathbf{b}$ as the following tableau:

Constraint No. (i)	$\mathbf{A}_{i:}^{\mathrm{T}} \cdot \mathbf{b}$
1	10
2	-7
3	38

So $P = \{1, 3\}$ and $N = \{2\}$, then the non-acute constraint relaxation is written as:

$$\begin{aligned}
-\text{Maximize } & w_1 - 2w_2 + 2w_3 - 3w_4 - 9w_5 \\
\text{subject to } & w_1 - 3w_4 - 3w_5 \leq 5 \\
& w_1 - 3w_2 - w_3 - 2w_4 - 3w_5 \leq 3
\end{aligned} \tag{3.41}$$

Since $\mathbf{x}_0 = \mathbf{0}$ is a feasible point, slack variables will be added into the problem (3.41).

$$\begin{aligned}
-\text{Maximize } & w_1 - 2w_2 + 2w_3 - 3w_4 - 9w_5 \\
\text{subject to } & w_1 - 3w_4 - 3w_5 + s_1 = 5 \\
& w_1 - 3w_2 - w_3 - 2w_4 - 3w_5 + s_2 = 3
\end{aligned} \tag{3.42}$$

Before we put these to the tableau, it will be converted to the standard form. Then, we get the following initial tableau:

	z	w_1^+	w_1^-	w_2^+	w_2^-	w_3^+	w_3^-	w_4^+	w_4^-	w_5^+	w_5^-	s_1	s_2	RHS
$-z$	1	-1	1	2	-2	-2	2	3	-3	9	-9	0	0	0
s_1	0	1	-1	0	0	0	0	-3	3	-3	3	1	0	5
s_2	0	1	-1	-3	3	-1	1	-2	2	-3	(3)	0	1	3

By Dantzig's pivot rule, w_5^- enters the basis and s_2 leaves the basis in the first iteration. After pivoting, we get

	z	w_1^+	w_1^-	w_2^+	w_2^-	w_3^+	w_3^-	w_4^+	w_4^-	w_5^+	w_5^-	s_1	s_2	RHS
$-z$	1	2	-2	-7	7	-5	5	-3	3	0	0	0	3	9
s_1	0	0	0	(3)	-3	1	-1	-1	1	0	0	1	-1	2
w_5^-	0	1/3	- 1/3	-1	1	- 1/3	1/3	- 2/3	2/3	-1	1	0	1/3	1

w_2^+ enters the basis and s_1 leaves the basis in the second iteration. After pivoting, we get

	z	w_1^+	w_1^-	w_2^+	w_2^-	w_3^+	w_3^-	w_4^+	w_4^-	w_5^+	w_5^-	s_1	s_2	RHS
$-z$	1	2	-2	0	0	-8/3	8/3	-16/3	16/3	0	0	7/3	2/3	41/3
w_2^+	0	0	0	1	1	1/3	- 1/3	- 1/3	1/3	0	0	1/3	- 1/3	2/3
w_5^-	0	1/3	- 1/3	0	0	0	0	-1	1	-1	1	1/3	0	5/3

NAR is unbounded. The second constraint will be added to the tableau as follows:

	z	w_1^+	w_1^-	w_2^+	w_2^-	w_3^+	w_3^-	w_4^+	w_4^-	w_5^+	w_5^-	s_1	s_2	s_3	RHS
$-z$	1	2	-2	0	0	-8/3	8/3	-16/3	16/3	0	0	7/3	2/3	0	41/3
w_2^+	0	0	0	1	-1	1/3	- 1/3	- 1/3	1/3	0	0	1/3	- 1/3	0	2/3
w_5^-	0	1/3	- 1/3	0	0	0	0	-1	1	-1	1	1/3	0	0	5/3
s_3	0	-1	1	2	-2	2	-2	2	-2	0	0	0	0	1	4

Apply with this basis, then we get

	z	w_1^+	w_1^-	w_2^+	w_2^-	w_3^+	w_3^-	w_4^+	w_4^-	w_5^+	w_5^-	s_1	s_2	s_3	RHS
$-z$	1	2	-2	0	0	-8/3	8/3	-16/3	16/3	0	0	7/3	2/3	0	41/3
w_2^+	0	0	0	1	-1	1/3	- 1/3	- 1/3	1/3	0	0	1/3	- 1/3	0	2/3
w_5^-	0	1/3	- 1/3	0	0	0	0	-1	1	-1	1	1/3	0	0	5/3
s_3	0	-1	1	0	0	4/3	-4/3	(8/3)	-8/3	0	0	-2/3	2/3	1	2 2/3

$w_7{}^+$ enters the basis and s_3 leaves the basis in the second iteration. After pivoting, we get

	z	$w_1{}^+$	$w_1{}^-$	$w_2{}^+$	$w_2{}^-$	$w_3{}^+$	$w_3{}^-$	$w_4{}^+$	$w_4{}^-$	$w_5{}^+$	$w_5{}^-$	s_1	s_2	s_3	RHS
$-z$	1	0	0	0	0	0	0	0	0	0	0	1	2	2	19
$w_4{}^+$	0	- 1/8	1/8	1	-1	1/2	- 1/2	0	0	0	0	1/4	- 1/4	1/8	1
$w_8{}^-$	0	0	0	0	0	1/2	- 1/2	0	0	-1	1	0	1/4	3/8	8/3
$w_7{}^+$	0	- 3/8	3/8	0	0	1/2	- 1/2	1	-1	0	0	- 1/4	1/4	3/8	1

The optimal solution of this problem is found at the third iteration. Since the dual of this problem is the original problem, the optimal solution of this tableau will be the optimal solution of the original problem. From lemma (2.23), we obtain the optimal solution (x_1^*, x_2^*, x_3^*) = (1,2,2) and the optimal value is -19. □

3.4 Comparison of the Problem Dimensions

3.4.1 SNAR vs Two-Phase Method

Recall the simplex algorithm, it performs on a linear programming problem in the standard form. So the original problem (3.1) is transformed to

$$\begin{aligned} &\text{Maximize } \mathbf{c}^{\mathrm{T}}\mathbf{x}^+ - \mathbf{c}^{\mathrm{T}}\mathbf{x}^- + \mathbf{0s} \\ &\text{subject to } \mathbf{Ax}^+ - \mathbf{Ax}^- + \mathbf{s} = \mathbf{b} \\ &\qquad\qquad \mathbf{x}^+, \quad \mathbf{x}^-, \quad \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{3.43}$$

The number of dimensions of the problem (3.43) is as follows:

A	**b**	**I**	**c**	Total
$m \times 2n$	m	$m \times m$	$2n + m$	$(m \times (2n + m)) + 2m + 2n$

If there exist $b_i < 0, i \in \{1, ..., m\}$, we can not choose identity matrix as the basis. Suppose $\mathbf{b}$ is split into $\mathbf{b}^+ \geq \mathbf{0}$ and $\mathbf{b}^- < \mathbf{0}$. Therefore, it will be multiplied by -1. Then, we can rewrite the system (3.43) as follows:

$$\begin{aligned}
\text{Maximize} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x}^{+} - \mathbf{c}^{\mathrm{T}}\mathbf{x}^{-} \\
\text{subject to} \quad & \mathbf{A}^{+}\mathbf{x}^{+} - \mathbf{A}^{+}\mathbf{x}^{-} + \mathbf{s}^{+} = \mathbf{b}^{+} \\
& -\mathbf{A}^{-}\mathbf{x}^{+} + \mathbf{A}^{-}\mathbf{x}^{-} - \mathbf{s}^{-} = -\mathbf{b}^{-} \\
& \mathbf{x}^{+}, \quad \mathbf{x}^{-}, \quad \mathbf{s}^{+}, \quad \mathbf{s}^{-} \geq \mathbf{0},
\end{aligned} \tag{3.44}$$

where $\mathbf{A} = \begin{bmatrix} \mathbf{A}^{+} \\ \mathbf{A}^{-} \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} \mathbf{b}^{+} \\ \mathbf{b}^{-} \end{bmatrix}$, $\mathbf{b}^{+}$ is an m^{+} dimensional vector and $\mathbf{b}^{-}$ is an m^{-} dimensional vector, $\mathbf{b}^{+} \geq \mathbf{0}$, $\mathbf{b}^{-} < \mathbf{0}$ and $m^{+} + m^{-} = m$.

Since the identity matrix is not the initial basis, the Two-Phase method need to add m^{-} artificial variables. Then, Phase-I can be written as the following.

$$\begin{aligned}
\text{Minimize} \quad & \mathbf{1}^{\mathrm{T}}\mathbf{x}_a \\
\text{subject to} \quad & \mathbf{A}^{+}\mathbf{x}^{+} - \mathbf{A}^{+}\mathbf{x}^{-} + \mathbf{s}^{+} = \mathbf{b}^{+} \\
& -\mathbf{A}^{-}\mathbf{x}^{+} + \mathbf{A}^{-}\mathbf{x}^{-} - \mathbf{s}^{-} + \mathbf{x}_a = -\mathbf{b}^{-} \\
& \mathbf{x}^{+}, \quad \mathbf{x}^{-}, \quad \mathbf{s}^{+}, \quad \mathbf{s}^{-}, \quad \mathbf{x}_a \geq \mathbf{0},
\end{aligned} \tag{3.45}$$

Consider SNAR, we can choose $\mathbf{x}_0$ which is a feasible point. So $\mathbf{A}_P\mathbf{x}_0 \leq \mathbf{b}_P$ and $\mathbf{b}_P - \mathbf{A}_P\mathbf{x}_0 \geq \mathbf{0}$. Then slack variables are added to transform inequality constraints to equality constraints.

$$\begin{aligned}
\text{Maximize} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x}^{+} - \mathbf{c}^{\mathrm{T}}\mathbf{x}^{-} + \mathbf{c}^{\mathrm{T}}\mathbf{x_0} \\
\text{subject to} \quad & \mathbf{A}_P\mathbf{x}^{+} - \mathbf{A}_P\mathbf{x}^{-} + \mathbf{s} = \mathbf{b}_P - \mathbf{A}_P\mathbf{x}_0 \\
& \mathbf{x}^{+}, \quad \mathbf{x}^{-}, \quad \mathbf{s} \geq 0
\end{aligned} \tag{3.46}$$

Therefore, we can compare the number of dimensions of the problem between SNAR and Two-Phase method as the following table.

Table 3.1: Comparison the number of dimensions of parameters between SNAR and Two-Phase Method

Method		Dimension of parameters
SNAR	NAR	$(m_1 \times (2n + m_1)) + 2n + 2m_1$
	AN	$(m \times (2n + m)) + 2n + 2m$
Two-Phase Method	Phase I	$(m \times (2n + m + m^-)) + 2n + 2m + m^-$
	Phase II	$(m \times (2n + m + m^-)) + 2n + 2m + m^-$

where m_1 is the number of the acute constraints, m^- is the number of artificial variables, NAR is the non-acute constraint relaxation and AN is the non-acute constraint reinsertion.

Here, we found that the number of dimensions of parameters solving by SNAR solved is less than or equal to the number of dimensions of the original problem while the number of dimensions of the problem solving by Two-Phase method is greater than the original problem.

3.4.2 Dual SNAR vs Two-Phase Method

Recall the standard form of Dual SNAR.

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{3.47}$$

where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{c}$ is an n-dimensional vector and $\mathbf{b}$ is an m-dimensional vector. For the two-phase method, by adding m artificial variables, the simplex method can start. Therefore, phase-I can be written as follows:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{1}^{\mathrm{T}}\mathbf{x}_a \\ \text{subject to} \quad & \mathbf{Ax} + \mathbf{x_a} = \mathbf{b} \\ & \mathbf{x}, \mathbf{x_a} \geq \mathbf{0}, \end{aligned} \tag{3.48}$$

Consider the number of dimensions of the problem solved by SNAR, it will be expanded as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c^T x} \\ \text{subject to} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & -\mathbf{Ax} \leq -\mathbf{b} \\ & -\mathbf{x} \leq \mathbf{0}. \end{aligned} \tag{3.49}$$

The number of constraints increases to $2m + n$ constraints and the number of variables is n while the dual of this standard form which is defined by:

$$\begin{aligned} \text{-Maximize} \quad & -\mathbf{b^T w} \\ \text{subject to} \quad & -\mathbf{A^T w} \leq -\mathbf{c}, \end{aligned} \tag{3.50}$$

the number of constraints is n and the number of variables is m. Therefore, from the subsection 2.4.1, the number of dimensions of SNAR, Dual SNAR and two-phase method will be summarized in Table 3.2.

Table 3.2: Comparison the number of dimensions between SNAR, Dual SNAR and Two-Phase Method

Method		The number of dimensions
SNAR	NAR	$(m_1 \times (2n + m_1)) + 2n + 2m_1$
	AN	$((2m + n) \times (3n + 2m)) + 4m + 4n$
Dual SNAR	NAR_D	$(n_1 \times (2m + n_1)) + 2m + 2n_1$
	AN_D	$(n \times (2m + n)) + 2m + 2n$
Two-Phase Method	Phase I	$(m \times (n + m)) + 2m + n$
	Phase II	$(m \times (n + m)) + 2m + n$

where n_1 is the number of the acute constraints in the dual problem, NAR_D is the non-acute constraint relaxation and AN_D is the non-acute constraint reinsertion of the dual problem.

Here, we found that the number of dimensions of the problem by SNAR solved is greater than the dimension of the original problem solving by Dual SNAR and Two-Phase method while the number of dimensions of the problem solving by Dual SNAR is likely as the number of dimensions of of the problem solving by the Two-Phase method depend on n_1.

CHAPTER IV
EXPERIMENTAL RESULTS

In this Chapter, we will describe problems designed to test our algorithms. The computational results are proposed and summarized. Finally, we will analyze our results and findings.

4.1 Experimental Designs

Since our algorithms: SNAR and Dual SNAR, were designed to suit for solving different problem structures, test problems were differently designed. Randomly generated linear programming test problems for comparing between SNAR, Two-Phase method and Arsham's method were called ***Problem P***, and randomly generated linear programming test problems for comparing between SNAR, Dual SNAR, Two-Phase method and Arsham's method were called ***Problem D***.

4.1.1 Problem P

We tested SNAR based on simulated linear programming problems. Randomly generated linear programming test problems

- are maximization problems;

- have a vector $\mathbf{c}$ with $c_i \in [-9, 9], i = 1, 2, ..., n$;

- have a matrix $\mathbf{A}$ with $a_{ij} \in [-9, 9], i = 1, 2, ..., m, j = 1, 2, ..., n$;

- have a vector $\mathbf{x}$ with $x_i \in [-9, 9], i = 1, 2, ..., n$;

Then we derive a vector $\mathbf{b}$ with $b_i = \mathbf{A}_{i:}\mathbf{x}$ where $i \in \{1, 2, ..., n\}$ and $b_j = \mathbf{A}_{j:}\mathbf{x} + 1$ to guarantee feasibility where $j \in \{n + 1, n + 2, ..., m\}$. Junior and Lins [21] used these interval to generate parameters for testing their algorithm.

We found that approximately 50% of the number of constraints are negative and approximately 50% constraints are non-acute.

The different sizes of the number of variables (n) and the number of constraints (m) were tested with SNAR, Two-Phase method and Arsham's method[15, 16] where $m \geq n$, $n \in \{5, 10, 20, 50, 100\}$ and m depended on sizes of variables as $m \in \{1n, 2n, 5n, 10n, 20n, 30n, 40n, 50n\}$.

Problems are in form of the problem (3.1) having n variables, m constraints, about 50% of "$\leq$" constraints ($\approx \frac{m}{2}$) or $O(m)$ space and RHS are negative. So Two-Phase method required approximately to add $\frac{m}{2}$ artificial variables. By comparing the number of dimension of problem as the table 3.1, the number of dimension of problem for Two-Phase method has ($\approx(m \times (2n + m + \frac{m}{2}))+ 2n+2m+\frac{m}{2}$) $O(mn+m^2)$. While the number of dimension of problem for NAR will be ($\approx(\frac{m}{2} \times (2n+\frac{m}{2}))+ 2n+m$) $O(mn+m^2)$ and the dimension of parameters of the last tableau will be $(m \times (2n + m))+ 2n + 2m$.

Arsham's method deals with the linear programming problems of the same dimension as SNAR.

For a small number of constraints with respect to n ($m \in \{n, 2n\}$) depended on n, relaxed problems solved by SNAR had unbounded optimal solutions while, for a large number of constraints with respect to $n(m \in \{5n, 10n, 20n, 30n, 40n, 50n\})$, relaxed problems solved by SNAR had optimal solutions. So we separate computational results into two sections: a small number of constraints and a large number of constraints.

4.1.2 Problem D

Dual SNAR were tested by simulated linear programming problems in the standard form. The randomly generated linear programming test problems

- are maximization problems;

- have a vector $\mathbf{c}$ with $c_i \in [-9, 9], i = 1, 2, ..., n$;

- have a matrix $\mathbf{A}$ with $a_{ij} \in [-9, 9], i = 1, 2, ..., m, j = 1, 2, ..., n$;

- have a vector $\mathbf{x}$ with $x_i \in [0, 9], i = 1, 2, ..., n$;

Then we derive a vector $\mathbf{b}$ with $\mathbf{b} = \mathbf{Ax}$ for guarantee the feasibility.

The different sizes of the number of variables (n) and the number of constraints (m) were tested with SNAR, Dual SNAR, Two-Phase method and Arsham's method where $m \geq n$, $n \in \{5, 10, 20\}$ and m depended on sizes of variables as $m \in \{1n, 2n, 5n, 10n, 20n, 30n, 40n, 50n\}$. We did not test algorithms with sizes $n \in \{50, 100\}$ since we could see the trend of the average number of iterations and the average running time from solving the smaller sizes.

4.2 Computational Results

According to a designed problem, the average number of iterations and the average running time were kept to compare efficiency. Since we had two collections of problems, the computational results were divided into two subsections: computational results on problems **P** and computational results on problems **D**.

4.2.1 Computational Results on Problems P

A Small Number of Constraints

Firstly, we report the comparison of the average number of iterations between SNAR, Two-Phase method and Arsham's method for a small number of constraints as in Table 4.1. Description of this table were shown in Table 4.2. Then, we report the average running time in the table 4.3, and ratios of the average number of iterations and the average running time by Two-Phase method to SNAR and by Arsham's method to SNAR were shown in Table 4.4. Then, the average number of iterations and the average running time were plotted.

Table 4.1: The average number of iterations between SNAR, Two-Phase method and Arsham's method for a small number of constraints

		SNAR				Two-Phase Method				Arsham's Method			
m	n	NAR	AN	NAR+AN	SD_1	PhaseI	PhaseII	PhaseI+II	SD_2	RP	$\geq$	RP+$\geq$	SD_3
5	5	1.91	2.45	**4.36**	*2.52*	5.20	2.42	7.62	3.35	1.20	3.78	4.98	2.95
10	5	4.14	3.60	**7.74**	*3.82*	10.39	3.51	13.90	4.50	2.18	8.53	10.71	5.63
10	10	3.98	5.55	**9.53**	*4.20*	12.62	4.78	17.40	6.33	2.86	10.52	13.38	6.43
20	10	9.04	17.21	**26.25**	11.61	26.21	6.70	32.91	*9.77*	4.98	31.41	36.39	16.46
20	20	8.71	14.10	**22.81**	*10.43*	28.11	8.11	36.22	10.67	5.58	28.30	33.88	11.81
40	20	20.68	66.94	87.62	34.69	61.93	18.48	**80.41**	*18.80*	10.91	110.34	121.25	46.07
50	50	24.79	69.37	**94.16**	33.83	79.47	15.07	94.54	*21.53*	16.06	112.63	128.69	43.29
100	50	58.64	369.92	428.56	136.90	181.95	46.40	**228.35**	*45.08*	32.90	549.48	582.38	163.68
100	100	50.66	222.17	272.83	70.61	177.05	29.16	**206.21**	*42.48*	37.57	318.88	356.45	94.60
200	100	127.69	1145.84	1273.53	306.97	411.36	101.59	**512.95**	*80.72*	77.22	1884.85	1962.07	448.18

In table 4.1, the boldface numbers identify the smallest average number of iterations and the italic numbers identify that the smallest standard deviations of iterations for solving linear programming problems of the same size. The description of columns in table 4.1 are shown in table 4.2.

Table 4.2: Description of the columns in table 4.1

NAR	The average number of iterations of the non-acute constraint relaxation
AN	The average number of iterations of the non-acute constraint reinsertion
NAR+AN	The summation of the average number of iterations of the non-acute constraint relaxation and the non-acute constraint reinsertion
SD_1	The standard deviation of the number of iterations of the SNAR algorithm
Phase I	The average number of iterations of Phase I
Phase II	The average number of iterations of Phase II
Phase I+II	The summation of the average number of iterations of Phase I and Phase II
SD_2	The standard deviation of the number of iterations of Two-Phase method
RP	The average number of iterations of the relaxed problem in Arsham's method
$\geq$	The average number of iterations of $\geq$ constraints reinsertion
RP+$\geq$	The summation of the average number of iterations of the relaxed problem and the average number of iterations of $\geq$ constraints reinsertion in Arsham's method
SD_3	The standard deviation of the number of iterations of Arsham's method

Then, the average running time to solve unbounded problems including their standard deviations are reported in the following table:

Table 4.3: The average running time between SNAR, Two-Phase method and Arsham's method for a small number of constraints

		SNAR		Two-Phase Method		Arsham's Method	
m	n	time (sec.)	SD	time (sec.)	SD	time (sec.)	SD
5	5	**0.00076**	0.00044	0.00090	*0.00033*	0.00088	0.00039
10	5	0.00171	*0.00072*	**0.00161**	0.00076	0.00184	0.00082
10	10	0.00200	0.00080	**0.00197**	*0.00073*	0.00231	0.00096
20	10	0.00539	0.00174	**0.00415**	*0.00148*	0.00651	0.00216
20	20	0.00594	0.00276	**0.00540**	*0.00257*	0.00774	0.00269
40	20	0.02352	0.00675	**0.01686**	*0.00428*	0.02940	0.00879
50	50	0.05020	0.02045	**0.03430**	*0.01273*	0.06260	0.02334
100	50	0.26800	0.06709	**0.16110**	*0.03423*	0.34040	0.08055
100	100	0.30590	0.06721	**0.22010**	*0.05241*	0.37610	0.086338
200	100	2.49010	0.54057	**1.33920**	*0.21153*	3.58020	0.70940

In table 4.3, the boldface numbers identify the smallest average running time and the italic numbers identify the smallest standard deviations of the running time for solving linear programming problems of the same size.

Table 4.4: Ratios of the average number of iterations and the average running time by Two-Phase method to SNAR and by Arsham's method to SNAR

		Ratio of iterations		Ratio of running time	
m	n	2-Phase/SNAR	Arsham/SNAR	2-Phase/SNAR	Arsham/SNAR
5	5	1.75	1.14	1.19	1.15
10	5	1.80	1.38	0.94	1.08
10	10	1.83	1.40	0.99	1.16
20	10	1.25	1.39	0.77	1.21
20	20	1.59	1.49	0.91	1.30
40	20	0.92	1.38	0.72	1.25
50	50	1.00	1.37	0.68	1.25
100	50	0.53	1.36	0.60	1.27
100	100	0.76	1.31	0.72	1.23
200	100	0.40	1.54	0.54	1.44

The results in table 4.1 and 4.3 are plotted as the following figures.

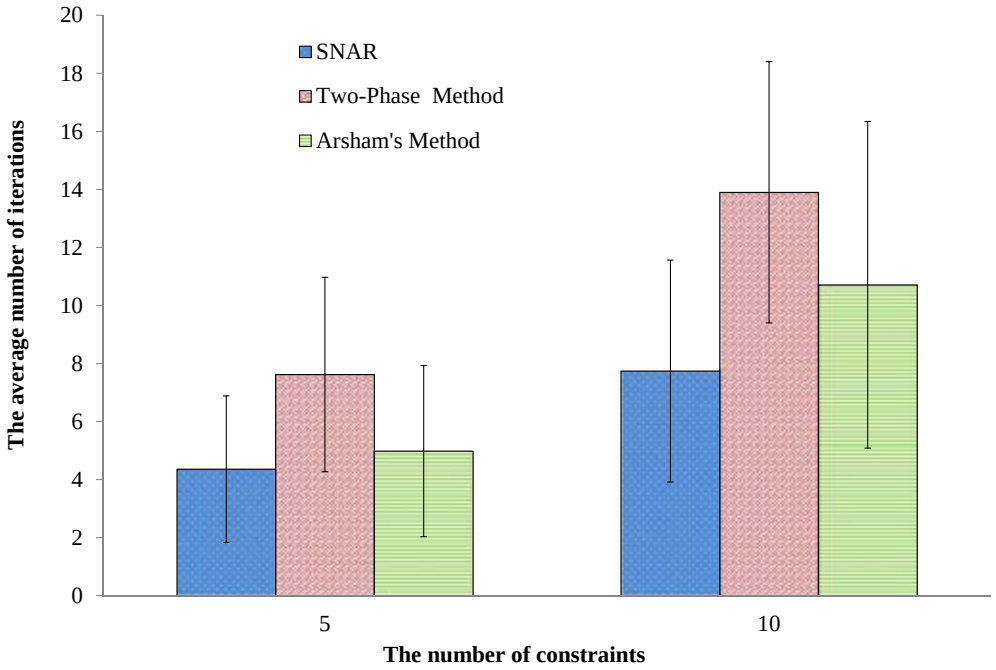


Figure 4.1: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 5 variables

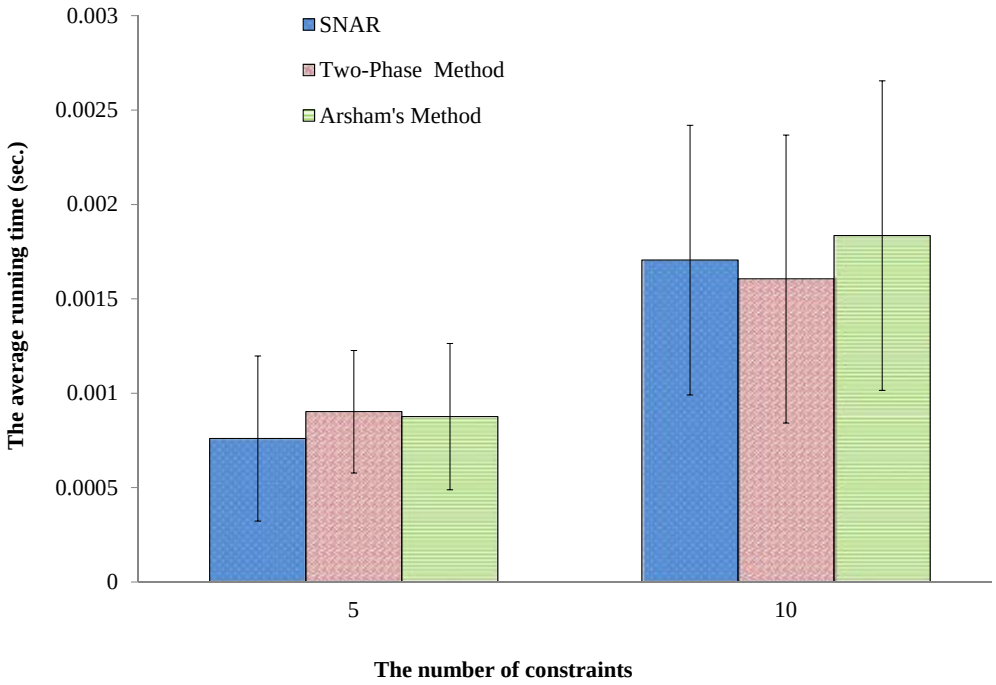


Figure 4.2: The average running time solved by SNAR, Two-Phase method and Arsham's method for 5 variables

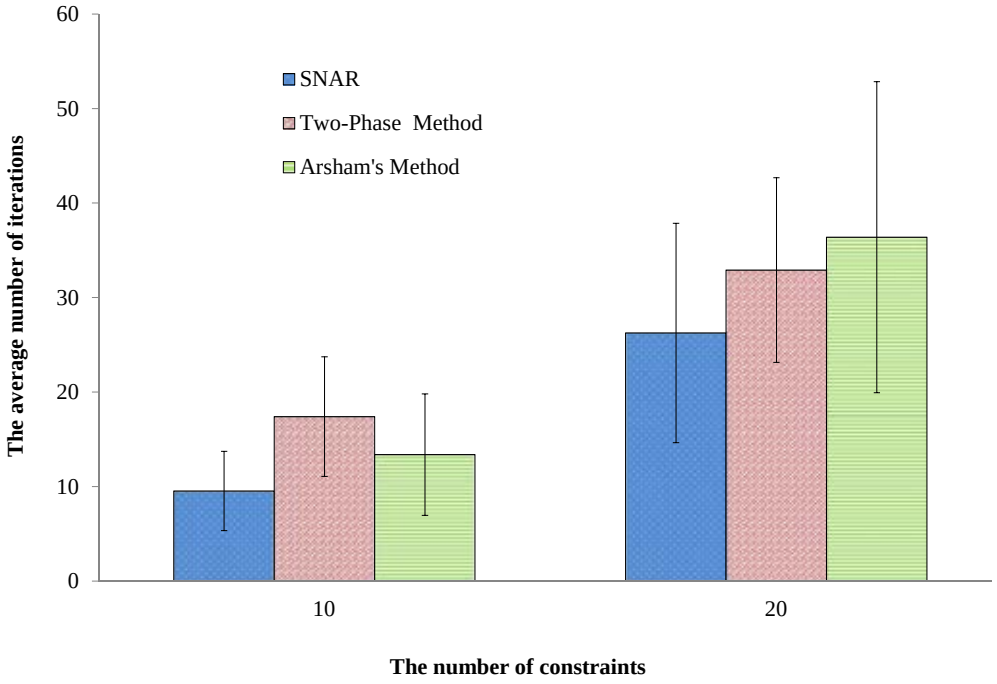


Figure 4.3: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 10 variables

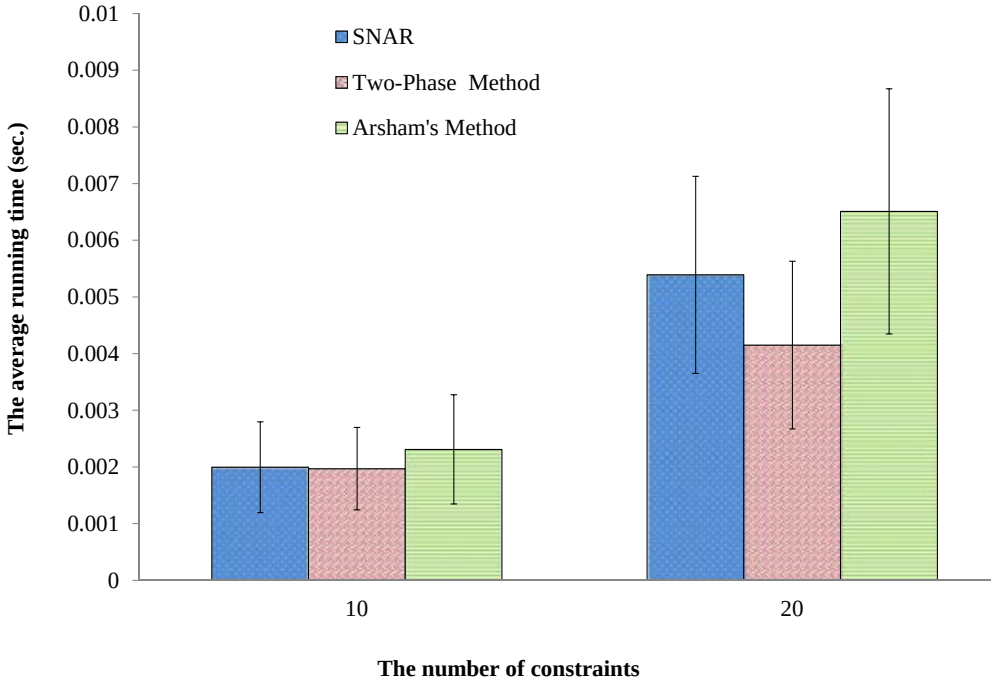


Figure 4.4: The average running time solved by SNAR, Two-Phase method and Arsham's method for 10 variables

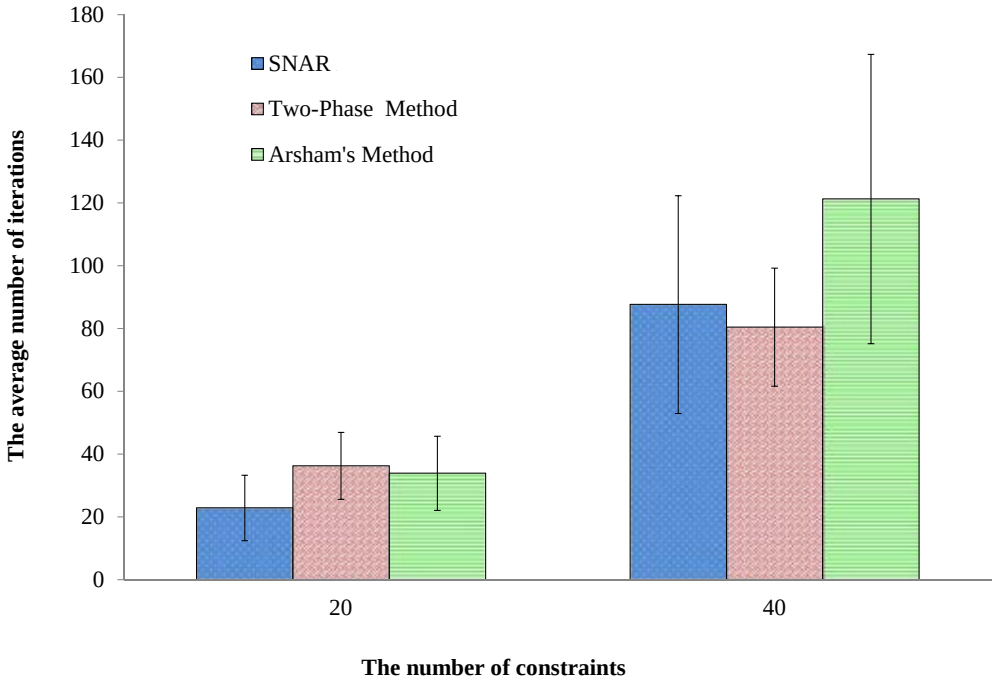


Figure 4.5: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 20 variables

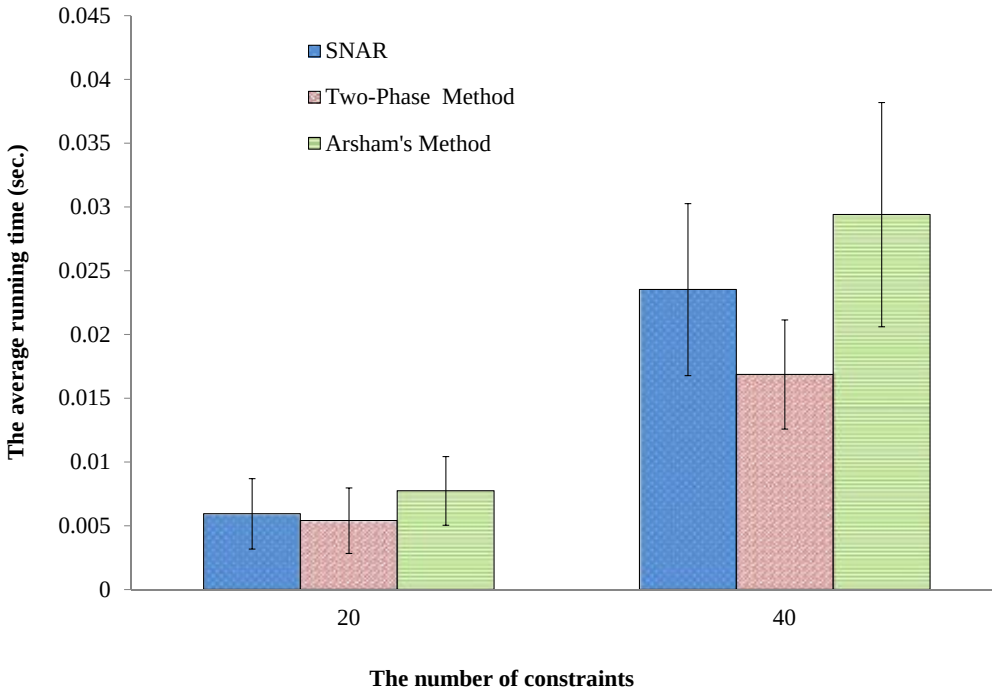


Figure 4.6: The average running time solved by SNAR, Two-Phase method and Arsham's method for 20 variables

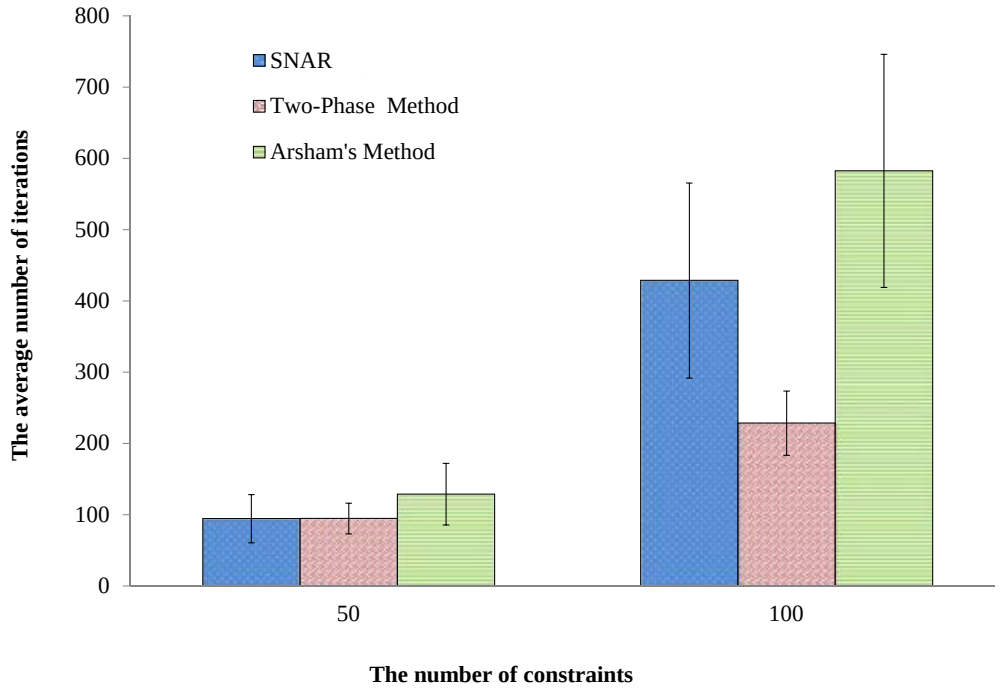


Figure 4.7: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 50 variables

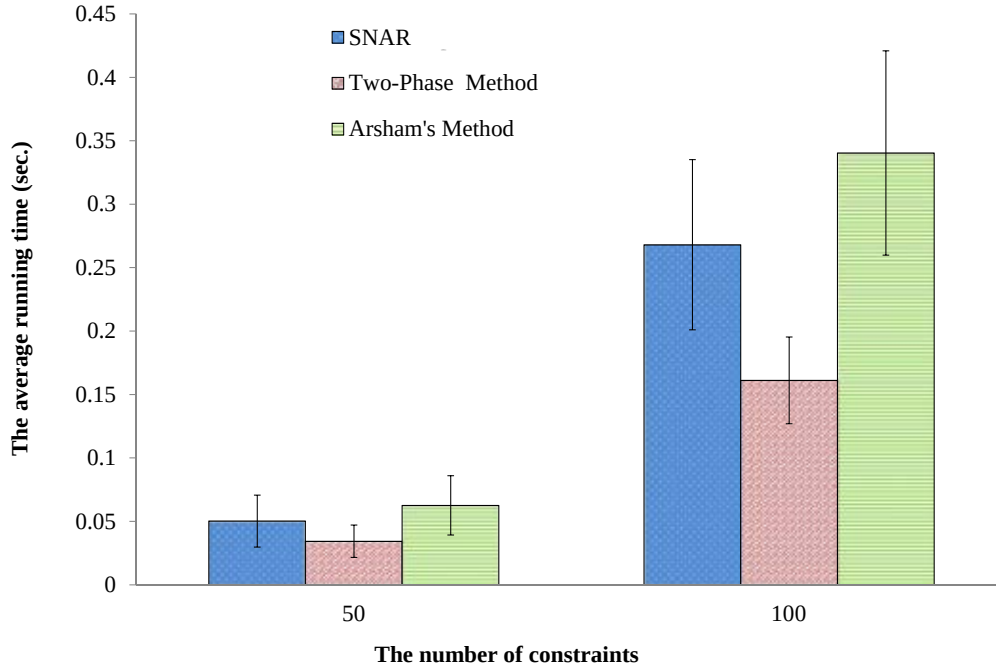


Figure 4.8: The average running time solved by SNAR, Two-Phase method and Arsham's method for 50 variables

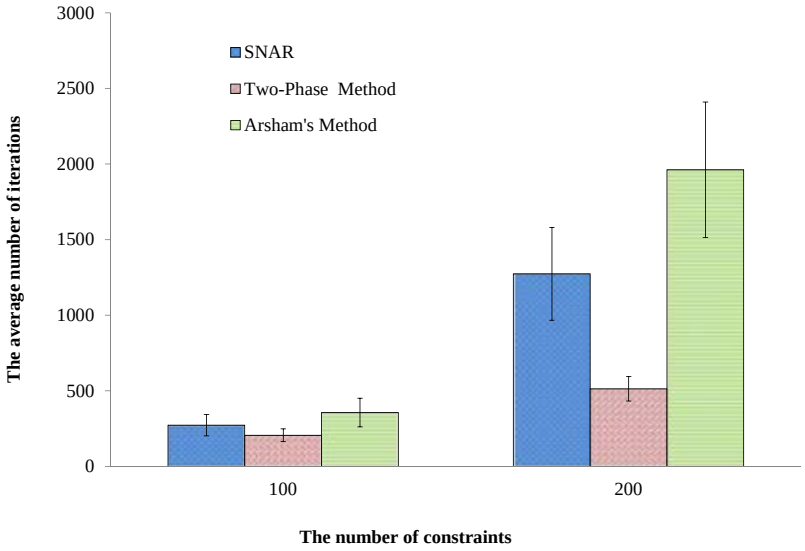


Figure 4.9: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 100 variables

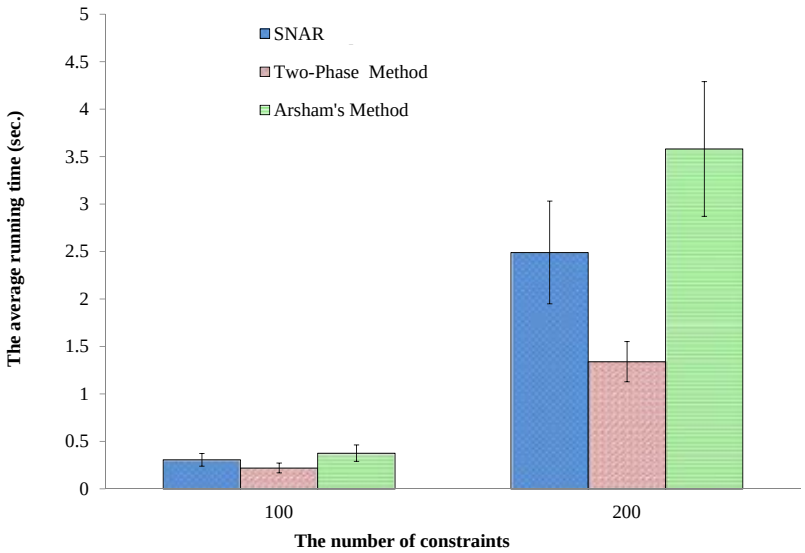


Figure 4.10: The average running time solved by SNAR, Two-Phase method and Arsham's method for 100 variables

A Large Number of Constraints

The average number of iterations and the average running time for solving problems with a large number of constraints are shown in Tables 4.5, 4.6 and 4.7. Then, ratios of the average number of iterations and the average running time are shown in Table 4.4.

Table 4.5: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 5, 10 and 20 variables

		SNAR				Two-Phase Method				Arsham's Method			
m	n	NAR	AN	NAR+AN	SD_1	PhaseI	PhaseII	PhaseI+II	SD_2	RP	$\geq$	RP+$\geq$	SD_3
25	5	8.67	1.93	**10.60**	*2.81*	20.28	3.97	24.25	4.46	4.90	14.03	18.93	6.25
50	5	10.99	1.43	**12.42**	*2.59*	36.13	4.46	40.59	7.24	6.20	19.78	25.98	8.86
100	5	14.33	1.37	**15.70**	*3.33*	63.51	5.52	69.03	9.59	7.90	18.14	26.04	10.55
150	5	15.00	1.24	**16.24**	*3.14*	90.90	6.02	96.92	12.02	8.99	21.59	30.58	13.69
200	5	16.42	1.29	**17.71**	*3.68*	122.14	6.53	128.67	15.38	8.89	24.84	33.73	13.43
250	5	17.76	1.38	**19.14**	*3.94*	149.18	6.17	155.35	14.31	9.40	25.96	35.36	15.13
50	10	20.79	7.07	**27.86**	*6.85*	48.20	10.15	58.35	8.26	10.19	59.10	69.29	18.06
100	10	28.45	4.25	**32.70**	*5.15*	86.87	12.72	99.59	13.50	13.53	68.12	81.65	26.80
200	10	34.69	3.51	**38.20**	*5.66*	150.81	14.31	165.12	18.73	17.55	74.56	92.11	35.94
300	10	38.63	5.25	**43.88**	*7.58*	212.81	14.91	227.72	24.11	20.52	76.24	96.76	41.83
400	10	41.55	4.31	**45.86**	*6.76*	272.93	16.31	289.24	28.08	18.80	95.48	114.28	41.52
500	10	44.03	4.75	**48.78**	7.46	328.08	17.22	345.30	27.72	23.24	81.08	104.32	41.55
100	20	51.23	20.27	**71.50**	24.97	120.54	27.07	147.61	*17.72*	22.72	228.51	251.23	55.30
200	20	69.28	10.20	**79.48**	*10.49*	205.47	35.58	241.05	26.91	31.14	254.76	285.90	81.51
400	20	88.15	10.55	**98.70**	*13.05*	359.10	40.11	399.21	35.43	37.92	279.33	317.25	109.80
600	20	98.66	10.77	**109.43**	*12.19*	500.45	42.69	543.14	49.14	42.46	298.23	340.69	117.65
800	20	104.87	11.31	**116.18**	*12.63*	645.40	44.99	690.39	52.09	49.52	303.24	352.76	125.74
1000	20	112.34	11.12	**123.46**	*12.97*	772.61	44.10	816.71	67.38	49.96	300.93	350.89	126.99

Table 4.6: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 50 and 100 variables

		SNAR				Two-Phase Method				Arsham's Method			
m	n	NAR	AN	NAR+AN	SD_1	PhaseI	PhaseII	PhaseI+II	SD_2	RP	$\geq$	RP+$\geq$	SD_3
250	50	172.5	51.19	**223.69**	*24.83*	405.39	103.88	509.27	44.04	67.38	1510.42	1577.8	242.69
500	50	237.82	42.41	**280.23**	*25.99*	686.87	132.06	818.93	58.35	85.93	1724.21	1810.14	363.73
1000	50	305.74	41.25	**346.99**	*28.18*	1164.99	154.85	1319.84	90.34	112.26	1717.33	1829.59	456.07
1500	50	340.18	42.09	**382.27**	*30.31*	1614.14	162.56	1776.70	114.84	126.39	1680.94	1807.33	489.22
2000	50	366.02	40.90	**406.92**	*35.55*	2063.34	173.49	2236.83	168.68	142.71	1820.62	1963.33	996.88
2500	50	384.22	41.23	**425.45**	*35.66*	2502.03	179.56	2681.59	212.32	146.43	1795.97	1942.40	535.65
500	100	475.54	142.33	**617.88**	*51.44*	1077.71	293.92	1371.63	78.42	159.83	6640.71	6800.54	1215.04
1000	100	624.21	119.99	**744.20**	*64.36*	1755.44	355.94	2111.38	138.24	189.67	7340.59	7530.26	1067.89
2000	100	793.40	103.69	**897.09**	*64.13*	2927.42	430.28	3357.70	226.21	234.63	6815.52	7050.15	1272.25
3000	100	906.13	106.80	**1012.93**	*75.03*	4102.53	498.27	4600.80	230.59	287.33	6863.40	7150.73	1196.67
4000	100	950.33	96.17	**1046.50**	*56.63*	5458	474.67	5932.67	160.90	436.33	5674.67	6111	1764.78
5000	100	1000.56	126.56	**1127.11**	*61.58*	6109.11	504.44	6613.56	343.01	302.33	7105.11	7407.44	1712.84

In Table 4.5 and 4.6, the boldface numbers identify the smallest average number of iterations and the italic numbers identify the smallest standard deviations of iterations for solving linear programming problems of the same sizes.

Table 4.7: The average running time solved by SNAR, Two-Phase method and Arsham's method

		SNAR		Two-Phase Method		Arsham's Method	
m	n	time (sec.)	SD	time (sec.)	SD	time (sec.)	SD
25	5	**0.0017**	*0.0008*	0.0040	0.0013	0.0046	0.0014
50	5	**0.0035**	*0.0009*	0.0093	0.0022	0.0135	0.0023
100	5	**0.0119**	*0.0026*	0.0375	0.0066	0.0494	0.0055
150	5	**0.0286**	*0.0032*	0.0989	0.0145	0.1261	0.0114
200	5	**0.0616**	*0.0087*	0.2442	0.0368	0.2681	0.0244
250	5	**0.1212**	*0.0122*	0.4474	0.0527	0.4955	0.0272
50	10	**0.0058**	*0.0021*	0.0142	0.0032	0.0209	0.0035
100	10	**0.0164**	*0.0030*	0.0554	0.0084	0.0674	0.0090
200	10	**0.0785**	*0.0141*	0.3197	0.0448	0.3194	0.0349
300	10	**0.2403**	*0.0181*	0.9308	0.1245	0.9039	0.0681
400	10	**0.5037**	*0.0281*	2.1378	0.2751	1.9202	0.1458
500	10	**0.9640**	*0.0519*	4.7602	0.4823	3.3199	0.2088
100	20	**0.0350**	*0.0127*	0.0925	0.0175	0.1463	0.0235
200	20	**0.1148**	*0.0159*	0.5004	0.0617	0.5075	0.0648
400	20	**0.6218**	*0.0471*	3.0493	0.3470	2.4188	0.2850
600	20	**2.0391**	*0.1049*	10.7710	1.1896	7.0536	0.6880
800	20	**5.3593**	*0.2456*	24.8665	2.5508	16.0027	1.3772
1000	20	**10.5502**	*0.4020*	45.9179	5.1068	29.6589	2.1707
250	50	**0.4455**	*0.0481*	1.7643	0.2022	2.7856	0.3727
500	50	**1.9987**	*0.1856*	12.2450	1.1287	10.9249	1.5327
1000	50	**14.6223**	*0.9170*	77.1635	7.0560	58.8373	7.7648
1500	50	**46.0241**	*1.9569*	257.0604	22.7486	163.2123	19.5543
2000	50	**107.9391**	*4.5046*	599.3633	60.0334	354.7317	40.1074
2500	50	**218.2526**	*7.2408*	1192.7648	126.0117	680.5636	66.4993
500	100	**4.8388**	*0.4453*	23.3188	1.9957	41.4612	7.1329
1000	100	**24.5372**	*2.0704*	131.4173	12.3284	179.1298	22.6895
2000	100	**150.5739**	*8.9914*	933.6161	81.9089	793.9966	96.3361
3000	100	**485.3817**	*25.7367*	3111.8113	213.7521	2216.7947	211.6782
4000	100	**1108.7317**	*25.0011*	7395.9633	323.9440	4258.3450	354.2545
5000	100	**2128.4811**	*83.0332*	12761.9222	957.3011	8470.2667	1126.8516

Table 4.8: Ratios of the average number of iterations and the average running time by Two-Phase method to SNAR and by Arsham's method to SNAR

		Ratio of iterations		Ratio of running time	
m	n	2-Phase/SNAR	Arsham/SNAR	2-Phase/SNAR	Arsham/SNAR
25	5	2.29	1.79	2.36	2.75
50	5	3.27	2.09	2.69	3.91
100	5	4.40	1.66	3.16	4.17
150	5	5.97	1.88	3.45	4.40
200	5	7.27	1.90	3.96	4.35
250	5	8.12	1.85	3.69	4.09
50	10	2.09	2.49	2.45	3.60
100	10	3.05	2.50	3.38	4.12
200	10	4.32	2.41	4.07	4.07
300	10	5.19	2.21	3.87	3.76
400	10	6.31	2.49	4.24	3.81
500	10	7.08	2.14	4.94	3.44
100	20	2.06	3.51	2.64	4.18
200	20	3.03	3.60	4.36	4.42
400	20	4.04	3.21	4.90	3.89
600	20	4.96	3.11	5.28	3.46
800	20	5.94	3.04	4.64	2.99
1000	20	6.62	2.84	4.35	2.81
250	50	2.28	7.05	3.96	6.25
500	50	2.92	6.46	6.13	5.47
1000	50	3.80	5.27	5.28	4.02
1500	50	4.65	4.73	5.59	3.55
2000	50	5.50	4.82	5.55	3.29
2500	50	6.30	4.57	5.47	3.12
500	100	2.22	11.01	4.82	8.57
1000	100	2.84	10.12	5.36	7.30
2000	100	3.74	7.86	6.20	5.27
3000	100	4.54	7.06	6.41	4.57
4000	100	5.67	5.84	6.67	3.84
5000	100	5.87	6.57	6.00	3.98

The results in Table 4.5, 4.6 and 4.7 are plotted as the following figures.

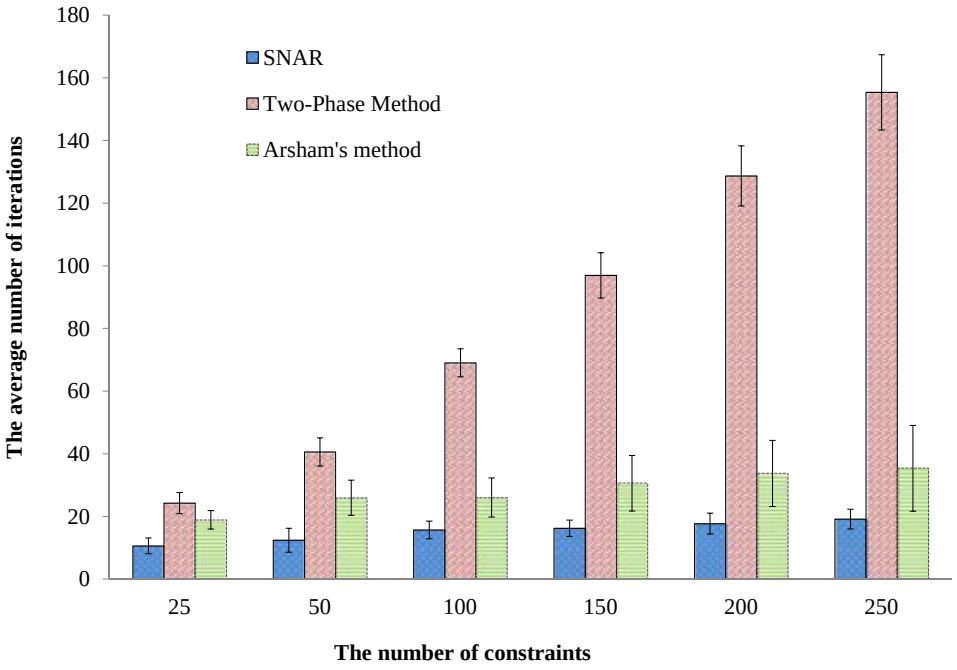


Figure 4.11: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 5 variables

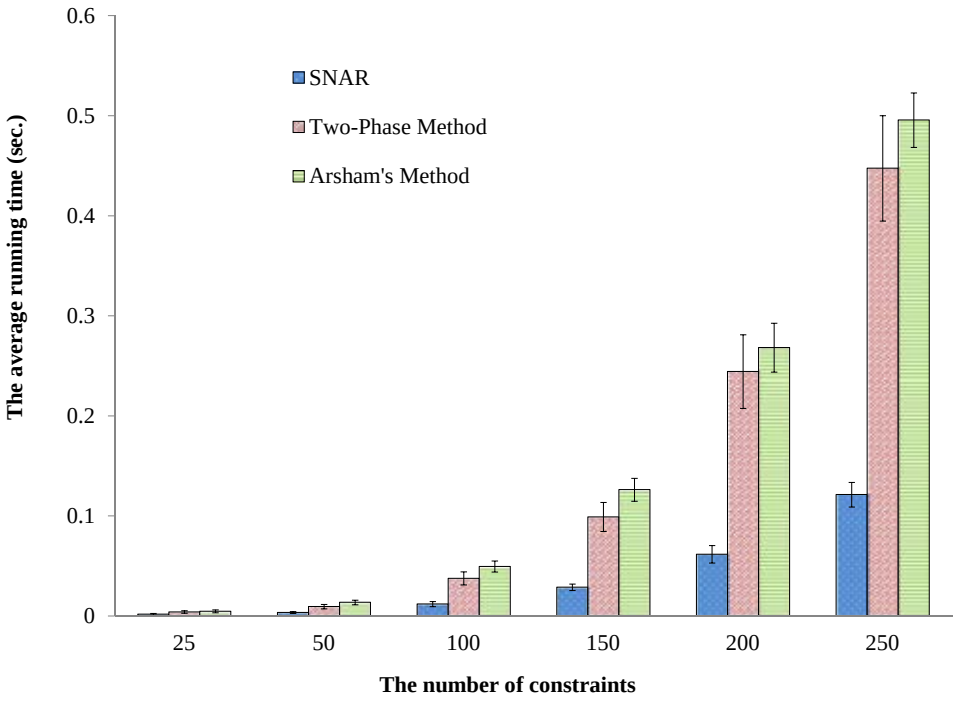


Figure 4.12: The average running time solved by SNAR, Two-Phase method and Arsham's method for 5 variables

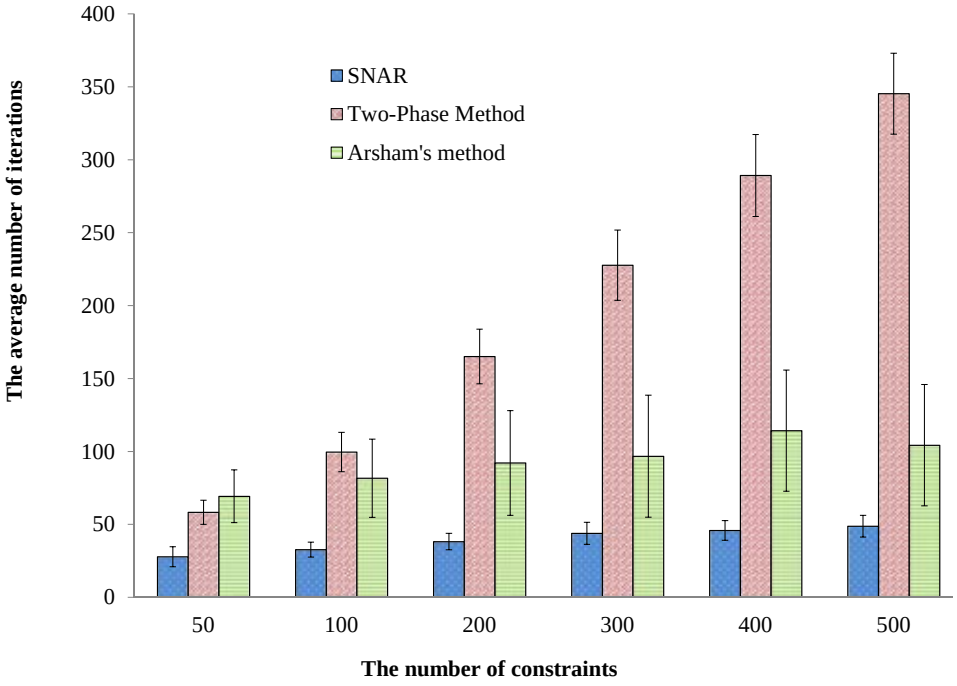


Figure 4.13: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 10 variables

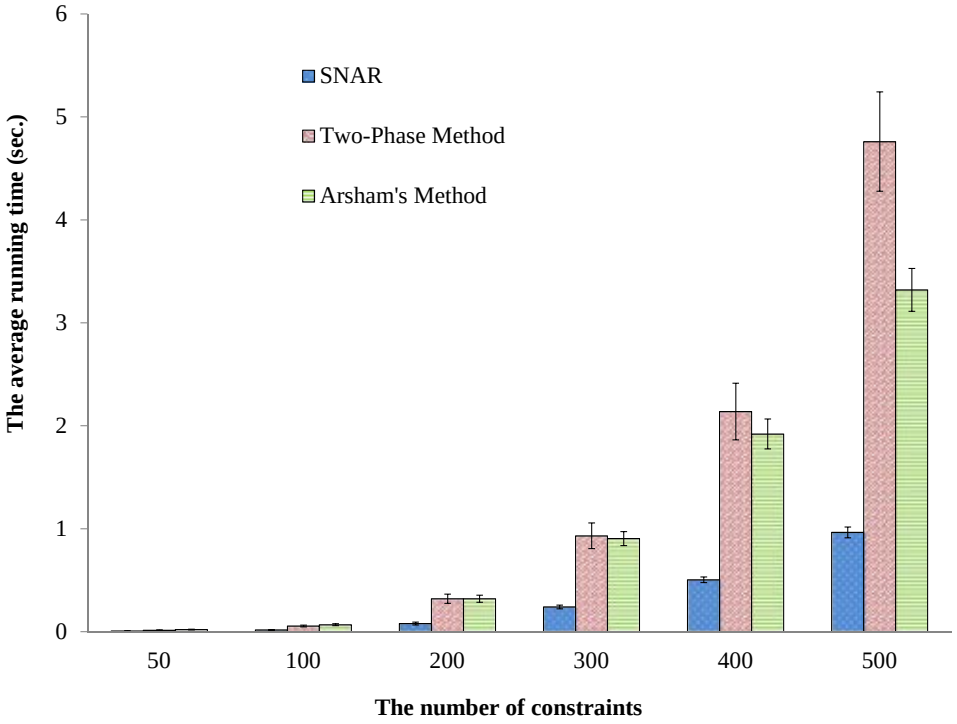


Figure 4.14: The average running time solved by SNAR, Two-Phase method and Arsham's method for 10 variables

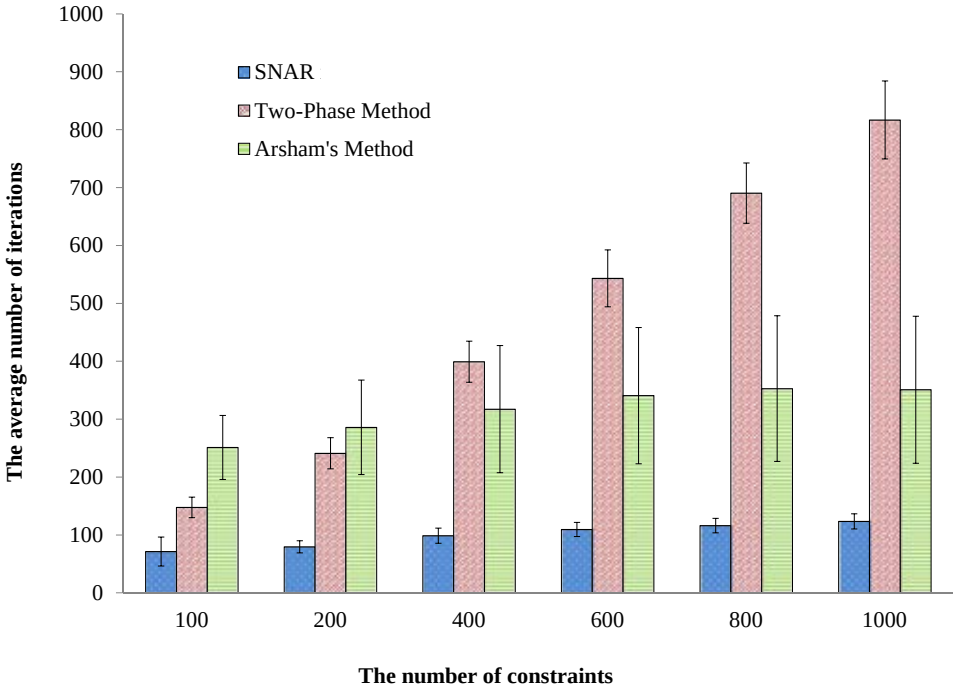


Figure 4.15: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 20 variables

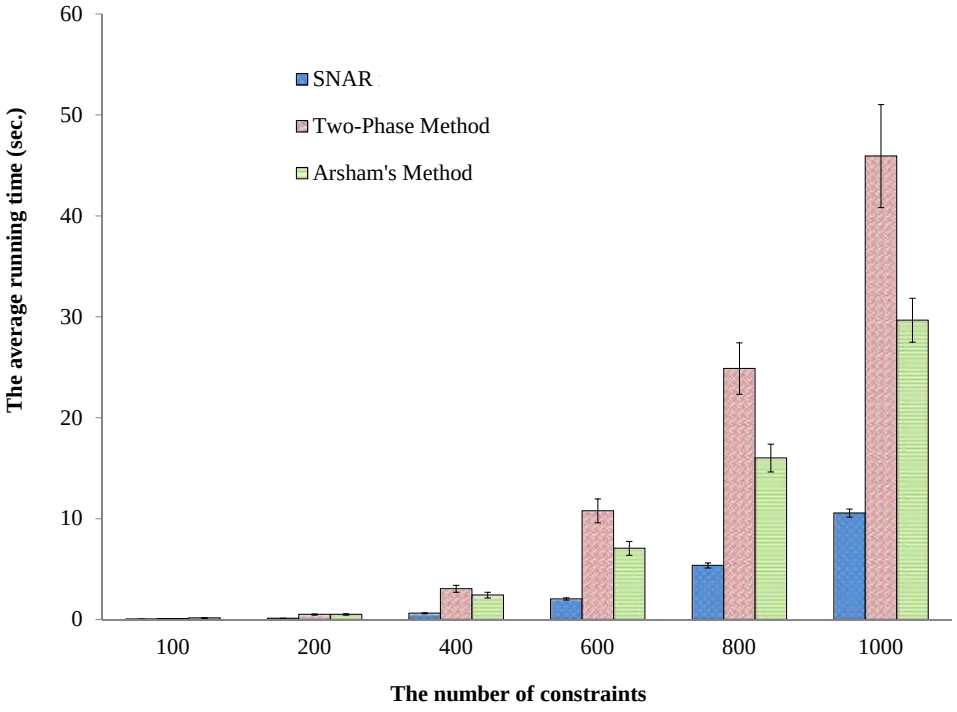


Figure 4.16: The average running time solved by SNAR, Two-Phase method and Arsham's method for 20 variables

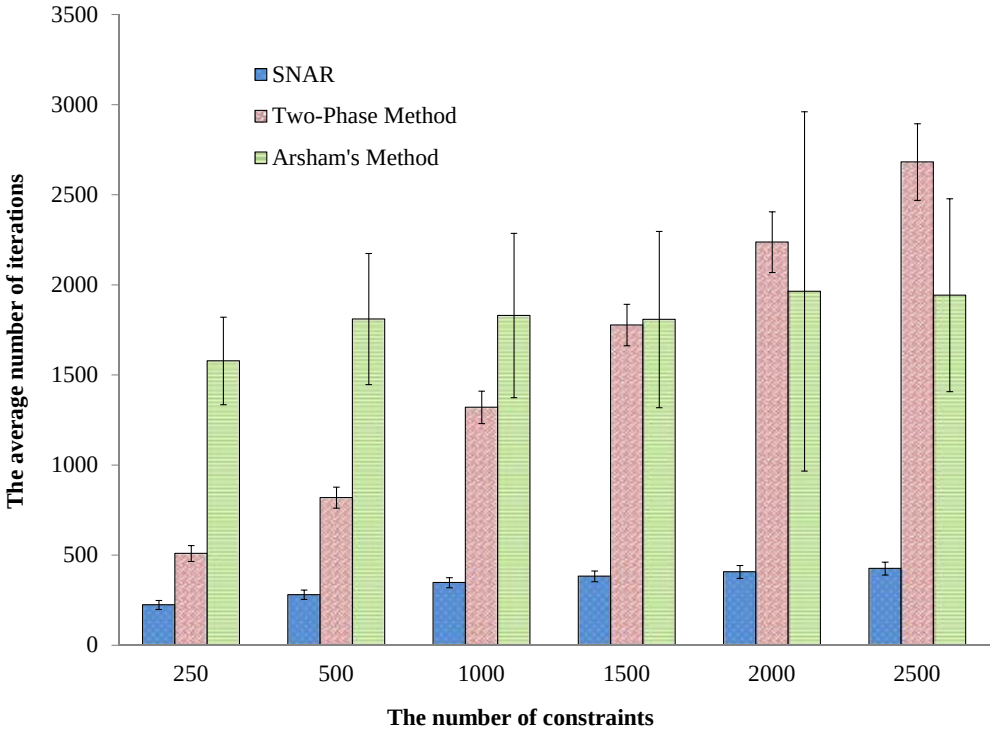


Figure 4.17: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 50 variables

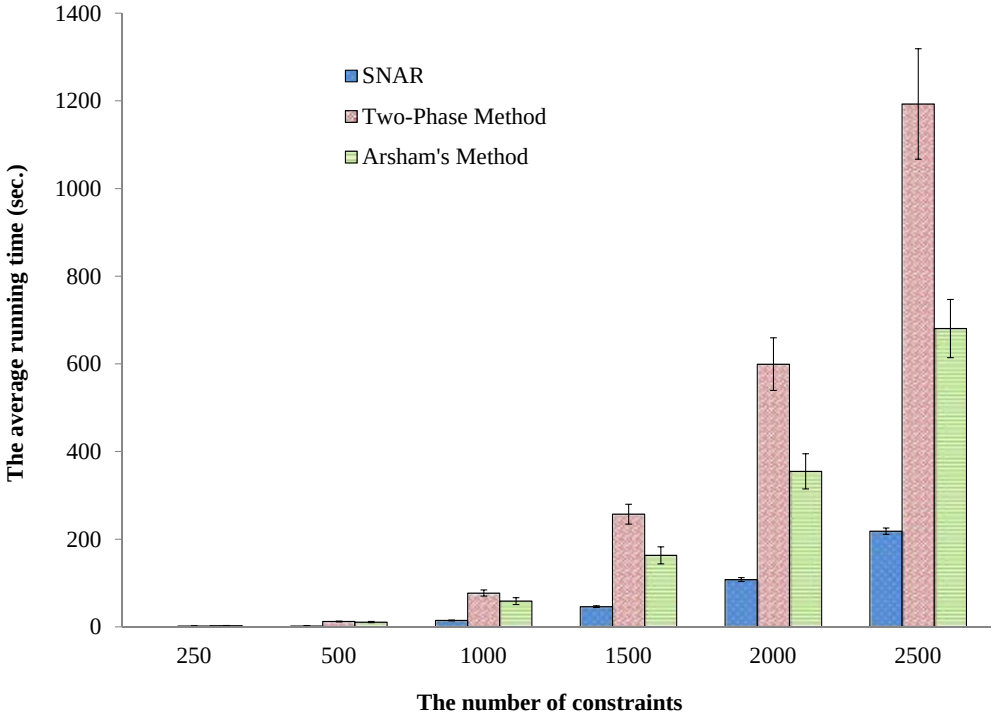


Figure 4.18: The average running time solved by SNAR, Two-Phase method and Arsham's method for 50 variables

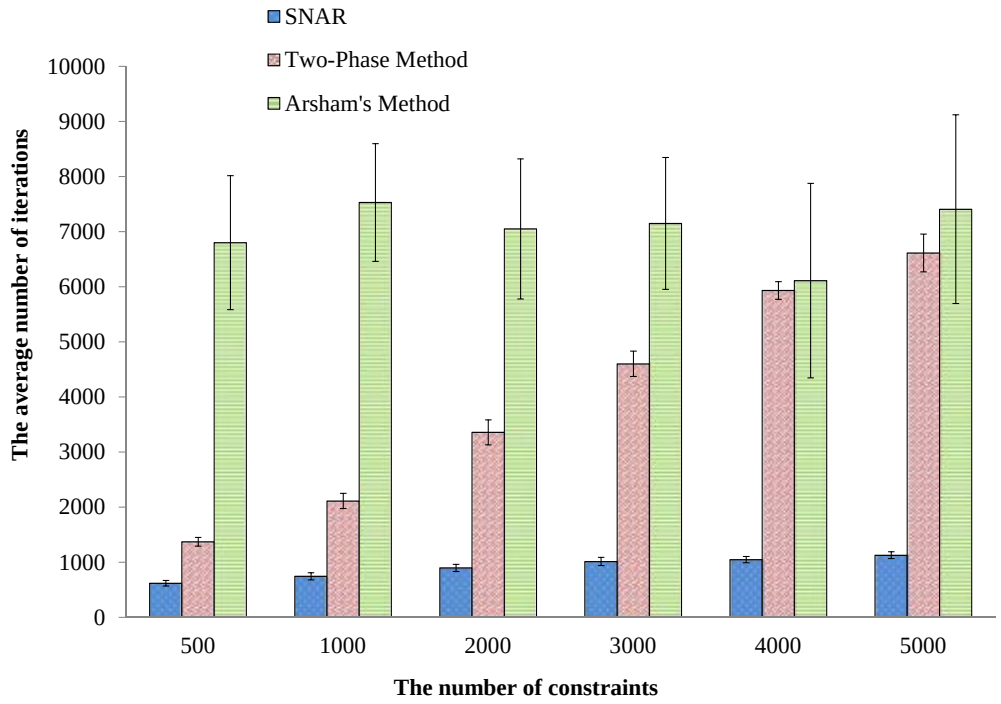


Figure 4.19: The average number of iterations solved by SNAR, Two-Phase method and Arsham's method for 100 variables

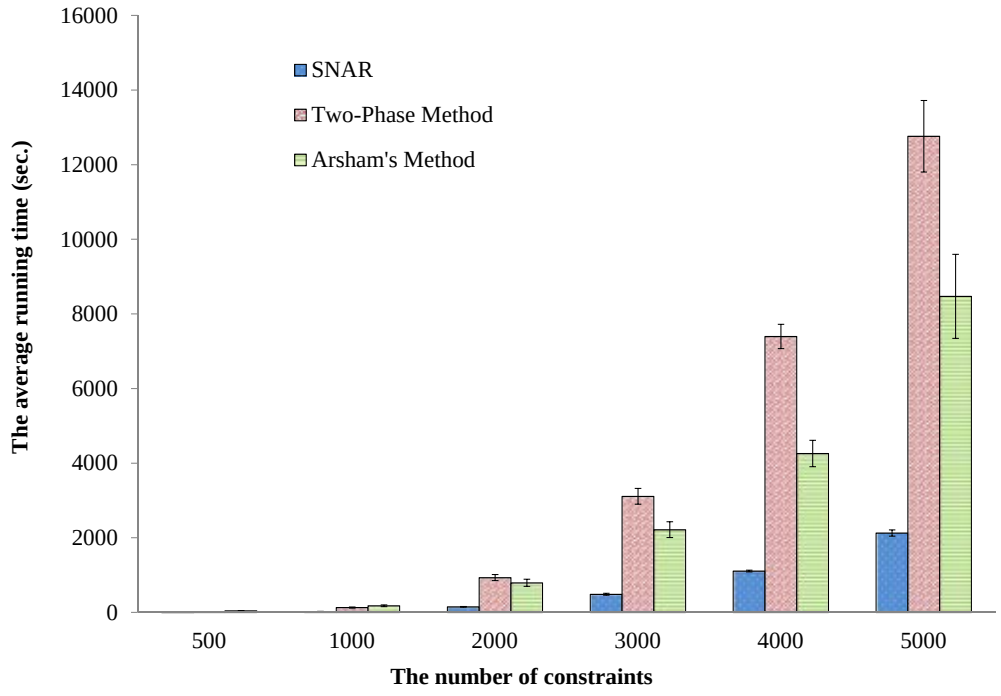


Figure 4.20: The average running time solved by SNAR, Two-Phase method and Arsham's method for 100 variables

4.2.2 Computational Results on Problem D

For comparison, we report only the average number of total iterations of four algorithms and their standard deviations. The average number of iterations of the relaxed problem for SNAR, Dual SNAR and Arsham's method and the average number of iterations of Phase-I for the simplex method were not reported. Then, we report the average running time as the table 4.10, and ratios of the average number of iterations and the average running time solved by SNAR to Dual SNAR, by Two-Phase method to Dual SNAR and by Arsham's method to Dual SNAR are shown in Table 4.11.

Table 4.9: The average number of iterations solved by SNAR, Dual SNAR, Two-Phase method and Arsham's method.

		SNAR		Dual SNAR		Two-Phase Method		Arsham's Method	
m	n	iterations	SD_1	iterations	SD_2	iterations	SD_3	iterations	SD_4
5	5	9.16	2.93	7.79	2.22	**5.08**	*0.27*	7.80	1.98
10	5	8.79	2.92	**6.81**	1.97	7.14	*1.19*	8.07	2.66
25	5	9.44	2.61	**5.95**	1.53	8.41	*1.48*	7.18	1.91
50	5	9.55	2.58	**5.74**	*1.28*	8.73	1.80	7.60	2.31
100	5	9.95	2.62	**5.59**	*1.20*	9.10	2.04	7.54	2.43
150	5	10.38	2.92	**5.46**	*1.08*	8.62	1.90	7.71	2.78
200	5	10.45	3.26	**5.30**	*0.72*	8.93	1.83	7.56	2.24
250	5	10.03	3.05	**5.46**	*0.98*	8.87	2.03	7.66	2.24
10	10	26.93	7.84	21.44	5.63	**10.43**	*0.70*	24.05	6.38
20	10	23.08	8.41	17.51	4.74	**15.69**	*1.98*	23.03	5.85
50	10	24.99	6.82	**16.73**	4.90	20.11	*3.17*	21.29	7.07
100	10	26.00	8.34	**13.97**	4.06	19.82	*3.49*	21.87	12.27
200	10	28.17	8.86	**13.16**	3.85	20.08	*3.06*	20.34	7.72
300	10	26.79	6.98	**12.02**	*2.96*	19.87	3.80	24.46	14.93
400	10	25.33	6.74	**11.95**	*2.41*	19.86	3.21	20.59	8.80
500	10	26.75	7.68	**11.50**	*1.80*	19.93	3.59	22.61	9.31
20	20	97.61	32.65	56.38	13.01	**22.24**	*1.83*	76.32	14.84
40	20	69.25	18.03	44.70	9.71	**36.55**	*4.25*	80.40	16.63
100	20	82.40	26.97	**40.30**	10.59	47.64	*5.80*	75.69	28.78
200	20	91.08	44.27	**34.67**	9.81	48.52	*5.97*	84.46	67.17
400	20	88.75	32.44	**29.63**	8.12	47.78	*6.79*	94.31	85.48
600	20	92.66	36.12	**28.04**	*6.39*	48.46	6.90	98.12	83.41
800	20	92.59	38.30	**25.85**	*4.94*	47.10	6.00	88.02	82.25
1000	20	92.82	32.59	**25.89**	*3.91*	47.43	6.76	96.30	85.75

Table 4.10: The average running time solved by SNAR, Dual SNAR, Two-Phase method and Arsham's method.

		SNAR		Dual SNAR		Two-Phase Method		Arsham's Method	
m	n	iterations	SD_1	iterations	SD_2	iterations	SD_3	iterations	SD_4
5	5	0.0015	0.0005	0.0009	0.0003	**0.0005**	*0.0001*	0.0014	0.0002
10	5	0.0016	0.0004	0.0008	0.0004	**0.0007**	*0.0002*	0.0027	0.0003
25	5	0.0040	0.0005	**0.0008**	0.0004	0.0013	*0.0003*	0.0121	0.0006
50	5	0.0133	0.0006	**0.0009**	*0.0004*	0.0023	*0.0004*	0.0534	0.0009
100	5	0.0723	0.0020	**0.0010**	*0.0006*	0.0057	0.0013	0.3049	0.0030
150	5	0.2319	0.0040	**0.0011**	*0.0006*	0.0093	0.0016	1.0342	0.0106
200	5	0.4933	0.0098	**0.0013**	*0.0006*	0.0152	0.0024	2.3548	0.0173
250	5	0.9968	0.0812	**0.0014**	*0.0006*	0.0234	0.0040	3.8536	0.1502
10	10	0.0047	0.0017	0.0025	0.0007	**0.0009**	*0.0004*	0.0043	0.0008
20	10	0.0049	0.0013	0.0022	0.0009	**0.0018**	*0.0005*	0.0094	0.0009
50	10	0.0172	0.0018	**0.0027**	0.0013	0.0043	*0.0012*	0.0507	0.0020
100	10	0.0810	0.0043	**0.0027**	*0.0014*	0.0097	0.0016	0.2758	0.0059
200	10	0.5653	0.0156	**0.0038**	*0.0020*	0.0293	0.0042	2.2965	0.0288
300	10	1.8107	0.0759	**0.0046**	*0.0017*	0.0694	0.0088	6.7843	0.2559
400	10	5.0841	0.1028	**0.0054**	*0.0018*	0.1317	0.0193	14.8202	0.5061
500	10	9.9067	0.2008	**0.0063**	*0.0022*	0.1947	0.0323	27.4845	0.6710
20	20	0.0259	0.0116	0.0089	0.0021	**0.0026**	*0.0002*	0.0199	0.0034
40	20	0.0316	0.0064	0.0090	0.0032	**0.0071**	*0.0020*	0.0495	0.0052
100	20	0.1513	0.0205	**0.0102**	0.0040	0.0214	*0.0026*	0.3529	0.0207
200	20	0.7621	0.0740	**0.0148**	*0.0061*	0.0672	0.0075	2.4944	0.1102
400	20	5.6864	0.1816	**0.0200**	*0.0077*	0.2816	0.0435	15.1222	0.4899
600	20	18.6959	0.5454	**0.0249**	*0.0056*	0.7159	0.1047	46.7817	1.3912
800	20	44.4151	0.9826	**0.0315**	*0.0104*	1.1433	0.1489	109.2029	2.2615
1000	20	92.0105	1.3487	**0.0391**	*0.0106*	1.8178	0.2583	215.7660	3.7612

In Table 4.9, the boldface numbers identify the smallest average number of iterations and the italic numbers identify the smallest standard deviations while, Table 4.10, the boldface numbers identify the smallest average running time and the italic numbers identify the smallest standard deviations of the running time for solving linear programming problems of the same size.

According to Table 4.10, since the average running time solved by SNAR and Arsham's method had very distinct time from the average running time solved by Dual SNAR and Two-Phase method, we will report only the average running time solved by Dual SNAR and Two-Phase method. Then, results in Tables 4.9 and 4.10 are plotted as follows.

Table 4.11: Ratios of the average number of iterations and the average running time solved by SNAR to Dual SNAR, by Two-Phase method to Dual SNAR and by Arsham's method to Dual SNAR

		Ratio of iterations			Ratio of the running time		
m	n	SN_P/SN_D	$2P/SN_D$	AM/SN_D	SN_P/SN_D	$2P/SN_D$	AM/SN_D
5	5	1.1759	0.6521	1.0013	1.5945	0.5396	1.4868
10	5	1.2907	1.0485	1.1850	2.0172	0.8892	3.3128
25	5	1.5866	1.4134	1.2067	5.1163	1.6667	15.5736
50	5	1.6638	1.5209	1.3240	15.5093	2.6682	62.3692
100	5	1.7800	1.6279	1.3488	72.3050	5.6850	304.8650
150	5	1.9011	1.5788	1.4121	211.7763	8.5342	944.5068
200	5	1.9717	1.6849	1.4264	391.5159	12.0913	1868.9206
250	5	1.8370	1.6245	1.4029	701.9718	16.4507	2713.8028
10	10	1.2561	0.4865	1.1217	1.8827	0.3678	1.7137
20	10	1.3181	0.8961	1.3152	2.2140	0.8333	4.2140
50	10	1.4937	1.2020	1.2726	6.4906	1.6113	19.1434
100	10	1.8611	1.4188	1.5655	29.7721	3.5699	101.3860
200	10	2.1406	1.5258	1.5456	149.5608	7.7566	607.5397
300	10	2.2288	1.6531	2.0349	396.6484	15.2026	1486.1555
400	10	2.1197	1.6619	1.7230	946.7598	24.5279	2759.8138
500	10	2.3261	1.7330	1.9661	1583.8050	31.1327	4394.0048
20	20	1.7313	0.3945	1.3537	2.9207	0.2961	2.2498
40	20	1.5492	0.8177	1.7987	3.5246	0.7924	5.5246
100	20	2.0447	1.1821	1.8782	14.9064	2.1064	34.7685
200	20	2.6271	1.3995	2.4361	51.3544	4.5276	168.0863
400	20	2.9953	1.6126	3.1829	284.6046	14.0961	756.8669
600	20	3.3046	1.7282	3.4993	750.2368	28.7287	1877.2753
800	20	3.5818	1.8221	3.4050	1410.0032	36.2952	3466.7587
1000	20	3.5852	1.8320	3.7196	2353.2097	46.4910	5518.3120

Names of columns in this table are described in the following table:

Table 4.12: Description of columns in table 4.11

SN_P	The SNAR algorithm
SN_D	The Dual SNAR algorithm
2P	The Two-Phase Method
AM	The Arsham's method

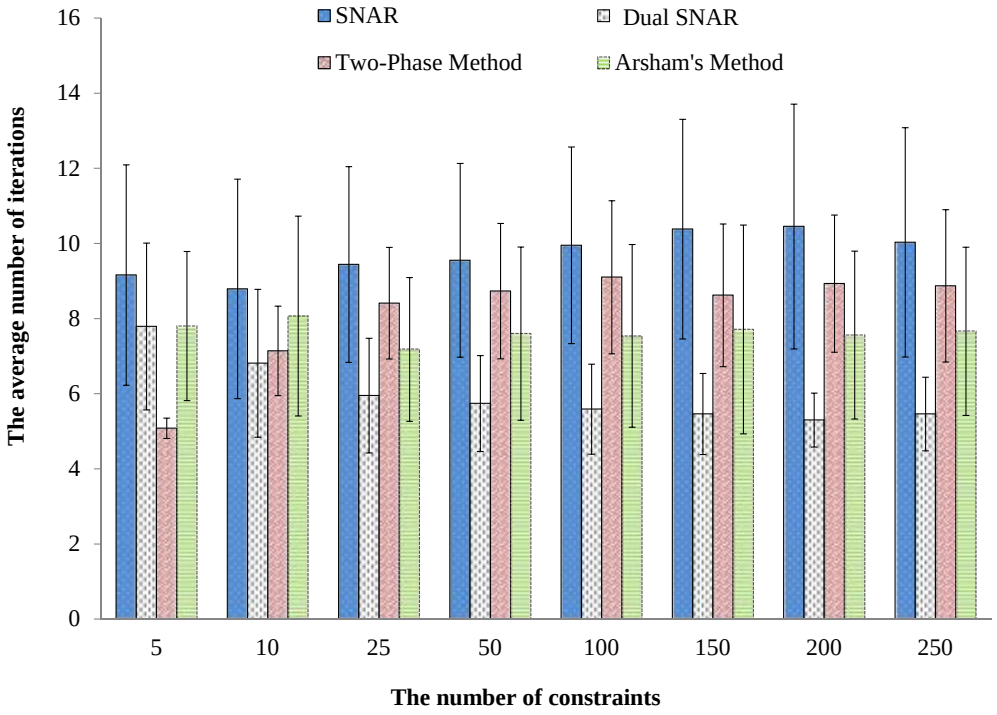


Figure 4.21: The average number of iterations solved by SNAR, Dual SNAR, Two-Phase method and Arsham's method for 5 variables

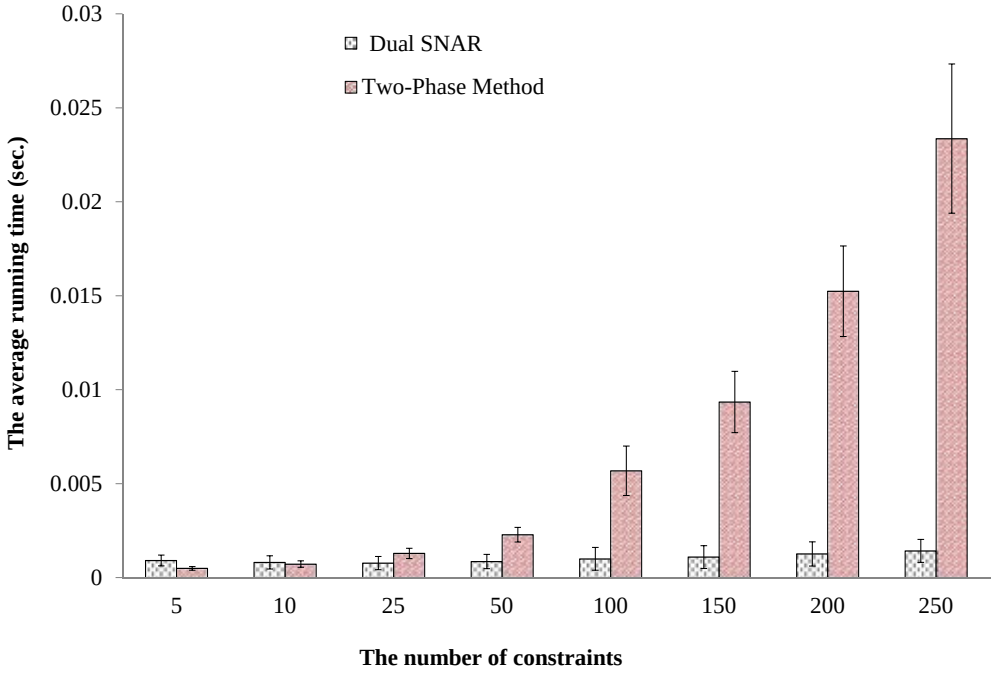


Figure 4.22: The average running time solved by Dual SNAR and Two-Phase method for 5 variables

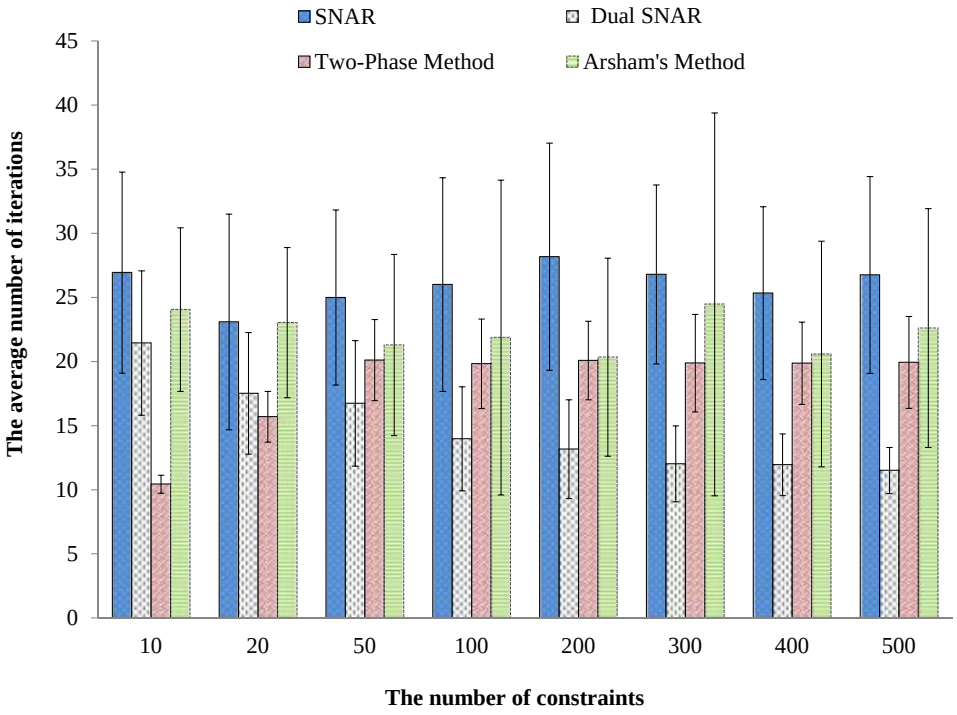


Figure 4.23: The average number of iterations solved by SNAR, Dual SNAR, Two-Phase method and Arsham's method for 10 variables

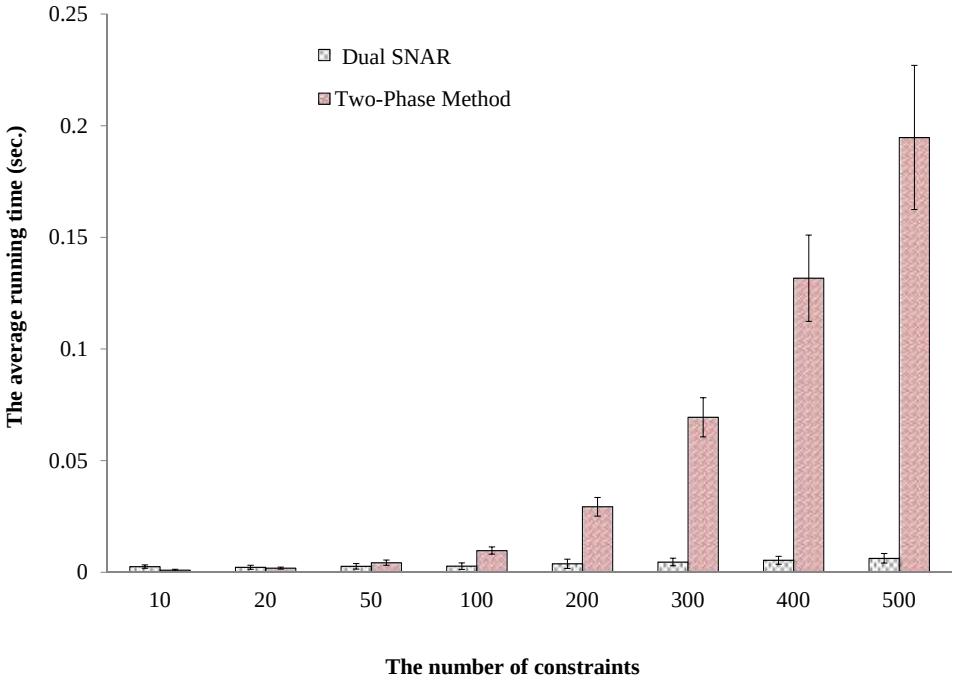


Figure 4.24: The average running time solved by Dual SNAR and Two-Phase method for 10 variables

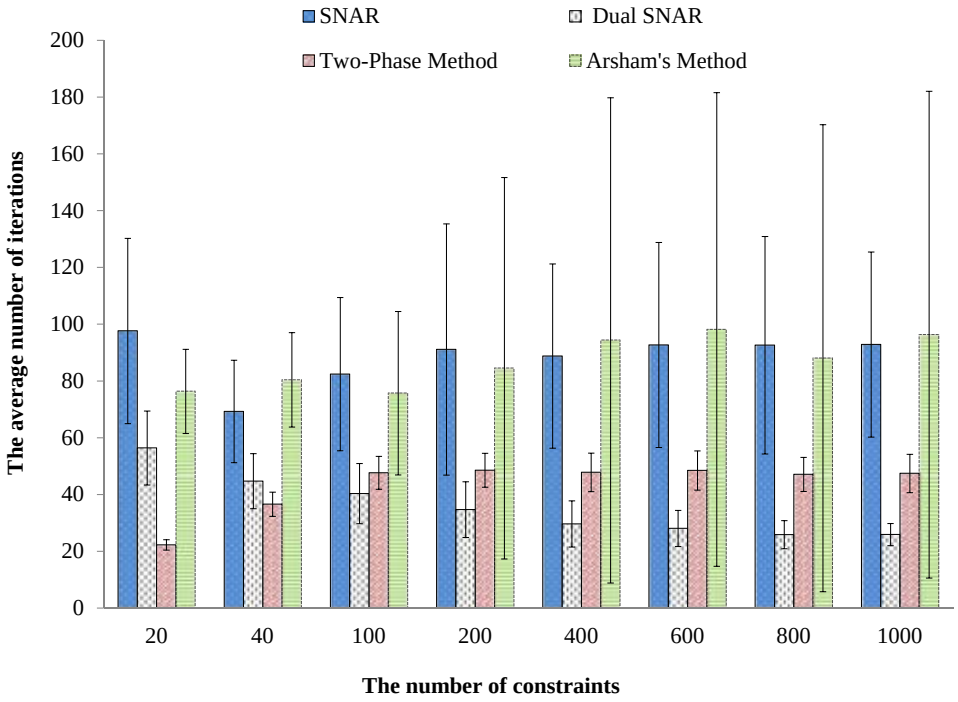


Figure 4.25: The average number of iterations solved by SNAR, Dual SNAR, Two-Phase method and Arsham's method for 20 variables

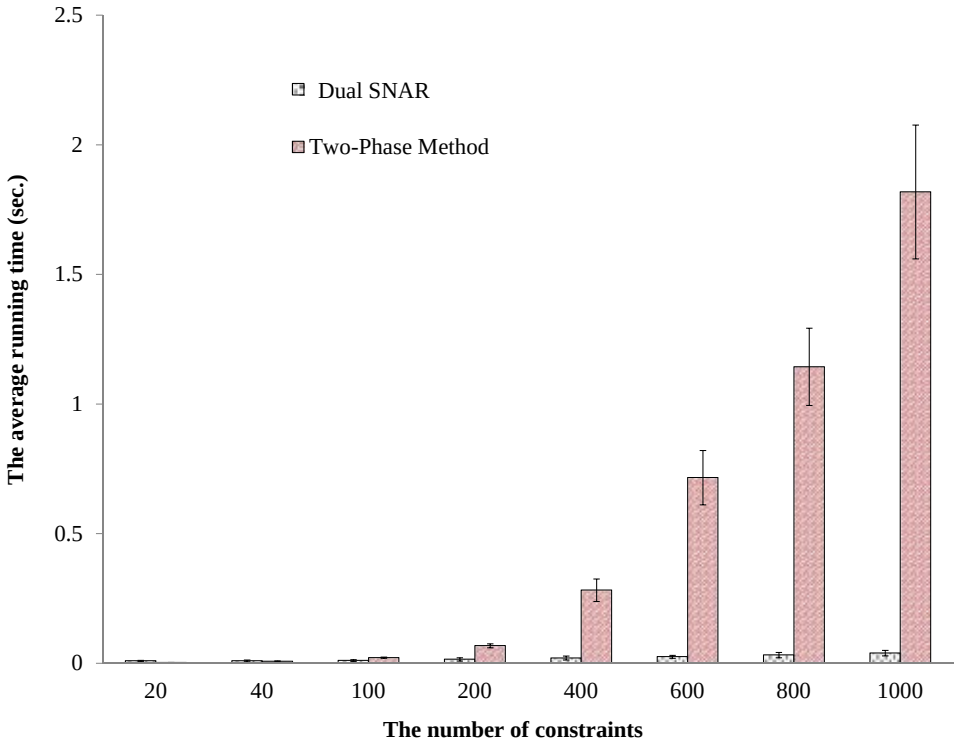


Figure 4.26: The average running time solved by Dual SNAR and Two-Phase method for 20 variables

4.3 Summary of Results

4.3.1 Problem P

For a small number of constraints, in Table 4.1, the majority of average number of iterations of SNAR are less than the average number of iterations of Two-Phase method. When the number of variables increases, the average number of iterations of SNAR is larger than the average number of iterations of Two-Phase method while the average number of iterations of Arsham's method are larger than the average number of iterations of SNAR for all sizes. Similarly, the majority of standard deviations of SNAR are less than the standard deviations of Two-Phase method while the standard deviations of Arsham's method are larger than the standard deviations of SNAR for all sizes.

Consider the average number of running time, in Table 4.3, there is only one size that the average running time solved by SNAR is less than the average running time solved by Two-Phase method, that is, 5 variables and 5 constraints. However, the average running time solved by SNAR are less than the average running time solved by Arsham's method for all sizes.

For a large number of constraints, in tables 4.5 and 4.6, the average number of iterations and the standard deviations of SNAR are less than the average number of iterations of Two-Phase method and Arsham's method for all sizes. Moreover, in Table 4.7, the average running time of SNAR are less than both of Two-Phase method and Arsham's method.

From the table of ratios, in Table 4.7, all ratios, both of the average number of iterations and the average running time, by Two-Phase method to SNAR and by Arsham's method to SNAR are greater than one. The greatest ratio of the average number of iterations by Two-Phase method to SNAR is 7.27 solving 5 variables and 200 constraints while the smallest ratio is 2.06 with 20 variables and 100 constraints. Additionally, the greatest ratio of the average number of iterations by Arsham's method to SNAR is 11.01 solving 100 variables and 500 constraints size while the smallest ratio is 1.66 with 5 variables and 100 constraints size.

For ratios of the average running time, the greatest ratio of it by the Two-Phase method to SNAR is 6.67 solving 100 variables and 4000 constraints size while the smallest ratio is 2.36 with 5 variables and 25 constraints size. Additionally, the greatest ratio of the average running time by Arsham's method to SNAR is 8.57 solving 100 variables and 500 constraints size while the smallest ratio is 2.75 with 5 variables and 25 constraints size.

4.3.2 Problem D

From figures 4.21, 4.23 and 4.25, the average number of iterations solved by SNAR are greater than the average number of iterations solved by Dual SNAR and Two-Phase method for all sizes while almost average number of iterations solved by SNAR are greater than the average number of iterations solved by Arsham's method except four sizes: 20 variables with 40, 400, 600 and 1000 constraints. While the average number of iterations solved by Dual SNAR are less than the average number of iterations solved by Arsham's method for all sizes, the majority of the average number of iterations solved by Dual SNAR are less than the average number of iterations solved by Two-Phase method except a small sizes as 5 variables with 5 constraints, 10 variables with 10 and 20 constraints and 20 variables with 20 and 40 constraints.

For comparing the average running time as shown in Table 4.10, solved by SNAR and Arsham's method much larger average running time than Dual SNAR and Two-Phase method. So we will consider only the average running time solved by Dual SNAR and Two-Phase method as Figures 4.21, 4.23 and 4.25 reported only the average running time solved by Dual SNAR and Two-Phase method. We found that, only small sizes as 5 variables with 5 and 10 constraints, 10 variables with 10 and 20 constraints and 20 variables with 20 and 40 constraints, the average running time solved by Dual SNAR are greater than the average running time solved by Two-Phase method.

From the table of ratios, in Table 4.11, all ratios, both of the average number of iterations and the average running time, by SNAR to Dual SNAR and by

Arsham's method to Dual SNAR are greater than one. The greatest ratio of the average number of iterations by SNAR to Dual SNAR is 3.5852 solving 20 variables and 1000 constraints size while the smallest ratio is 1.1759 with 5 variables and 5 constraints size. While the greatest ratio of the average number of iterations by Arsham's method to Dual SNAR is 3.7196 solving 20 variables and 1000 constraints size , and the smallest ratio is 1.0013 with 5 variables and 5 constraints size. Additionally, the greatest ratio of the average number of iterations by Two-Phase method to Dual SNAR is 1.8320 solving 20 variables and 1000 constraints size while the smallest ratio is 0.3945 with a small size as 20 variables with 20 constraints.

For the average running time, the greatest ratio of the average running time by SNAR to Dual SNAR is very high to 2353.2097 solving 20 variables and 1000 constraints size while the smallest ratio is 1.5945 with 5 variables and 5 constraints size. While the greatest ratio of the average running time by Arsham's method to Dual SNAR is very high to 5518.3120 solving 20 variables and 1000 constraints size , and the smallest ratio is 1.4868 with 5 variables and 5 constraints size. Additionally, the greatest ratio of the average running time by Two-Phase method to Dual SNAR is 46.4910 solving 20 variables and 1000 constraints size while the smallest ratio is 0.2961 with a small size as 20 variables with 20 constraints.

4.4 Discussion

4.4.1 Problems P

From computational results, we found that SNAR outperforms Arsham's method for all problem sizes since SNAR initially solves only acute constraints which has more chance to find the optimal solution than Arsham's method. Moreover, we found that a large proportion of the number of iterations of SNAR spent on solving the relaxed problem. This implies that most constraints which form the optimal solution are included in the relaxed problem while the number of iterations in the relaxed problem of the Arsham's method is small with respect to the whole

process. In addition, all standard deviations of SNAR are lower than the standard deviations of Arsham's method. This means that the total iterations of SNAR solving the linear programming problems of that size are approximately the same while this could not be concluded for Arsham's method.

For comparing between Two-Phase method and SNAR, we found that both of the average number of iterations and the average running time solved by SNAR outperforms Two-Phase method for the large problem sizes which the relaxed problems have optimal solutions. Additionally, the smallest ratio of the average number of iterations by Two-Phase method to SNAR is 2.06, the minimum iterations that SNAR can reduce to 2.06 of iterations solved by Two-Phase method. Moreover, the maximum iterations that SNAR can reduce to 7.27 of iterations solved by Two-Phase method. Not only it can reduce iterations but also the running time can be reduced. The minimum running time that SNAR can reduce to 2.36 of the running time solved by Two-Phase method. Moreover, the maximum running time that SNAR can reduce to 6.67 of the running time solved by the Two-Phase method.

From the Table 4.6, the average number of iterations for each size solved by SNAR is in the relaxed NAR problem. This implies that most constraints which form the optimal solution are included in the relaxed problem. Moving only a few steps, the optimal solution is found. While Two-Phase method wasted most of time to find a feasible solution in Phase I, SNAR can generate a feasible solution for our relaxed problem and solve the smaller problem size among constraints which form the optimal solution. Therefore, SNAR can reduce the iterations and the running time to solve a linear programming problem.

For a small number of constraints, although the majority of average number of iterations of SNAR are less than the average number of iterations of Two-Phase method, most of average running time of SNAR are greater than the average running time of the Two-Phase method. SNAR wastes the time computing dot product values while Two-Phase method can start immediately. So SNAR uses more time than Two-Phase method which the average number of iterations is not

different for small problem sizes.

For the problem size of 100 variables 200 constraints, the ratio of average number of iterations is 0.4 which is the smallest ratio of average running time for small problem sizes. Although Two-Phase method needs to introduce artificial variables to solve the linear programming problem, Two-Phase method outperforms SNAR for the small problem size of constraints with respect to the number of variables since SNAR spent more iterations after NAR is done. Two-Phase method deals with all constraints at the same time while SNAR after NAR is completed with unbounded optimal solution includes one constraint at a time. When it adds a constraint from the collection of non-acute constraints, it will be used the dual simplex to perturb the point from the relaxed problem for satisfying additional constraints. So it takes more iterations than Two-Phase method. Consider the following figure,

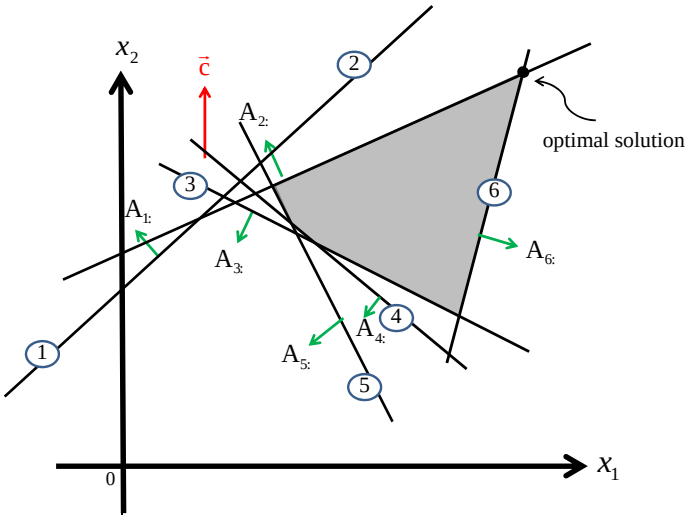


Figure 4.27: The original problem has the optimal solution.

In Figures 4.27, $P = \{1, 2\}$ and $N = \{3, 4, 5, 6\}$. Then, NAR is drawn as the following figure.

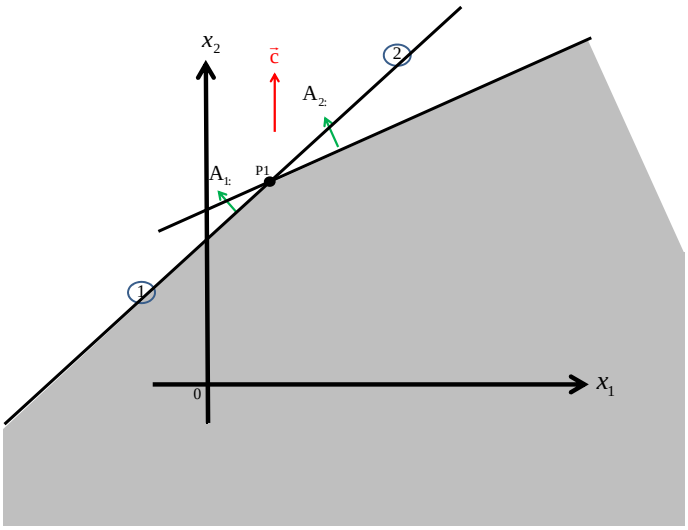
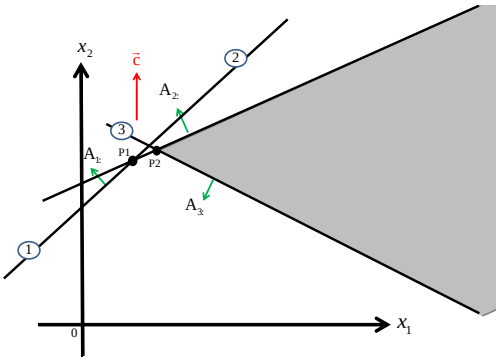
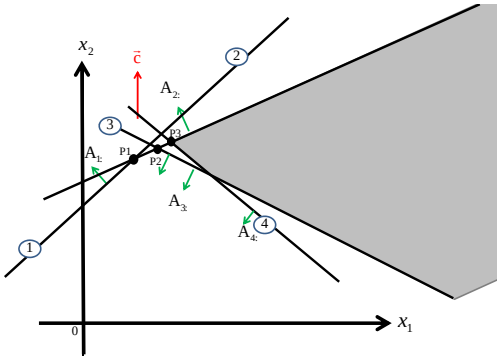


Figure 4.28: NAR is unbounded.

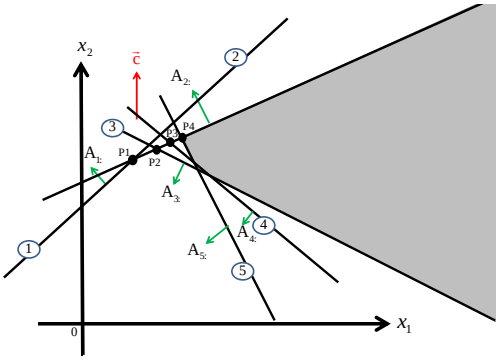
We found that NAR is unbounded and the current solution is at the point P1. Then, non-acute constraints from N will be added one by one which SNAR selects by ascending order of the index. The next point is drawn as follows:



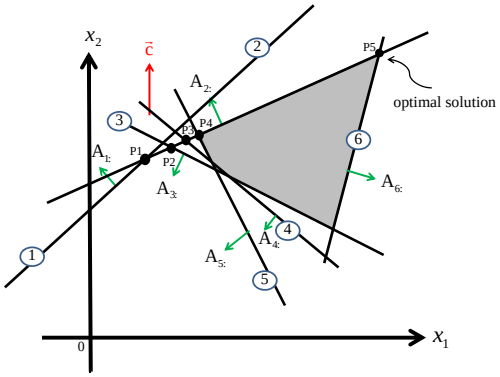
(a) Third constraint is added.



(b) Fourth constraint is added.



(c) Fifth constraint is added.



(d) Sixth constraint is added.

Figure 4.29: Example of NAR is unbounded and non-acute constraits are added.

From Figure 4.29a, the third constraints is added which causes the current solution P1 being primal and dual infeasible. So we will perturb original costs of objective function for dual feasibility and the dual simplex is performed to move the point from P1 for satisfying this additional constraint. Then, the solution moves to P2 and some perturbed costs are restored, but the relaxed problem is still unbounded. Then, the next non-acute constraint is added and the algorithm repeats by perturbing and using the dual simplex to move from P2 to P3 and P3 to P4 as shown in Figures 4.29b and 4.29c. Until the last constraint is added, the optimal solution is found at P5.

If we know that the sixth constraint will be used to form the optimal solution, and add this constraint first then the optimal solution is found immediately. We can only check feasibility with another non-acute constraints then it is done. So

the order of reinsertion constraints is important.

4.4.2 Problem D

From computational results, we found that SNAR and Arsham's method used much more iterations and time to solve this problem than Dual SNAR and Two-Phase method. Since the number of constraints of the standard double, the dimension of parameters solved by SNAR and Arsham's method is very large. It is different from Dual SNAR and Two-Phase method which do not increase the number of constraints.

Consider Dual SNAR and Two-Phase method, the majority of the average number of iterations and the average running time solved by Dual SNAR are less than the average number of iterations solved by Two-Phase method except the small problem sizes of constraints with respect to the number of variables. The reason is similar to SNAR which wastes time computing dot product values while Two-Phase method can start immediately. For the large problem size of constraints with respect to the number of variables, computing dot product values for classification constraints takes less time with respect to the total process. Dual SNAR can generate a feasible solution for the relaxed problem and solve the smaller problem size among constraints which form the optimal solution while Two-Phase method wastes time in Phase I to find the feasible point. The increasing number of constraints causes much more time to solve the problem by Two-Phase method. Moreover, the standard deviations of iterations solved by Dual SNAR are very low. This implied that the total iterations of Dual SNAR solving linear programming problems of that size are approximately the same.

CHAPTER V
CONCLUSIONS

In this dissertation, we proposed the artificial-variable-free technique to improve the simplex algorithm by relaxing the non-acute constraints. The relaxed problem will be transformed for starting the simplex algorithm from the origin point without using artificial variables. The collection of non-acute constraints are added to determine the solution of the original linear programming problem.

From the computational results, SNAR outperforms Arsham's method for all sizes and all problem structures and Two-Phase method except the small size of constraints which the relaxed problem has the unbounded optimal value. However, SNAR is not efficient for the linear programming problem in the standard form then we can use Dual SNAR instead the SNAR.

Additionally, if P and N_e are empty then we can conclude that the original problem is unbounded.

However, for the small size problem, SNAR is not efficient with respect to Two-Phase method since SNAR wasted the time in the non-acute constraint reinsertion. If the non-acute constraint which forms the optimal solution is inserted early, the optimal solution will be found rapidly.

In future work, we would like to choose the order of inserting. If the non-acute constraint which forms the optimal solution is inserted early, the optimal solution will be found quickly.

REFERENCES

[1] G.B. Dantzig.: Linear programming and extensions, Princeton Univ. Press, Princeton, New Jersey, 1963.

[2] V. Klee, G. J. Minty, How good is the simplex algorithm? in inequalities, New York: Academic Press. (1972) 159-175.

[3] N. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica. 4(1984) 373–395.

[4] R. Marsten, R. Subramanian, M. Saltzman, I. Lustig,D. Shanno, Interior point methods for linear programming: Just call Newton, Lagrange, and Fiacco and McCormick!, Interfaces. 20(1990) 105–116.

[5] H.A. Eiselt, C.L. Sandblom, Linear Programming and its Applications(online service), Berlin, Heidelberg : Springer Berlin Heidelberg, 2007.

[6] D. Goldfarb, K. Scheinberg, A product-form Cholesky factorization method for handling dense columns in interior point methods for linear programming, Math. Prog. 99(2004), 1-–34.

[7] C. M*észá*ros, Detecting "dense" columns in interior point methods for linear programs, Comput. Optim. Appl. 36(2007) 309–320.

[8] L. Blum, F. Cucker, M. Shub, S. Smale, Complexity and Real Computation, Springer-Verlag (1997).

[9] S. Smale, Mathematical Problems for the Next Century, Math. Intell. 20(1998) 7–15.

[10] S. Zionts, The criss-cross method for solving linear programming problems, Manage. Sci. 15(1969) 426-445.

[11] H. Arsham, An artificial-free simplex-type algorithm for general LP models, Math. Comput. Modelling. 25(1997) 107-123.

[12] H. Arsham, Initialization of the simplex algorithm: an artificial-free approach, SIAM Rev. 39(1997) 736-744.

[13] A. Enge, P. Huhn, A counterexample to H. Arsham's "Initialization of the simplex algorithm: an artificial-free approach", SIAM Rev. 40(1998) online.

[14] P.Q. Pan, Primal perturbation simplex algorithms for linear programming, J. Comput. Math. 18(2000) 587-596.

[15] H. Arsham, Big-M free solution algorithm for general linear programs, IJPAM. 32(2006) 37-52.

[16] H. Arsham, A computationally stable solution algorithm for linear programs, Appl. Math. Comput. 188(2007) 1549-1561.

[17] H.W. Corley, J. Rosenberger, W.C. Yeh, T.K. Sung, The cosine simplex algorithm, IJAMT. 27(2006) 1047-1050.

[18] P.Q. Pan, Practical finite pivoting rules for the simplex method, OR Spektrum. 12(1990) 219-225.

[19] N.V. Stojkovic, P.S. Stanimirovic, Two direct methods in linear programming, Eur. J. Oper. Res. 131(2001) 417-439.

[20] W. Li, A note on "two direct methods in linear programming", Eur. J. Oper. Res. 158(2004) 262-265.

[21] H.V. Junior, M.P.E. Lins, An improved initial basis for the simplex algorithm, Comput. Oper. Res. 32(2005) 1983-1993.

[22] J.F. Hu, A note on "an improved initial basis for the simplex algorithm", Comput. Oper. Res. 34(2007) 3397-3401.

[23] W.C. Yeh, H.W. Corley, A simple direct cosine simplex algorithm, Appl. Math. Comput. 214(2009) 178-186.

[24] W. Li, H. Li, On simplex method with most-obtuse-angle rule and cosine rule, Appl. Math. Comput. 217(2011) 7867-7873.

[25] M. S. Bazaraa , J. J. Jarvis, H. D. Sherali, Linear programming and network flows, New York : John Wiley & Sons, 1990.

[26] E. H. Moore, On the reciprocal of the general algebraic matrix, Bull. Amer. Math. Soc. 26(1920) 394–395.

[27] R. Penrose, A generalized inverse for matrices, Proc. Camb. Philos. Soc. 51(1955) 406–413.

VITA

Name	Miss Aua-aree Boonperm
Date of Birth	22 February 1981
Place of Birth	Surin, Thailand
Education	B.Sc.(Mathematics)(Second Class Honors), Khon Kaen University, 2003 M.Sc.(Computational Science), Chulalongkorn University, 2009
Scholarship	Development and Promotion of Science and Technology talents project (DPST)
Publications	• A.-A. Boonperm and K. Sinapiromsaran: Linear time Algorithm in term of Number of the Constraints for Linear Programming in 2D, *Proceeding of OR-NET 2010*, pp. 48-53. • A.-A. Boonperm and K. Sinapiromsaran: The New Origin Point for Starting Simplex Algorithm, *Proceeding of OR-NET 2012*, pp. 148-152. • A.-A. Boonperm and K. Sinapiromsaran: The Artificial-free Technique along the Objective Direction for the Simplex Algorithm, On-line in *Journal of Physics: Conference Series* 490(2014) 012193. • A.-A. Boonperm and K. Sinapiromsaran: Artificial-Free Simplex Algorithm Based on the Non-acute Constraint Relaxation, *Applied Mathematics and Computation*, Vol. 234 (2014), pp. 385-401.