การกระโดดวนซ้ำไปยังจุดยึดสำหรับวิธีซิมเพล็กซ์

นางสาวรุจิรา วิสุทธิรัตนมณี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562

ITERATIVE JUMP TO BINDING POINT FOR SIMPLEX METHOD

Miss Rujira Visuthirattanamanee

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 2019

Dissertation Title	ITERATIVE JUMP TO BINDING POINT FOR SIMPLEX METHOD
By	Miss Rujira Visuthirattanamanee
Field of Study	Applied Mathematics and Computational Science
Dissertation Advisor	Assistant Professor Krung Sinapiromsaran, Ph.D.
Dissertation Co-advisor	Assistant Professor Aua-aree Boonperm, 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 Polkit Sangvanich, Ph.D.)

DISSERTATION COMMITTEE

..... Chairman
(Associate Professor Phantipa Thipwiwatpotjana, Ph.D)

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

..... Dissertation Co-advisor
(Assistant Professor Aua-aree Boonperm, Ph.D.)

..... Examiner
(Assistant Professor Boonyarit Intiyot, Ph.D)

..... Examiner
(Assistant Professor Kitiporn Plaimas, Ph.D)

..... External Examiner
(Assistant Professor Wutiphol Sintunavarat, Ph.D.)

รุจิรา วิสุทธิรัตนมณี : การกระโดดวนซ้ำไปยังจุดยึดสำหรับวิธีซิมเพล็กซ์. (ITERATIVE JUMP TO BINDING POINT FOR SIMPLEX METHOD) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. กรุง สินอภิรมย์สราญ, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม : ผศ.ดร. เอื้ออารี บุญเพิ่ม, 176 หน้า.

แนวคิดพื้นฐานของวิธีกระโดดแบบวนซ้ำคือการเคลื่อนที่จากจุดที่เป็นไปได้ไปตามทิศทางที่ปรับปรุงค่าวัตถุประสงค์ที่คงความเป็นไปได้ วิธีดังกล่าวถูกนำมาประยุกต์ใช้กับการหาผลเฉลยของปัญหากำหนดการเชิงเส้นไร้ตัวแปรเทียมโดยการประยุกต์วิธีกระโดดแบบวนซ้ำกับปัญหากำหนดการเชิงเส้นแบบผ่อนคลายที่ประกอบไปด้วยเงื่อนไขมุมแหลมเทียบกับทิศทางของวัตถุประสงค์และแทรกเงื่อนไขมุมไม่แหลมเข้าใหม่เพื่อหาผลเฉลยที่เหมาะที่สุด โดยตั้งชื่อว่าเอสเอเจเอส อย่างไรก็ตามวิธีดังกล่าวอาจทำให้จุดกระโดดสุดท้ายไกลจากจุดที่เหมาะที่สุด ดังนั้นวิธีอื่นสำหรับการหาจุดเริ่มต้นที่เหมาะสมจึงถูกนำเสนอ วิธีใหม่ที่ถูกนำเสนอ เอเจเอสพี ได้ใช้เทคนิคนี้กับการรบกวนค่าด้านขวาของเงื่อนไขบังคับที่ไม่สอดคล้องเพื่อให้วิธีนั้นสามารถเริ่มต้นด้วยจุดที่เป็นไปได้ก่อนการประยุกต์ใช้วิธีกระโดดแบบวนซ้ำ ทั้งเอสเอเจเอสและเอเจเอสพีมีประสิทธิภาพเหนือกว่าวิธีซิมเพล็กซ์มาตรฐานและขั้นตอนวิธีซิมเพล็กซ์ปราศจากตัวแปรเทียมขึ้นอยู่กับการผ่อนคลายเงื่อนไขที่มุมไม่แหลมบนปัญหากำหนดการเชิงเส้นที่ถูกสร้างขึ้นและปัญหาจากเน็ตลิบ

ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต .....
		ลายมือชื่อ อ.ที่ปรึกษาหลัก .....
สาขาวิชา	คณิตศาสตร์ประยุกต์และวิทยาการคณนา	ลายมือชื่อ อ.ที่ปรึกษาร่วม .....
ปีการศึกษา	2562	

5772884323 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE
KEYWORDS : ARTIFICIAL-FREE LINEAR PROGRAMMING METHOD / NON-ACUTE CONSTRAINT RELAXATION / SIMPLEX METHOD / JUMP TECHNIQUE

RUJIRA VISUTHIRATTANAMANEE : ITERATIVE JUMP TO BINDING POINT FOR SIMPLEX METHOD. ADVISOR : ASST. PROF. KRUNG SINAPIROMSARAN, Ph.D., CO-ADVISOR : ASST. PROF. AUA-AREE BOONPERM, Ph.D., 176 pp.

The basic idea of an iterative jump method is moving a feasible point along the direction that improves the objective value maintaining the feasibility. It is applied for solving a linear programming (LP) model without artificial variables by applying the iterative jump on the LP relaxation having only acute constraints with respect to the objective direction and reinsert all non-acute constraints to find the optimal solution which is named SAJS. However, it may cause the last jump point to locate far away from the optimal solution so another approach for initially finding a suitable starting point is proposed. The new proposed method, AJSP use this technique together with the perturbation of the right-hand side values of violated constraints to be able to start at the feasible point before applying the iterative jump method. Both SAJS and AJSP outperform the standard simplex method and the artificial-free simplex algorithm based on the non-acute constraint relaxation on synthetic linear programming problems and Netlib problems.

Department	: Mathematics and Computer Science	Student's Signature	.....
		Advisor's Signature	.....
Field of Study	: Applied Mathematics and Computational Science	Co-advisor's Signature	.....
Academic Year	: 2019		

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Assistant Professor Dr. Krung Sinarpiromsaran, and my co-advisor, Assistant Professor Dr. Aua-aree Boonperm for their valuable guidance, support, motivation, and enormous cognizance throughout my doctorate degree. They are excellent consultants who invented me to achieve my goals.

Besides my advisor and my co-advisor, I would like to thank my thesis committees, Associate Professor Dr. Phantipa Thipwiwatpotjana, Assistant Professor Dr. Boonyarit Intiyot, Assistant Professor Dr. Kitiporn Plaimas, and my dissertation external examiner, Assistant Professor Dr. Wutiphol Sintunavarat, for their knowledgeable comments and valuable suggestions.

In addition, I would like to thank my friends, Mr. Teeradech Laisupannawong, Mr. Ampol Duangpan, Miss Chittima Chiamanusorn, Mr. Panote Songwattanasiri, Mr. Phiraphat Sutthimat, Mr. Chaiyod Kamthorncharoen, and Mr. Senee Kitimoon for their help, support, advice, and encouragement throughout my graduate studies.

Moreover, I would like to gratefully thank the Science Achievement Scholarship of Thailand (SAST) for financial support throughout my Ph.D. study and the Applied Mathematics and Computational Science Program in the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University for the resource support in running my research.

Finally, I would like to thank my family and my friends for always supporting and encouraging me throughout the period of studying my Ph.D.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Literature reviews

The simplex method proposed by Dantzig [1] in 1947 is a popular method for solving the small-sized or medium-sized LP model. It starts at the feasible origin point and moves along the edge of the polytope to an adjacent extreme point until it reaches the new optimal one. However, if the origin point is infeasible, artificial variables are added into the LP model that will make the origin point feasible in the extended LP model. Therefore, the size of the extended LP model will increase due to the number of artificial variables and the number of constraints.

There are many researchers attempt to improve the method for solving the LP model using the simplex method without artificial variables [2, 3, 4, 5]. Some researchers require finding an initial point closer to the optimal point that will reduce the total running time [6, 7, 8, 9, 10, 5, 11, 12, 13, 14]. Besides, the relation of angles between the gradient vector of the objective function and the gradient vector of each constraint has an important role to improve the simplex method [6, 3, 15, 4, 11, 12].

In 2000, the method for solving an LP model without artificial variables was proposed by Pan [2]. Generally, an LP model might not be feasible, then the simplex method could not be applied. Therefore, the perturbation technique was applied to generate the feasible initial starting point before performing the standard simplex method or the dual simplex method [1]. After the solution of the perturbation LP model was found, the perturbed values were restored to the original value of the LP model in order to find the solution of the original LP model.

In 2005, Junior and Lins [6] presented the algorithm for finding the initial basis near the optimal point, claiming that the gradient vector of the first constraint that made the most acute angle with the gradient vector of the objective function should be one of

the constraints created basis near the optimal point. Their algorithm used the primal and dual LP problem relationships [1] to find the initial basis since the variables in the dual LP problem were related to the constraints in the primal LP problem. The basic variables in the dual LP problem were selected based on the angle between the gradient vector of constraints and the gradient vector of the objective function that gave the largest acute angle to the obtuse angle according to the number of the bases of the dual LP problem. Although this method could improve the number of iterations by 33% from the simplex method. However, their test problems were small. Consequently, Hu [7] proposed the counterexample of the algorithm of Junior and Lins in 2007. Since creating the basis of the dual LP problem using the Junior and Lins algorithm might make its matrix singular. Therefore, Hu proposed the process to improve the Junior and Lins algorithm using the LU decomposition. Furthermore, most Netlib problems applied with the Junior and Lins algorithm exhibit initial singular bases.

The algorithm for solving the LP model without using artificial variables and dealing with redundant constraints was proposed by Corley et al. [3] in 2006. Their algorithm began by creating the relaxation model with a single bounded constraint. After that, the next constraint was reinserted into the LP model one by one until the optimal solution was found. For each sequence of the relaxation model, the single constraint that did not satisfy to the previous relaxation model was added based on the angle between the gradient vector of the rest of constraints and the gradient vector of the objective function with the cosine rule. Moreover, in 2009, Yeh and Corley [15] proposed the algorithm for selecting the entering variables of the simplex method by considering the relation of angles in dual problems with the cosine rule.

In 2009, Nabli [8] introduced the new algorithm for finding the initial basic feasible without using the nonfeasible basis method (NFB). After the initial basic feasible was found, the revised simplex method was performed to find the solution. In addition, the concept of the formal tableau was applied to the NFB, called the formal nonfeasible basis method (FNFB). The formal tableau was the dual tableau that was created from the primal tableau. The two-phase simplex method and the big-M simplex method were used to compare with NFB. The result was shown that NFB used the number of iterations less than both methods. However, the number of iterations of both NFB and FNFB was slightly different. Afterward, in 2009, Stojkovic, Stanimirovic, and Petkovic [9] proposed

the algorithm for finding the initial basic feasible solution similar to the Nabli's algorithm by improving the phase-1 of the two-phase method. It created the new pivot rule to select entering and leaving variables, called M1. Moreover, they also proposed the M2 algorithm, which was based on the improvements to M1. Subsequently, in 2012, Stojkovic, Stanimirovic, and Petkovic [10] showed the comparison of the number of iterations and CPU time of M1, M2, and NFB. As a result from the Netlib problems, in the phase of finding the initial basic feasible solution, M1 used the least number of iterations. While M2 used the least number of total iterations including the phase of finding the initial basic feasible solution and the phase of finding the optimal solution. Therefore, the position of the initial basic feasible solution had affected the number of iterations for solving the LP model.

In 2014, Boonperm and Sinapiromsaran [4] proposed the algorithm for solving the LP problem without artificial variables. Their algorithm began with the relaxation model, it consisted of the constraints having acute angle between the gradient vector of constraints and the gradient vector of the objective function. After the solution of the relaxation model was found by performing the simplex method, the rest of the constraints were reinserted to the model one by one in order to find the solution of the original model. However, the solution of the relaxation model might be far from the optimal solution, which slowed the processing time significantly.

In 2015, Nabli and Chahdoura [5] proposed the algorithm for finding the initialized simplex algorithm without artificial variables. It was able to detect redundant constraints and verified the infeasibility of the LP problem. This algorithm proposed the new pivot rule based on the NFB and FNFB methods, called $\overline{\text{NFB}}$ and $\overline{\text{FNFB}}$, respectively, to increase the efficiency of finding the LP solutions.

In 2016, the idea of the jump was proposed by Yawila, Intiyot, and Sinapiromsaran [11]. They had been developed the method for solving an LP model using the jump technique. The concept of their method was an improvement of the initial extreme point for the simplex method. The first process of this method moved the origin point along the direction of the gradient vector of the objective function until it hit some constraints. However, if it was not an extreme point, then artificial constraints were added into the LP model to generate the extreme point of this extended LP model. After that the

simplex method would be performed to seek the extreme point of the original LP model. Nevertheless, their method only worked on the positive vector of the objective function together with the less than or equal to constraint and the positive right-hand side.

In 2019, Kafakthong and Sinapiromsaran [12] had been improved the Yawila's method without adding artificial constraints. Their algorithm started with the same initial jump point as the Yawila's method. Then the binding constraints at the next jump point were solved to find the direction toward the extreme point of the original problem by the least square method. The process was repeated until the extreme point of the LP model was obtained. However, if the origin point was infeasible, the two-phase simplex method was performed to find a feasible point. Additionally, if the direction of the gradient vector of the objective function points was away from the feasible region, this method could not be used.

For the large-sized LP model, in 1984, Karmarkar [16] proposed an effective method that was the classical interior-point method. The concept of an interior-point method began with finding an initial interior point. After that, the LP model would be re-scaled to find the suitable direction to improve the objective value of the initial point. After the direction was found, the initial point was moved by the step size that maintained the interiority. The process would iterate until the last interior point was near the optimal point, based on the duality gap. However, the transformation of the LP problem into the initial Karmarkar's form required excessive computational time. Thus, the gravitational method was developed for solving the LP model using the interior point concept in 1986 by Chang [17], avoiding the complexity of the Karmarkar's method. The initialization of the gravitational method was the creation of a small ball, covering the initial interior point as the center of the ball. Whereupon, it dropped along the gravitational force with the steepest descent direction until it hit the boundary of the feasible region. After that, it moved on the surfaces of the feasible region until it achieved the optimal point. Moreover, many variations have been developed including the affine scaling problem [18, 19]. However, the solution of the interior-point method was not exact.

From the efficiency of the interior-point method, researchers had applied the interior-point method together with the simplex method. In 2002, Luh and Tsaih [13] proposed the auxiliary algorithm for finding the good initial feasible solution before applying the

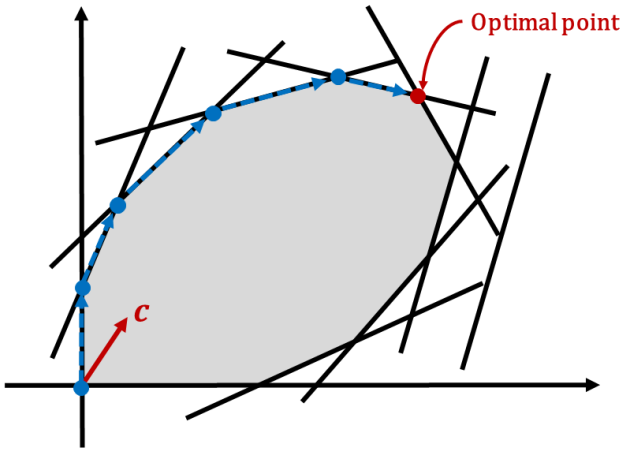


Figure 1.1: Motivation of the simplex method

simplex method. It started with a feasible point and moved it to the corner point that improved the objective value with the direction using the idea of the interior point method. As a result, it could reduce the number of iterations of the simplex method after the initial feasible point was found. However, this algorithm could be applied when it had an initial feasible point. So their algorithm suggested using the two-phase simplex method to find an initial feasible point. Moreover, the problems that were used to test their algorithm were small having the number of variables equals to the number of constraints.

In 2011, Al-Najjar and Malakooti [14] applied an idea of the interior-point method to propose the algorithm of finding an initial basic feasible solution. It started with a basic solution and moved to the improved initial basic feasible solution within the feasible region, which it could dodge some extreme points. After the initial basic feasible solution was found, the simplex method was performed to find the optimal solution. However, for the phase of finding an initial basic feasible solution, the parameter was set up to find the direction which affected the position of the derived initial basic feasible solution.

1.2 Motivation

For the simplex method, both the number of variables and the number of constraints affect computational time. If the origin point of the LP model is infeasible, the artificial variables are added to make it feasible. As a result, the size of the original LP model

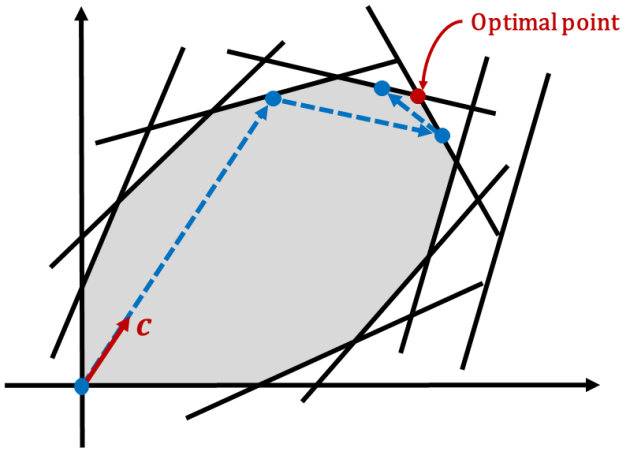


Figure 1.2: Motivation of the iterative jump method

is expanded since the number of artificial variables impact on the number of variables. Moreover, if it has a large number of constraints, it leads to a larger number of extreme points presented in Figure 1.1. Therefore, this dissertation proposes two new methods for solving the LP model without artificial variables using the jump technique called the self-regulating artificial-free linear programming solver using a jump and simplex method (SAJS) and the artificial-free linear programming using a jump and the simplex method by starting with perturbed constraints (AJSP). The iterative jump method is applied to both SAJS and AJSP to improve the current feasible point having better objective values, or in other words, to move the current feasible point near an optimal point. The iterative jump method attempts to seek a suitable initial extreme point of the simplex method by avoiding unnecessary visited extreme points shown in Figure 1.2.

1.3 Dissertation overview

This dissertation is organized into 6 chapters as follows: Chapter 1 (Introduction) explains the literature reviews related to this dissertation and the motivation of this dissertation. Chapter 2 (Background and knowledge) presents background and knowledge for this dissertation. It covers linear algebra, a linear programming model, a dual in linear programming, artificial-free variable methods, artificial variable methods, and an artificial-free simplex algorithm based on the non-acute constraint relaxation (SNAR). Chapter 3 (The new technique for solving the unrestricted variable model) proposes the

new method for solving the unrestricted variable model without introducing two newly non-negative variables for each unrestricted one. Chapter 4 (Self-regulating artificial-free linear programming solver using a jump and simplex method) presents the new method for solving the LP model without artificial variables by performing the iterative jump method based on the relaxation LP model. Chapter 5 (Artificial-free linear programming using a jump and the simplex method by starting with perturbed constraints) introduces the new method for solving the LP model without artificial variables by performing the iterative jump method based on the perturbation LP model. In the last chapter (Conclusion), the analyses of all methods are explained and concluded.

CHAPTER II

BACKGROUND AND KNOWLEDGE

In this chapter, the basic knowledge related to this dissertation is described. First, the notations used in this dissertation are explained. Next, basic linear algebra is described. Then, a linear programming model and a dual linear programming model are proposed, respectively. After that, some methods for solving the LP model are presented.

2.1 Notation

Let $\mathbf{A}$ be a matrix and $\mathbf{b}$ be a right-hand side vector from the LP model.

$\mathbf{A}_{i:}$	is the row vector of i^{th} constraint from $\mathbf{A}$.
$\mathbf{A}_{:i}$	is the column vector of i^{th} constraint from $\mathbf{A}$.
b_i	is the i^{th} element of $\mathbf{b}$.
$\|\cdot\|$	is the Euclidean norm.
$Index$	is the set of constraint indices.
$\mathbf{B}$	is the basis.
$\mathbf{N}$	is the non-basic matrix.
$\mathbf{I_B}$	is the index set of the basic variables.
$\mathbf{I_N}$	is the index set of the non-basic variables.
$\mathbf{I_R}$	is a set of restricted variables.
$\mathbf{I_U}$	is a set of unrestricted variables.
$\mathbf{I}_{Acute}$	is the set of all acute constraints.
$\mathbf{I}_{NonAcute}$	is the set of all non-acute constraints.
$\mathbf{A}_{\mathbf{I}_{Acute}}$	is the submatrix of $\mathbf{A}$ that row indices from $\mathbf{I}_{Acute}$.
$\mathbf{A}_{\mathbf{I}_{NonAcute}}$	is the submatrix of $\mathbf{A}$ that row indices from $\mathbf{I}_{NonAcute}$.

$\mathbf{b}_{\mathbf{I}_{Acute}}$	is the column vector of $\mathbf{b}$ that corresponding to $\mathbf{I}_{Acute}$.
$\mathbf{b}_{\mathbf{I}_{NonAcute}}$	is the column vector of $\mathbf{b}$ that corresponding to $\mathbf{I}_{NonAcute}$.
$\mathbf{x}^{(0)}$	is the initial feasible point.
$\mathbf{x}^{(k)}$	is the iterative feasible point for each iteration.
α	is the step length.
$\mathbf{d}^{(k)}$	is the direction for each iteration.
$\gamma^{(k)}$	is the index of constraint which $\mathbf{x}^{(k)}$ is binding.
ε	is the stopping criterion.

2.2 Basic linear algebra

Some basic linear algebra and notations related to this dissertation are introduced. Principally, elements of the vector and the matrix in this dissertation are real numbers. A column vector $\mathbf{v}$ is a vector having n elements, that is, $\mathbf{v} \in \mathbb{R}^n$. The v_i is the i^{th} element of $\mathbf{v}$. $\mathbf{A}$ is a matrix having m rows and n columns, that is, $\mathbf{A} \in \mathbb{R}^{m \times n}$. The a_{ij} is the i^{th} row and j^{th} column of $\mathbf{A}$. Moreover, $\mathbf{A}_{i:}$ represents a row vector of i^{th} constraint from $\mathbf{A}$ and $\mathbf{A}_{:i}$ represents the column vector of i^{th} constraint from $\mathbf{A}$.

Let $\mathbf{A}$ be $\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$ and $\mathbf{v}$ be $\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$.

Definition 2.1. The transpose matrix of $\mathbf{A}$ denoted by $\mathbf{A}^\top$ where

$$\mathbf{A}^\top = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}.$$

Definition 2.2. The zero vector denoted by $\mathbf{0}$ is the vector which all elements are zeros.

Definition 2.3. $\mathbf{e}_i$ is the vector which the i^{th} position is 1 while other positions are 0s.

Definition 2.4. Let $\mathbf{a} = [a_1, a_2, ..., a_n]^\top$ and $\mathbf{b} = [b_1, b_2, ..., b_n]^\top$ be column vectors of size n. The addition of two vectors is denoted as $\mathbf{a} + \mathbf{b}$ which is the following vector:

$$\mathbf{a} + \mathbf{b} = [a_1, a_2, ..., a_n]^\top + [b_1, b_2, ..., b_n]^\top = [a_1 + b_1, a_2 + b_2, ..., a_n + b_n]^\top.$$

Definition 2.5. Let $\mathbf{a} = [a_1, a_2, ..., a_n]^\top \in \mathbf{R}^n$ and the scalar $k \in \mathbf{R}$. The operation of multiplying scalar k with vector $\mathbf{a}$, it is denoted as $k\mathbf{a}$ which can be written as $k\mathbf{a} = k[a_1, a_2, ..., a_n]^\top = [ka_1, ka_2, ..., ka_n]^\top$.

Definition 2.6. Let $\mathbf{a} = [a_1, a_2, ..., a_n]^\top \in \mathbf{R}^n$ and $\mathbf{b} = [b_1, b_2, ..., b_n]^\top \in \mathbf{R}^n$. The inner product or dot product is determined by

$$\mathbf{a}^\top\mathbf{b} = [a_1, a_2, ..., a_n][b_1, b_2, ..., b_n]^\top = a_1b_1 + a_2b_2 + ... + a_nb_n.$$

Definition 2.7. Let $\mathbf{a} = [a_1, a_2, ..., a_n]^\top \in \mathbf{R}^n$. The Euclidean norm of $\mathbf{a}$ is denoted as $\|\mathbf{a}\|$ which can be written as $\|\mathbf{a}\| = \sqrt{\mathbf{a}^\top\mathbf{a}} = \sqrt{a_1^2 + a_2^2 + ... + a_n^2}$.

Definition 2.8. Let $\mathbf{a}$ and $\mathbf{b} \in \mathbf{R}^n$. The angle θ between two vectors can be computed by the inner product and the norm of two vectors as follows: $\theta = \arccos\left(\dfrac{\mathbf{a}^\top\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}\right)$.

Since $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are always positive. Therefore, the categories of the angle between two vectors are divided by the sign of the inner product of two vectors.

If $\mathbf{a}^\top\mathbf{b} > 0$, then the angle between $\mathbf{a}$ and $\mathbf{b}$ is an acute angle.

If $\mathbf{a}^\top\mathbf{b} = 0$, then the angle between $\mathbf{a}$ and $\mathbf{b}$ is a right angle.

If $\mathbf{a}^\top\mathbf{b} < 0$, then the angle between $\mathbf{a}$ and $\mathbf{b}$ is an obtuse angle.

Definition 2.9. $\mathbf{b} \in \mathbf{R}^n$ is a linear combination of $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_l \in \mathbb{R}^n$ if

$\mathbf{b} = \lambda_1\mathbf{a}_1 + \lambda_2\mathbf{a}_2 + ... + \lambda_l\mathbf{a}_l$, where $\lambda_1, \lambda_2, ..., \lambda_l \in \mathbb{R}$.

Definition 2.10. $\mathbf{X} \subseteq \mathbb{R}^n$ is called a convex set if $x_1, x_2 \in \mathbf{X}$ implies $\lambda x_1 + (1 - \lambda) x_2 \in \mathbf{X}, \forall\lambda \in [0, 1]$.

Definition 2.11. Let x be in a convex set $\mathbf{X}$. It is an extreme point of $\mathbf{X} \subseteq \mathbb{R}^n$ if and only if $x = \lambda x_1 + (1 - \lambda) x_2$ for $\lambda \in (0, 1)$ and $x_1, x_2 \in \mathbf{X}$ implies $x = x_1 = x_2$.

Definition 2.12. A collection of $\mathbf{A}_{:1}, \mathbf{A}_{:2}, \mathbf{A}_{:3}, ..., \mathbf{A}_{:k}$ of dimension n is called linearly independent if $\lambda_1\mathbf{A}_{:1} + \lambda_2\mathbf{A}_{:2} + \lambda_3\mathbf{A}_{:3} + ... + \lambda_k\mathbf{A}_{:k} = 0$ implies that $\lambda_j = 0$ for $j = 1, 2, 3, ..., k$.

2.3 Linear programming

A linear programming (LP) model is a mathematical model formulated from a real-world problem to find the best solution among other feasible solutions such as airline problems, transportation problems, medical problems, agricultural problems [1]. Three mathematical components of the LP model consist of decision variables, an objective function, and the constraints. Normally, the objective function is linear which can be maximized or minimized and the constraints can be written either linear equality or linear inequality while the decision variables are created in order to store optimal quantities after the optimization algorithm terminates. Consider the following LP model.

$$\begin{aligned}
&\text{Maximize/Minimize} && z = c_1x_1 + c_2x_2 + c_3x_3 + ... + c_nx_n && \\
&\text{subject to} && a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n && \leq b_1, \\
& && a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n && \leq b_2, \\
& && \vdots && \vdots \\
& && a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n && \leq b_m, \\
& && x_1, x_2, ..., x_n \geq 0. &&
\end{aligned} \tag{2.1}$$

For LP (2.1), $z = c_1x_1 + c_2x_2 + ... + c_nx_n$ is a linear combination of the decision variables and the objective coefficients called the linear objective function while

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &\leq b_1, \\
a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &\leq b_2, \\
\vdots &\quad \vdots \\
a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n &\leq b_m, \\
x_1, x_2, ..., x_n \geq 0, &
\end{aligned}$$

are called linear constraints. $x_1 \geq 0, x_2 \geq 0, ..., x_n \geq 0$ are called the non-negative constraints.

For LP (2.1), it can be written in the following vector/matrix form

$$\begin{aligned} &\text{Maximize/Minimize} && \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} && \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & && \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.2}$$

where $\mathbf{c}$ is a vector of coefficients of the objective function, $\mathbf{c} \in \mathbb{R}^n$,
$\mathbf{A}$ is a coefficient matrix of the constraints, $\mathbf{A} \in \mathbb{R}^{m \times n}$,
$\mathbf{b}$ is a right-hand side vector, $\mathbf{b} \in \mathbb{R}^m$,
$\mathbf{x}$ is a vector of decision variables, $\mathbf{x} \in \mathbb{R}^n$.

For the maximization of LP (2.2), it is called the canonical form of the maximization of the LP model that it can be written as follows:

$$\begin{aligned} &\text{Maximize} && \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} && \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & && \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.3}$$

For the minimization of the LP model in the canonical form, it can be written as follows:

$$\begin{aligned} &\text{Minimize} && \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} && \mathbf{A}\mathbf{x} \geq \mathbf{b}, \\ & && \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.4}$$

Meanwhile, the LP model in the standard form having equality constraints is used for the simplex method. It can be either the maximization of the LP model or the minimization of the LP model. In this dissertation, the standard form will be the maximization of the LP model which is written as follows:

$$\begin{aligned} &\text{Maximize} && \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & && \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.5}$$

where $\mathbf{b}$ is the non-negative vector.

For the minimization of the LP model, it can be converted to the standard form by multiplying -1 in the objective function. To minimize $\mathbf{c}^\top \mathbf{x}$ is equivalent to $-$ maximize $-\mathbf{c}^\top \mathbf{x}$. Moreover, all LP models can always be converted to the standard form using the following processes.

1. If the constraint is in the form $\mathbf{A}_{i:}\mathbf{x} \leq b_i$, the slack variable (s_i^+) is added to the constraint as follows: $\mathbf{A}_{i:}\mathbf{x} + s_i^+ = b_i$ where $s_i^+ \geq 0$.

2. If the constraint is in the form $\mathbf{A}_{i:}\mathbf{x} \geq b_i$, the surplus variable (s_i^-) is subtracted from the constraint as follows: $\mathbf{A}_{i:}\mathbf{x} - s_i^- = b_i$ where $s_i^- \geq 0$.

3. If a decision variable (x_i) is an unrestricted variable, it must be replaced by subtracting two newly defined variables x_i^+ and x_i^- where $x_i^+ \geq 0$ and $x_i^- \geq 0$.

2.4 Dual linear programming model

Modeling the LP model is to transform data from a problem into a mathematical model. The LP model created directly from the problem data is called the primal LP model while there exists a corresponding pair of the primal LP model called the dual LP model which the objective value of the optimal solution of both LP models will always be equal. Let the primal LP model be

$$\begin{aligned} \text{Maximize} \quad & c_1x_1 + c_2x_2 + ... + c_nx_n & \\ \text{subject to} \quad & a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n & \leq b_1, \\ & a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n & \leq b_2, \\ & \vdots & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n & \leq b_m, \\ & x_1, x_2, ..., x_n & \geq 0. \end{aligned} \tag{2.6}$$

Associated with the primal LP model, there is the corresponding dual LP model

written as

$$\begin{aligned} \text{Minimize} \quad & b_1y_1 + b_2y_2 + ... + b_my_m \\ \text{subject to} \quad & a_{11}y_1 + a_{21}y_2 + ... + a_{m1}y_m \quad \geq c_1, \\ & a_{12}y_1 + a_{22}y_2 + ... + a_{m2}y_m \quad \geq c_2, \\ & \qquad \vdots \qquad\qquad\qquad\quad \vdots \\ & a_{1n}y_1 + a_{2n}y_2 + ... + a_{mn}y_m \quad \geq c_n, \\ & y_1, y_2, ..., y_m \geq 0. \end{aligned} \tag{2.7}$$

Observe that the number of constraints in LP (2.6) is equal to the number of variables in LP (2.7) and vice versa. In addition, for each i in $\{1, 2, ..., m\}$, i^{th} constraint of LP (2.6) will correspond to i^{th} variable of LP (2.7). Similarly, for each j in $\{1, 2, ..., n\}$, j^{th} variable of LP (2.6) will correspond to j^{th} constraint of LP (2.7).

Therefore, it exhibits that a constraint in the primal LP model is related to the variable in the dual LP model likewise a variable in the primal LP model is related to the constraint in the dual LP model. If the primal LP model has a large number of constraints, then the dual LP model will have a large number of variables. Similarly, if the primal LP model has a large number of variables, then the dual LP model will have a large number of constraints too. In addition, for every primal LP model, the dual LP model which related to the primal LP model can always be constructed. The relationships of the primal LP model and the dual LP model are shown in Table 2.1.

Table 2.1: The relationships of the primal LP model and the dual LP model.

Primal LP model (Maximize)	Dual LP model (Minimize)
Constraint is in the form of " $\leq$ ".	The corresponding variable is ≥ 0.
Constraint is in the form of " $\geq$ ".	The corresponding variable is ≤ 0.
Constraint is in the form of " $=$ ".	The corresponding variable is unrestricted.
The corresponding variable is ≥ 0.	Constraint is in the form of " $\geq$ ".
The corresponding variable is ≤ 0.	Constraint is in the form of " $\leq$ ".
The corresponding variable is unrestricted.	Constraint is in the form of " $=$ ".

The solution of the LP model can be summarized as follows:

1. optimal solution: the LP model can find the best feasible solution,
2. infeasible solution: the LP model has no solution,
3. unbounded solution: For the maximization of the LP model, the value of the objective function can be increased infinitely. Vice versa, for the minimization of the LP model, the value of the objective function can be decreased infinitely.

Note the LP model which has an unbounded feasible region may not have an unbounded solution. Next, the relationship of the solution between the primal LP model and the dual LP model is explained as follows:

- The primal LP model has an optimal solution if and only if the dual LP model has an optimal solution which their optimal objective values are the same.
- If the primal (dual) LP model is infeasible, then the dual (primal) LP model can be either infeasible or unbounded.
- If primal (dual) LP model is unbounded, then the dual (primal) LP model is infeasible.

Example 2.1. Consider the following primal LP model:

$$\begin{array}{lll} \text{Maximize} & x_1 + 2x_2 - x_3 & \\ \text{subject to} & 2x_1 - x_2 & \geq 2, \\ & -x_1 + 3x_2 + 2x_3 & \leq 3, \\ & -3x_1 + x_2 + x_3 & = -5, \\ & x_1 \geq 0, x_2 \leq 0, x_3 & \text{unrestricted.} \end{array} \tag{2.8}$$

The dual LP model which corresponds with the primal LP (2.8) can be written as below:

$$\begin{array}{lll} \text{Minimize} & 2y_1 + 3y_2 - 5y_3 & \\ \text{subject to} & 2y_1 - y_2 - 3y_3 & \geq 1, \\ & -y_1 + 3y_2 + y_3 & \leq 2, \\ & 2y_2 + y_3 & = -1, \\ & y_1 \leq 0, y_2 \geq 0, y_3 & \text{unrestricted.} \end{array} \tag{2.9}$$

Theorems and corollaries which are related to the primal and dual models of the LP model are stated without proof below:

Theorem 2.1. If $\mathbf{x}$ is a feasible solution in LP (2.6) and if $\mathbf{y}$ is a feasible solution in LP (2.7), then $\mathbf{c}^\top\mathbf{x} \leq \mathbf{b}^\top\mathbf{y}$.

Corollary 2.1. If LP (2.6) and its dual (2.7) are both feasible then both are bounded feasible.

Corollary 2.2. If there exists feasible $\mathbf{x}^*$ and $\mathbf{y}^*$ for LP (2.6) and its dual (2.7) such that $\mathbf{c}^\top\mathbf{x}^* = \mathbf{b}^\top\mathbf{y}^*$, then both $\mathbf{x}^*$ and $\mathbf{y}^*$ are the optimal solution for LP (2.6) and LP (2.7), respectively.

The methods for solving the LP model proposed in the next section are separated into two groups: artificial-free variable methods and artificial variable methods. For the artificial variable method, artificial variables are added into the LP model to make it easier to provide a starting point for finding the solution. While the artificial-free variable method does not use any artificial variables. However, adding variables to the LP model will cause the size of the LP model to be expanded.

2.5 Artificial-free variable methods

The popular methods for solving the LP model without artificial variables are the graphical method, the simplex method with a feasible origin point, and the dual simplex method. For the graphical method, it is a popular method for the LP model having the number of decision variables no more than three. For the small-sized or medium-sized LP model, the simplex method is the popular method for solving it. It can solve the LP model without artificial variables when the initial basic feasible solution is found. Another method for solving the LP model without artificial variables is the dual simplex method. It is applied to solve the dual LP model by considering the primal simplex tableau. First, the mathematical terms discussed in this section are explained.

Consider an LP model as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.10}$$

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

The technical terms used in this section are explained as follows:

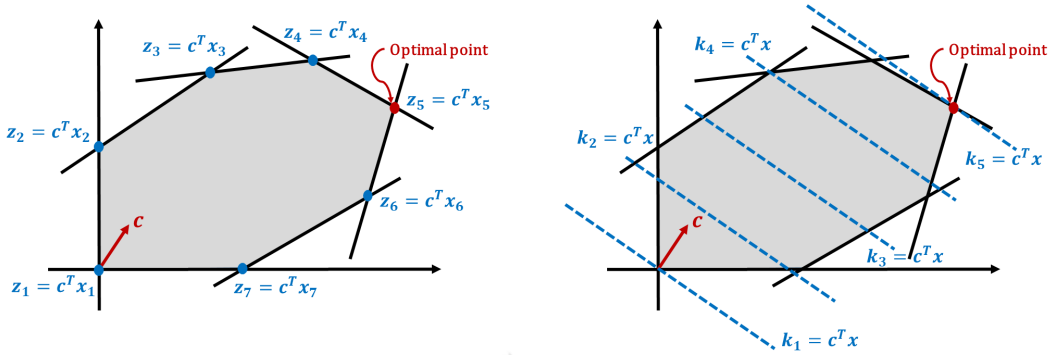
- **Feasible region** is the area that covers all possible solutions.
- **Iso-profit line** (for maximization model) is a linear equation resulting from the objective function that is assigned for a fixed constant. It will drag through all solution points in the graph that give the largest objective value function via the graphical method. Note that for the minimization model, the iso-profit line is called **the iso-cost line**.
- **Feasible solution** is any solution that is in accordance with the constraint function. Thus, the set of feasible solutions of LP (2.10) can be written as $F = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b} \text{ and } \mathbf{x} \geq \mathbf{0}\}$. **Infeasible solution** is any solution that is not in accordance with at least one constraint.

2.5.1 The graphical method

The graphical method is a method for solving the LP model by expressing in the graph form. Therefore, it is applied to the LP model with no more than three decision variables. Two popular graphical methods are a corner point method and the iso-profit method. The corner point method starts by creating the graph from relationships of constraints of the LP model, then it computes the objective value for all corner points in the feasible region. The optimal solution for the maximization LP model is the feasible corner point that gives the highest objective value shown in Figure 2.1(a). On the other hand, the optimal solution for the minimization LP model is the feasible corner point that gives the lowest objective value. The iso-profit method starts by moving the iso-profit line in the direction of the objective function until it reaches the top edge or top corner of the feasible region for the maximization LP model presented in Figure 2.1(b). Likewise, for the minimization

LP model, the iso-cost line will move in the opposite direction of the objective function.

point.png point.bb profit.png profit.bb



(a) The corner point method. **(b)** The iso-profit method.

Figure 2.1: The graphical method.

2.5.2 The simplex method

The simplex method starts with the LP model in the standard form as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.11}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b}$ is a non-negative vector in $\mathbb{R}^m$, and $\mathbf{A}$ is a matrix in $\mathbb{R}^{m \times n}$ with rank($\mathbf{A}$)$= m$. Let $m > n$ and $Index = \{1, 2, \ldots, m\}$.

The simplex method is easy to calculate when the LP model in the standard form is transformed to the simplex tableau. Let $\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{N} \end{bmatrix}$ where $\mathbf{B}$ is an $m \times m$ invertible matrix and $\mathbf{N}$ is an $m \times (n-m)$ matrix. Thus, $\mathbf{x}$ can be decomposed into $\mathbf{x_B}$ and $\mathbf{x_N}$ where $\mathbf{x_B}$ is a basic variable and $\mathbf{x_N}$ is a nonbasic variable. Moreover, $\mathbf{c}$ can be decomposed as $\mathbf{c_B^\top}$ and $\mathbf{c_N^\top}$, and letting $I_\mathbf{B}$ be the current set of the indices of the basic variables and $I_\mathbf{N}$

be the current set of the indices of the non-basic variables. Therefore,

$$\begin{aligned}
\mathbf{Ax} &= \mathbf{b} \\
\begin{bmatrix} \mathbf{B} & \mathbf{N} \end{bmatrix} \begin{bmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{bmatrix} &= \mathbf{b} \\
\mathbf{Bx_B} + \mathbf{Nx_N} &= \mathbf{b} \\
\mathbf{x_B} &= \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{Nx_N}
\end{aligned}$$

and

$$\begin{aligned}
z &= \begin{bmatrix} \mathbf{c_B}^\top & \mathbf{c_N}^\top \end{bmatrix} \begin{bmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{bmatrix} \\
z &= \mathbf{c_B}^\top \mathbf{x_B} + \mathbf{c_N}^\top \mathbf{x_N} \\
z &= \mathbf{c_B}^\top \left(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{Nx_N} \right) + \mathbf{c_N}^\top \mathbf{x_N} \\
z &= \mathbf{c_B}^\top \mathbf{B}^{-1}\mathbf{b} - \left(\mathbf{c_B}^\top \mathbf{B}^{-1}\mathbf{N} - \mathbf{c_N}^\top \right) \mathbf{x_N} \\
z + \left(\mathbf{c_B}^\top \mathbf{B}^{-1}\mathbf{N} - \mathbf{c_N}^\top \right) \mathbf{x_N} &= \mathbf{c_B}^\top \mathbf{B}^{-1}\mathbf{b} \\
z - \left(\mathbf{c_N}^\top - \mathbf{c_B}^\top \mathbf{B}^{-1}\mathbf{N} \right) \mathbf{x_N} &= \mathbf{c_B}^\top \mathbf{B}^{-1}\mathbf{b}.
\end{aligned}$$

Consequently, the initial simplex tableau is created in Table 2.2.

	$\mathbf{x_B}$	$\mathbf{x_N}$	RHS
z	$\mathbf{0}$	$-\left(\mathbf{c_N}^\top - \mathbf{c_B}^\top \mathbf{B}^{-1}\mathbf{N}\right)$	$\mathbf{c_B}^\top \mathbf{B}^{-1}\mathbf{b}$
$\mathbf{x_B}$	$\mathbf{I}$	$\mathbf{B}^{-1}\mathbf{N}$	$\mathbf{B}^{-1}\mathbf{b}$

Table 2.2: The initial simplex tableau.

For each iteration of the simplex tableau, $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b}$ and $\mathbf{x_N} = \mathbf{0}$, $\begin{bmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{bmatrix}$ is the basic solution of LP (2.11). If $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$, then $\mathbf{x}$ is called the basic feasible solution.

Let $z_0 = \mathbf{c}_\mathbf{B}^\top \mathbf{B}^{-1}\mathbf{b}$. Thus, $z = \mathbf{c}_\mathbf{B}^\top \mathbf{B}^{-1}\mathbf{b} + \left(\mathbf{c}_\mathbf{N}^\top - \mathbf{c}_\mathbf{B}^\top \mathbf{B}^{-1}\mathbf{N}\right)\mathbf{x_N} = z_0 + \left(\mathbf{c}_\mathbf{N}^\top - \mathbf{c}_\mathbf{B}^\top \mathbf{B}^{-1}\mathbf{N}\right)\mathbf{x_N}$ and $\mathbf{c}_\mathbf{B}^\top \mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_\mathbf{N}^\top$ is called the reduced cost. Let $\mathbf{c}_\mathbf{B}^\top \mathbf{B}^{-1}\mathbf{A}_{:j} - c_j = z_j - c_j$, for $j = 1, 2, ..., n$ $\mathbf{B}^{-1}\mathbf{b} = \bar{\mathbf{b}}$ and $\mathbf{y}_r = \mathbf{B}^{-1}\mathbf{A}_{:r}$ for $r = 1, 2, ..., m$. If $\mathbf{c}_\mathbf{N}^\top - \mathbf{c}_\mathbf{B}^\top \mathbf{B}^{-1}\mathbf{N} \leq \mathbf{0}$, the current basic feasible solution is optimal. Otherwise, the simplex method is performed to improve the objective value.

The basic steps of the standard simplex method by Dantzig's pivot rule are summarized (Maximization problem) as follows:

Step 0: (Initialization) Choose the initial basic feasible solution and construct the simplex tableau as follows:

	$\mathbf{x_B}$	$\mathbf{x_N}$	RHS
z	$\mathbf{0}$	$-\left(\mathbf{c}_\mathbf{N}^\top - \mathbf{c}_\mathbf{B}^\top \mathbf{B}^{-1}\mathbf{N}\right)$	$\mathbf{c}_\mathbf{B}^\top \mathbf{B}^{-1}\mathbf{b}$
$\mathbf{x_B}$	$\mathbf{I}$	$\mathbf{B}^{-1}\mathbf{N}$	$\mathbf{B}^{-1}\mathbf{b}$

Step 1: (Choice of the entering variable) Choose $z_r - c_r = \text{minimize}\{z_j - c_j \mid j \in I_N\}$. If $z_r - c_r \geq 0$, then $\begin{bmatrix}\mathbf{x_B} & \mathbf{x_N}\end{bmatrix}^\top$ is an optimal solution and the algorithm stops. Otherwise, go to Step 2.

Step 2: (Choice of the leaving variable) Let $\mathbf{y}_r = \mathbf{B}^{-1}\mathbf{A}_{:r}$. If $\mathbf{y}_r \leq \mathbf{0}$, the LP model has an unbounded optimal solution. Otherwise, the index of leaving variable $\mathbf{x_B}_\eta$ is calculated by the minimum ratio test as

$$\eta = \text{argmin}\left\{\frac{\bar{b}_i}{y_{r_i}} \;\middle|\; \bar{b}_i \geq 0 \text{ and } y_{r_i} > 0\right\},$$

where y_{r_i} is the i^{th} element of $\mathbf{y}_r$.

Step 3: (Pivoting) Perform the pivot operation using the entering variable and the leaving variable and repeat from step 1 until the optimal solution is found or the LP model has the unbounded optimal solution.

Example 2.2. Consider the following linear programming model.

$$\begin{aligned}
\text{Maximize} \quad & x_1 + 2x_2 - x_3 \\
\text{subject to} \quad & -3x_1 + x_2 - x_3 \leq 5, \\
& x_1 - 2x_2 - 4x_3 \leq 4, \\
& -4x_1 + 3x_3 \leq 5, \\
& 2x_2 + 2x_3 \leq 6, \\
& x_1, x_2, x_3 \geq 0.
\end{aligned} \tag{2.12}$$

The standard form of the linear programming model can be written as:

$$\begin{aligned}
\text{Maximize} \quad & x_1 + 2x_2 - x_3 \\
\text{subject to} \quad & -3x_1 + x_2 - x_3 + s_1 = 5, \\
& x_1 - 2x_2 - 4x_3 + s_2 = 4, \\
& -4x_1 + 3x_3 + s_3 = 5, \\
& 2x_2 + 2x_3 + s_4 = 6, \\
& x_1, x_2, x_3, s_1, s_2, s_3, s_4 \geq 0.
\end{aligned} \tag{2.13}$$

The identity matrix is an initial basis where the basic variables are defined as s_1, s_2, s_3, and s_4 and the non-basic variables are x_1, x_2, and x_3. Since $[x_1, x_2, x_3, s_1, s_2, s_3, s_4]^\top = [0, 0, 0, 5, 4, 5, 6]^\top \geq \mathbf{0}$, thus, it is an initial basic feasible solution. The first step, the initial simplex tableau is created as follows:

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

From this tableau, $z_r - c_r = \text{minimize}\{z_j - c_j \mid j \in I_{\mathbf{N}}\} = \text{minimize}\{-1, -2, 1\} = -2$ with $r = 2$. Since $z_2 - c_2 = -2$ which is a negative value. Thus, x_2 is the entering variable. Next, the leaving variable is calculated by the minimum ratio test. Since $\mathbf{y}_2 = \mathbf{B}^{-1}\mathbf{A}_{:2} = [1, -2, 0, 2]^\top$, so, the index of the leaving variable is calculated as $\eta = \text{argmin}\left\{\frac{\bar{b}_i}{y_{2_i}} \;\middle|\; \bar{b}_i \geq 0 \text{ and } y_{2_i} > 0\right\} = \text{argmin}\left\{\frac{5}{1}, \frac{6}{2}\right\} = 4$ that it corresponds to s_4.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	-1	0	3	0	0	0	1	6
s_1	-3	0	-2	1	0	0	-1/2	2
s_2	**1**	0	-2	0	1	0	1	10
s_3	-4	0	3	0	0	1	0	5
x_2	0	1	1	0	0	0	1/2	3

The second tableau has the entering variable as x_1 since it has the smallest reduced cost. The leaving variable is s_2 since it obtains the minimum ratio test from the Dantzig's pivot rule.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	0	0	1	0	1	0	2	16
s_1	0	0	-8	1	3	0	5/2	32
x_1	1	0	-2	0	1	0	1	10
s_3	0	0	-5	0	4	1	4	45
x_2	0	1	1	0	0	0	1/2	3

After the second simplex tableau is updated, the result tableau is the optimal tableau since the reduced costs of this tableau are non-negative. Thus, the solution of this model is $[x_1, x_2, x_3]^\top = [10, 3, 0]^\top$ with the objective value = 16.

From Example 2.2, since the origin point is feasible, the simplex method can start immediately. However, if the origin point is infeasible, the artificial variables are added into the LP model. Then either the two-phase simplex method or the big-M simplex method is applied to find the solution of the LP model which both methods are shown in Section 2.6.

2.5.3 The dual simplex method

The dual simplex method is another method for solving the dual linear programming model directly on the primal simplex tableau when the dual solution is feasible and the primal solution is infeasible. The process is operated until the primal solution is feasible while maintaining the feasibility of the dual solution.

Consider a linear programming model in the standard form as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.14}$$

The primal simplex tableau is created as follows:

	$\mathbf{x_B}$	$\mathbf{x_N}$	RHS
z	$\mathbf{0}$	$-\left(\mathbf{c_N^\top} - \mathbf{c_B^\top B^{-1} N}\right)$	$\mathbf{c_B^\top B^{-1} b}$
$\mathbf{x_B}$	$\mathbf{I}$	$\mathbf{B^{-1} N}$	$\mathbf{B^{-1} b}$

If $\mathbf{B^{-1}b} \geq \mathbf{0}$, then the primal solution of the tableau is feasible. Moreover, if $\mathbf{c_N^\top} - \mathbf{c_B^\top B^{-1} N} \leq \mathbf{0}$, then the current primal simplex tableau is optimal. Let $\mathbf{w}^\top = \mathbf{c_B^\top B^{-1}}$. Multiply $\mathbf{B}$ from the right and transpose both sides to get $\mathbf{c_B} = \mathbf{B}^\top \mathbf{w}$. At the optimal tableau, $\mathbf{c_N^\top} - \mathbf{c_B^\top B^{-1} N} = \mathbf{c_N^\top} - \mathbf{w}^\top \mathbf{N} \leq \mathbf{0}$ implies that $\mathbf{c_N^\top} \leq \mathbf{w}^\top \mathbf{N}$ or $\mathbf{c_N} \leq \mathbf{N}^\top \mathbf{w}$. Therefore, $\mathbf{c} \leq \mathbf{A}^\top \mathbf{w}$. Consequently, $\mathbf{w}^\top = \mathbf{c_B^\top B^{-1}}$ keeps a dual feasible solution. Likewise, if there are some elements of the vector $\mathbf{c_N^\top} - \mathbf{c_B^\top B^{-1} N}$ positive, then this dual solution is infeasible.

The basic steps of the dual simplex method are summarized (Maximization problem) as follows:

Step 0: (Initialization)

Create a simplex tableau by choosing a basis $\mathbf{B}$ which $z_j - c_j = \mathbf{c_B^\top B^{-1} A}_{:j} - \mathbf{c}_j \geq \mathbf{0}$ for all j.

	$\mathbf{x_B}$	$\mathbf{x_N}$	RHS
z	$\mathbf{0}$	$-\left(\mathbf{c_N^\top} - \mathbf{c_B^\top B^{-1} N}\right)$	$\mathbf{c_B^\top B^{-1} b}$
$\mathbf{x_B}$	$\mathbf{I}$	$\mathbf{B^{-1} N}$	$\mathbf{B^{-1} b}$

Step 1: (Choice of the leaving variable)

Let $\bar{\mathbf{b}} = \mathbf{B^{-1} b}$. Choose $\bar{b}_k = \text{minimize}\left\{\bar{b}_i \mid i \in \mathbf{I_B}\right\}$. If $\bar{b}_k \geq 0$, then $\begin{bmatrix}\mathbf{x_B} & \mathbf{x_N}\end{bmatrix}^\top$ is an optimal solution and the algorithm stops. Otherwise, go to Step 2.

Step 2: (Choice of the entering variable)

Let $\bar{\mathbf{y}}_k$ be the vector of the k^{th} row of the current tableau. If $\bar{\mathbf{y}}_k \geq \mathbf{0}$, the LP model has an unbounded optimal solution. Otherwise, the index of the entering variable is calculated as follows:

$$\mathbf{x}_r = \text{argmin}\left\{ \frac{-(z_j - c_j)}{\bar{y}_{k_j}} \;\middle|\; \bar{y}_{k_j} < 0 \text{ and } z_j - c_j \geq 0 \right\},$$

where $\bar{y}_{k_j}$ is the j^{th} element of $\bar{\mathbf{y}}_k$.

Step 3: (Pivoting)

Perform the pivot operation using the entering variable and the leaving variable and repeat from step 1 until the optimized solution is found or the LP model has an unbounded optimal solution.

Example 2.3. Consider a linear programming model as follows:

$$\begin{aligned}
\text{Maximize} \quad & -2x_1 - x_2 - 5x_3 \\
\text{subject to} \quad & -2x_1 + 5x_2 + s_1 = -3, \\
& x_1 - 8x_3 + s_2 = -12, \\
& 3x_2 - 4x_3 + s_3 = -5, \\
& -5x_1 - 2x_2 + 2x_3 + s_4 = -6, \\
& x_1, x_2, x_3, s_1, s_2, s_3, s_4 \geq 0.
\end{aligned} \tag{2.15}$$

For LP (2.15), the initial basic variables are defined as s_1, s_2, s_3 and s_4 and the non-basic variables are x_1, x_2. Therefore, the initial primal simplex tableau is created as follows:

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	2	1	5	0	0	0	0	0
s_1	-2	5	0	1	0	0	0	-3
s_2	1	0	**-8**	0	1	0	0	-12
s_3	0	3	-4	0	0	1	0	-5
s_4	-5	-2	2	0	0	0	1	-6

For the current simplex tableau the primal solution is infeasible and the dual solution is feasible. Thus, the dual simplex method can be used. Since $\bar{b}_k = \text{minimize}\left\{\bar{b}_i \mid i \in \mathbf{I_B}\right\}$ $= \{-3, -12, -5, -6\} = -12$. Then s_2 is the leaving variable and 2 is the index of the leaving variable. Next, the entering variable is calculated. Since $\bar{\mathbf{y}}_2 = [1, 0, -8, 0, 1, 0, 0]$. Therefore, x_r is calculated as follows: $x_r = \text{argmin}\left\{\dfrac{-(z_j - c_j)}{\bar{y}_{2_j}} \;\middle|\; \bar{y}_{2_j} < 0 \text{ and } z_j - c_j \geq 0\right\}$ $= \left\{\dfrac{-5}{-8}\right\} = \dfrac{5}{8}$. Thus, the entering variable is $\mathbf{x}_3$.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	21/8	1	0	0	5/8	0	0	-15/2
s_1	-2	5	0	1	0	0	0	-3
x_3	-1/8	0	1	0	-1/8	0	0	3/2
s_3	-1/2	3	0	0	-1/2	1	0	1
s_4	-19/4	**-2**	0	0	1/4	0	1	-9

The leaving variable is s_4 and the entering variable is x_2.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	1/4	0	0	0	3/4	0	1/2	-12
s_1	**-111/8**	0	0	1	5/8	0	5/2	-51/2
x_3	-1/8	0	1	0	- 1/8	0	0	3/2
s_3	-61/8	0	0	0	-1/8	1	3/2	-25/2
x_2	19/8	1	0	0	-1/8	0	-1/2	9/2

For the current simplex tableau, the leaving variable is s_1 and the entering variable is x_1.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	0	0	0	0	3/4	0	1/2	-25/2
x_1	1	0	0	0	0	0	-1/6	11/6
x_3	0	0	1	0	-1/8	0	0	7/4
s_3	0	0	0	-5/9	-1/2	1	1/8	3/2
x_2	0	1	0	1/6	0	0	0	1/7

Since both the primal solution and the dual solution are feasible. Thus, this simplex tableau is optimal with the optimal value $= \frac{-25}{2}$.

2.6 Artificial variable methods

Artificial variables are added to guarantee the solution of the extended LP model when the initial point is infeasible. For the standard simplex method, it starts with a feasible origin point. Thus, if the origin point is infeasible, the artificial variables are added

to the LP model. After that, either the two-phase simplex method or the big-M simplex method is applied. Similarly, with the interior point method, if an initial interior feasible point is not available, the artificial variables are added to the LP model to guarantee the existence of the interior feasible point.

2.6.1 The two-phase simplex method

The two-phase simplex method is a method for solving the LP model when the origin point is infeasible. It is divided into two phases that are phase 1 applied to find an initial basic feasible solution and phase 2 performed the simplex method using the basic feasible solution from phase 1.

Consider the LP model as following:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.16}$$

where $\mathbf{b}$ is a non-negative vector.

If the origin point of LP (2.16) is feasible, the simplex method can start immediately. Otherwise, artificial variable $\mathbf{x}_a$ is added to the constraints of LP (2.16) as $\mathbf{A}\mathbf{x} + \mathbf{x}_a = \mathbf{b}, \mathbf{x}, \mathbf{x}_a \geq \mathbf{0}$. After that, the artificial variable LP model is created as follows:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{x}_a \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{x}_a = \mathbf{b}, \\ & \mathbf{x}, \mathbf{x}_a \geq \mathbf{0}. \end{aligned} \tag{2.17}$$

The initial basic feasible solution is given by $\mathbf{x}_a = \mathbf{b} \geq \mathbf{0}$ and $\mathbf{x} = 0$. Then the simplex method is applied to find the solution of LP (2.17). This process is called phase 1 of the two-phase simplex method. At optimality, if $\mathbf{x}_a \neq \mathbf{0}$, the solution of LP (2.16) is infeasible. Otherwise, the current basic feasible solutions are set as an initial basic feasible solution of phase 2. Then all artificial variables are removed from the LP model

along with the original objective function is restored. Assume that all artificial variables left the basis. After that, phase 2 of the two-phase simplex method is performed by the simplex method with the current initial basic feasible solution.

Example 2.4. Consider the following linear programming model.

$$\begin{aligned}
\text{Maximize} \quad & x_1 - x_2 + 2x_3 & \\
\text{subject to} \quad & -x_1 - 2x_3 & \leq 8, \\
& 2x_1 + x_2 + x_3 & \leq 10, \\
& -2x_2 + 5x_3 & \geq 5, \\
& 3x_1 + 4x_2 - 2x_3 & \geq 6, \\
& x_1, x_2, x_3 & \geq 0.
\end{aligned} \tag{2.18}$$

LP (2.18) will be transformed to the standard form by adding the slack variables (s_1 and s_2) and the surplus variables (s_3 and s_4) into LP (2.18) as follows:

$$\begin{aligned}
\text{Maximize} \quad & x_1 - x_2 + 2x_3 & \\
\text{subject to} \quad & -x_1 - 2x_3 + s_1 & = 8, \\
& 2x_1 + x_2 + x_3 + s_2 & = 10, \\
& -2x_2 + 5x_3 - s_3 & = 5, \\
& 3x_1 + 4x_2 - 2x_3 - s_4 & = 6, \\
& x_1, x_2, x_3, s_1, s_2, s_3, s_4 & \geq 0,
\end{aligned} \tag{2.19}$$

Since the origin point is infeasible. Thus, artificial variables (a_1, a_2) are added to

make the origin point feasible for the extended LP model.

$$\begin{aligned}
\text{Minimize} \quad & a_1 + a_2 \\
\text{subject to} \quad & -x_1 - 2x_3 + s_1 = 8, \\
& 2x_1 + x_2 + x_3 + s_2 = 10, \\
& -2x_2 + 5x_3 - s_3 + a_1 = 5, \\
& 3x_1 + 4x_2 - 2x_3 - s_4 + a_2 = 6, \\
& x_1, x_2, x_3, s_1, s_2, s_3, s_4 \geq 0,
\end{aligned} \tag{2.20}$$

After that, phase 1 of the two-phase simplex method is performed to find the initial basic feasible solution by pushing all artificial variables out of the basis. For LP (2.20), the initial basic variables are s_1, s_2, a_1 and a_2 while the nonbasic variables are x_1, x_2, x_3, s_3 and x_4. Therefore, the initial simplex tableau of phase 1 is constructed as follows:

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	a_1	a_2	RHS
z	-3	-2	-3	0	0	1	1	0	0	-11
s_1	-1	0	-2	1	0	0	0	0	0	8
s_2	2	1	1	0	1	0	0	0	0	10
a_1	0	-2	5	0	0	-1	0	1	0	5
a_2	**3**	4	-2	0	0	0	-1	0	1	6

The minimum of the negative reduced cost of the non-basic variables is -3 which corresponding with x_1 and x_3. Thus, both x_1 and x_3 can be the entering variable. For this example, x_1 is chosen to enter the basis. For the leaving variable which corresponding with the minimum ratio test is a_2. After the simplex tableau is updated, the second tableau is shown as follows:

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	a_1	a_2	RHS
z	0	2	-5	0	0	1	0	0	1	-5
s_1	0	4/3	-8/3	1	0	0	- 1/3	0	1/3	10
s_2	0	-5/3	7/3	0	1	0	2/3	0	- 2/3	6
a_1	0	-2	**5**	0	0	-1	0	1	0	5
x_1	1	4/3	- 2/3	0	0	0	- 1/3	0	1/3	2

For the second tableau, the entering variable is x_3 and the leaving variable is a_1.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	a_1	a_2	RHS
z	0	0	0	0	0	0	0	1	1	0
s_1	0	1/4	0	1	0	- 1/2	- 1/3	1/2	1/3	38/3
s_2	0	- 3/4	0	0	1	1/2	2/3	- 1/2	- 2/3	11/3
x_3	0	- 2/5	1	0	0	- 1/5	0	1/5	0	1
x_1	1	1	0	0	0	- 1/7	- 1/3	1/7	1/3	8/3

After the second tableau is updated, the basis of the simplex tableau does not have any artificial variables. Therefore, phase 1 terminates and the current basic variable is set as the initial basic feasible solution of LP (2.20). Next, the simplex tableau with the initial basic feasible variables s_1, s_2, x_3 and x_1 is created in order to start phase 2 of the two-phase simplex method as follows:

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	0	5/4	0	0	0	-1/2	-1/3	14/3
s_1	0	1/4	0	1	0	-1/2	-1/3	38/3
s_2	0	-3/4	0	0	1	**1/2**	2/3	11/3
x_3	0	-2/5	1	0	0	-1/5	0	1
x_1	1	1	0	0	0	-1/7	-1/3	8/3

For this tableau, the entering variable is s_3 and the leaving variable is s_2. After pivoting, the simplex tableau is updated as follows:

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	0	3/7	0	0	8/7	0	3/7	62/7
s_1	0	- 4/7	0	1	8/7	0	3/7	118/7
s_3	0	-11/7	0	0	15/7	1	10/7	55/7
x_3	0	-5/7	1	0	3/7	0	2/7	18/7
x_1	1	6/7	0	0	2/7	0	-1/7	26/7

Since the reduced costs of the current tableau are non-negative, meaning no variable can enter and improve the objective value. Hence, this simplex tableau is the optimal tableau with the objective value is 62/7 and $[x_1, x_2, x_3]^\top = [26/7, 0, 18/7]^\top$.

2.6.2 The big-M simplex method

The big-M simplex method is another method for solving the LP model which the origin point is infeasible.

LP (2.16) is assumed that the origin point is infeasible. Therefore, the artificial variable is added to the LP model with penalty M where M is a very large positive number as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \mathbf{x} - M\mathbf{1}^\top \mathbf{x}_a \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{x}_a = \mathbf{b}, \\ & \mathbf{x}, \mathbf{x}_a \geq \mathbf{0}. \end{aligned} \tag{2.21}$$

After that, the simplex method is performed to find the solution of LP (2.21). If the solution of LP (2.21) is an optimal solution with $\mathbf{x}_a = \mathbf{0}$, the solution of the original LP (2.16) is feasible and the optimal solution is found. If the solution of LP (2.21) is an optimal solution with $\mathbf{x}_a \neq \mathbf{0}$, then the original LP (2.16) is infeasible. While if the solution of LP (2.21) is unbounded with $\mathbf{x}_a = \mathbf{0}$, the solution of the original LP (2.16) is unbounded. And if the solution of LP (2.21) is unbounded with $\mathbf{x}_a \neq \mathbf{0}$, the solution of the original LP (2.16) is infeasible.

Example 2.5. The standard form of the LP model can be represented as follows:

$$\begin{aligned} \text{Maximize} \quad & x_1 - x_2 + 2x_3 \\ \text{subject to} \quad & -x_1 - 2x_3 + s_1 = 8, \\ & 2x_1 + x_2 + x_3 + s_2 = 10, \\ & -2x_2 + 5x_3 - s_3 = 5, \\ & 3x_1 + 4x_2 - 2x_3 - s_4 = 6, \\ & x_1, x_2, x_3, s_1, s_2, s_3, s_4 \geq 0. \end{aligned} \tag{2.22}$$

The big-M simplex method starts by adding artificial variables to the model shown

below.

$$\begin{aligned}
\text{Maximize} \quad & x_1 - x_2 + 2x_3 - Ma_1 - Ma_2 \\
\text{subject to} \quad & -x_1 - 2x_3 + s_1 = 8, \\
& 2x_1 + x_2 + x_3 + s_2 = 10, \\
& -2x_2 + 5x_3 - s_3 + a_1 = 5, \\
& 3x_1 + 4x_2 - 2x_3 - s_4 + a_2 = 6, \\
& x_1, x_2, x_3, s_1, s_2, s_3, s_4 \geq 0,
\end{aligned} \tag{2.23}$$

where M is a very large positive number.

The initial simplex tableau can be created as follows:

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	a_1	a_2	RHS
z	-1	1	-2	0	0	0	0	M	M	0
s_1	-1	0	-2	1	0	0	0	0	0	8
s_2	2	1	1	0	1	0	0	0	0	10
a_1	0	-2	**5**	0	0	-1	0	1	0	5
a_2	3	4	-2	0	0	0	-1	0	1	6

The minimum of the negative reduced cost is -2 and its column index is 3. Therefore, the entering variable is x_3. The row index of the minimum ratio is 3. Thus, the leaving variable is a_1.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	a_1	a_2	RHS
z	-1	1/5	0	0	0	-2/5	0	M+2/5	M	2
s_1	-1	-4/5	0	1	0	-2/5	0	2/5	0	10
s_2	2	7/5	0	0	1	1/5	0	-1/5	0	9
x_3	0	-2/5	1	0	0	-1/5	0	1/5	0	1
a_2	**3**	16/5	0	0	0	-2/5	-1	2/5	1	8

For this tableau, the entering variable is x_1 and the leaving variable is a_2.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	a_1	a_2	RHS
z	0	5/4	0	0	0	-1/2	-1/3	M+19/35	M+1/3	14/3
s_1	0	1/4	0	1	0	-1/2	-1/3	1/2	1/3	38/3
s_2	0	-3/4	0	0	1	**1/2**	2/3	-1/2	-2/3	11/3
x_3	0	-2/5	1	0	0	-1/5	0	1/5	0	1
x_1	1	1	0	0	0	-1/7	-1/3	1/7	1/3	8/3

The entering variable is s_3 and the leaving variable is s_2.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	a_1	a_2	RHS
z	0	3/7	0	0	8/7	0	3/7	M+3/70	M-8/21	62/7
s_1	0	-4/7	0	1	8/7	0	3/7	0	-3/7	118/7
s_3	0	-11/7	0	0	15/7	1	10/7	-1	-10/7	55/7
x_3	0	-5/7	1	0	3/7	0	2/7	0	-2/7	18/7
x_1	1	6/7	0	0	2/7	0	-1/7	0	1/7	26/7

Since the reduced cost of the simplex tableau is positive. Hence, the optimal solution has been reached with the value of variables as $x_1 = 26/7, x_2 = 0$ and $x_3 = 18/7$ and the objective value as 62/7.

2.6.3 The interior-point method

The idea of the interior-point method is proposed which differs to the simplex method. It starts from moving an interior feasible point along with the direction which improved the objective value without crossing the boundary of the feasible region. The process is repeated until the interior point method converges to the optimal point. Traditionally, the Karmarkar's method has been proposed and later on the primal affine scaling method and the gravitational method are proposed.

2.6.3.1 The primal affine scaling method

Consider the standard form LP model as follows:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.24}$$

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

If $\mathbf{x}^k$ is feasible for LP (2.24) and $\mathbf{x}^k > \mathbf{0}$, then $\mathbf{x}^k$ is called the interior feasible point. The direction applied to improve the current point is $-\mathbf{c}$ since it can reduce the objective value.

The detail of the primal affine scaling algorithm with four steps are explicated as the following.

Step 1: Define the parameters and an initial feasible point.

Set $k = 0$, $\varepsilon > 0$ and $0 < \alpha < 1$.

Determine $\mathbf{x}^0$ which $\mathbf{x}^0 > \mathbf{0}$ and $\mathbf{A}\mathbf{x}^0 = \mathbf{b}$.

Step 2: Check the optimality.

Compute $\mathbf{g}^k = (\mathbf{A}\mathbf{X}_k^2\mathbf{A}^\top)^{-1}\mathbf{A}\mathbf{X}_k^2\mathbf{c}$

where $\mathbf{X}_k = diag(\mathbf{x}^k) = \begin{bmatrix} x_1^k & 0 & \cdots & 0 \\ 0 & x_2^k & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & x_n^k \end{bmatrix}$ and $\mathbf{h}^k = \mathbf{c} - \mathbf{A}^\top\mathbf{g}^k$.

If $\mathbf{h}^k \geq \mathbf{0}$ and $\mathbf{e}^\top\mathbf{X}_k\mathbf{h}^k \leq \varepsilon$ where $\mathbf{e}^\top = [1, 1, ..., 1]^\top$, then the algorithm stops. Otherwise, go to Step 3.

Step 3: Find the direction.

Compute $\mathbf{d}_y^k = -\mathbf{X}_k\mathbf{h}^k$.

If $\mathbf{d}_y^k > \mathbf{0}$, then the algorithm stops with the unbounded solution.

If $\mathbf{d}_y^k = \mathbf{0}$, then the algorithm stops.

Otherwise, go to step 4.

Step 4: Find the step length and update the current point.

Compute $\alpha_k = \min_i \left\{ \frac{\alpha}{-(d_y^k)_i} \middle| (d_y^k)_i < 0 \right\}$.

Compute $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k\mathbf{X}_k\mathbf{d}_y^k$.

Set $k = k + 1$ and go to Step 2.

The first step of the implementation of the affine scaling method is the finding of the initial feasible point. If the starting interior feasible point is found, the process of the affine scaling can start immediately. Otherwise, the two-phase affine scaling method is performed. The two-phase affine scaling method is separated as phase 1 and phase 2. In phase 1, it finds an initial strictly interior feasible point. For phase 2, it uses the result from phase 1 to initialize the affine scaling method to find the optimal solution. Phase 1 of the two-phase affine scaling method is explained next.

Let $\mathbf{u}$ be any positive vector and let $\mathbf{t} = \mathbf{b} - \mathbf{A}\mathbf{u}$. If $\mathbf{t} = \mathbf{0}$, then $\mathbf{u}$ is a strictly interior feasible point. Thus, the initial interior feasible point ($\mathbf{x}^0$) is set as $\mathbf{u}$ and then the phase 2 of the affine scaling method is performed. For case of $\mathbf{t} \neq \mathbf{0}$, the artificial method is constructed as follows:

$$\begin{aligned} \text{Minimize} \quad & x_{n+1} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{t}x_{n+1} = \mathbf{b}, \\ & \mathbf{x}, x_{n+1} \geq \mathbf{0}, \end{aligned} \tag{2.25}$$

Artificial LP (2.25) has an obvious strictly interior feasible point as

$$\begin{bmatrix} \mathbf{x} \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^0 \\ 1 \end{bmatrix}.$$

Thus, it has the optimal solution, However, if the optimal value of artificial LP (2.25) is strictly positive, then the original model is infeasible.

Example 2.6. Consider a linear programming as follows:

$$\begin{aligned} \text{Minimize} \quad & -18x_1 + 2x_2 \\ \text{subject to} \quad & 2x_2 && \leq 15, \\ & 2x_1 - 3x_2 && \leq 6, \\ & -2x_1 - x_2 && \leq 12, \\ & x_1, x_2 && \geq 0. \end{aligned} \tag{2.26}$$

The LP model is transformed into the standard form as follows:

$$\begin{aligned} \text{Minimize} \quad & -18x_1 + 2x_2 \\ \text{subject to} \quad & 2x_2 + x_3 && = 15, \\ & 2x_1 - 3x_2 + x_4 && = 6, \\ & -2x_1 - x_2 + x_5 && = 12, \\ & x_1, x_2, x_3, x_4, x_5 && \geq 0. \end{aligned} \tag{2.27}$$

Thus, $\mathbf{A} = \begin{bmatrix} 0 & 2 & 1 & 0 & 0 \\ 2 & -3 & 0 & 1 & 0 \\ 2 & -1 & 0 & 0 & 1 \end{bmatrix}$, $\mathbf{b} = [15, 6, 12]^\top$ and $\mathbf{c} = [-18, 2, 0, 0, 0]^\top$.

Let $x^0 = [1, 1, 13, 7, 11]^\top$ where it is feasible.

Hence, $\mathbf{X}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 13 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 11 \end{bmatrix}$ and α is defined as 0.99.

Iteration 1:

$\mathbf{g}^0 = (\mathbf{A}\mathbf{X}_0^2\mathbf{A}^\top)^{-1}\mathbf{A}\mathbf{X}_0^2\mathbf{c} = [-0.0024, -0.6477, -0.2656]^\top$ and

$\mathbf{h}^0 = \mathbf{c} - \mathbf{A}^\top\mathbf{g}^0 = [-16.1733, -0.2038, 0.0024, 0.6476, 0.2656]^\top$.

Since, elements in $\mathbf{h}^0$ are not non-negative and $\mathbf{e}^\top\mathbf{X}_0\mathbf{h}^0 = -8.8901$.

Thus, the current solution is not the optimal solution.

The direction is calculated as follows:

$\mathbf{d}_y^0 = -\mathbf{X}_0\mathbf{h}^0 = [16.1734, 0.2038, -0.0314, -4.5336, -2.9221]^\top$.

Since α is defined as 0.99, thus the step-length

$$\alpha_0 = \frac{0.99}{-(-4.5336)} = 0.2184.$$

Hence, the new solution is

$\mathbf{x}^1 = \mathbf{x}^0 + \alpha_0\mathbf{X}_0\mathbf{d}_y^0 = [4.5318, 1.0445, 12.9110, 0.0700, 3.9810]^\top$.

Iteration 2:

$\mathbf{g}^1 = (\mathbf{A}\mathbf{X}_1^2\mathbf{A}^\top)^{-1}\mathbf{A}\mathbf{X}_1^2\mathbf{c} = [-0.2302, -5.8754, -2.4235]^\top$ and

$\mathbf{h}^1 = \mathbf{c} - \mathbf{A}^\top\mathbf{g}^1 = [-1.4023, -17.5891, 0.2302, 5.8754, 2.435]^\top$.

Since, elements in $\mathbf{h}^1$ are not non-negative and $\mathbf{e}^\top\mathbf{X}_1\mathbf{h}^1 = -11.6951$.

Thus, the current solution is not the optimal solution.

The direction is calculated as follows:

$\mathbf{d}_y^1 = -\mathbf{X}_1\mathbf{h}^1 = [6.3549, 18.3719, -2.9726, -0.4113, -9.6478]^\top$.

Since α is defined as 0.99, thus the step-length

$$\alpha_1 = \frac{0.99}{-(-9.6478)} = 0.1026.$$

Hence, the new solution is

$\mathbf{x}^2 = \mathbf{x}^1 + \alpha_1\mathbf{X}_1\mathbf{d}_y^1 = [7.4869, 3.0136, 8.9728, 0.0670, 0.0398]^\top$.

The algorithm will be continued until it converges to optimal solution $\mathbf{x}^* = [9.75, 7.5]^\top$ with the optimal value = -160.5 which it can be shown in Figure 2.2.

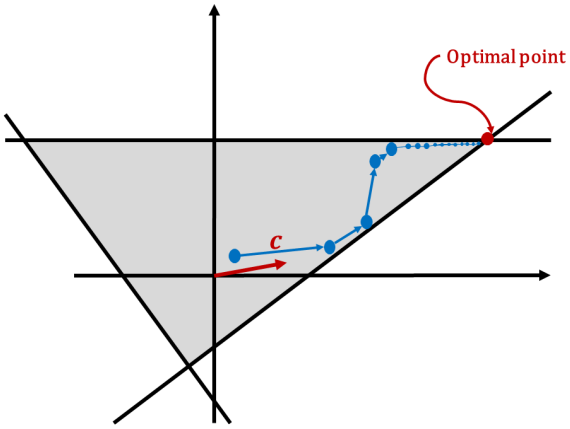


Figure 2.2: Example of the interior-point method.

2.6.3.2 The gravitational method

Consider a standard form of the LP model as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{b}^\top \mathbf{y} \\ \text{subject to} \quad & \mathbf{A}^\top \mathbf{y} = \mathbf{c}, \\ & \mathbf{y} \geq \mathbf{0}, \end{aligned} \tag{2.28}$$

where $\mathbf{b} \in \mathbf{R}^m$, $\mathbf{y} \in \mathbf{R}^m$, $\mathbf{A}^\top$ is an matrix of $n \times m$ and $\mathbf{c} \in \mathbf{R}^n$.

The dual model of LP (2.28) can be written as

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{b}. \end{aligned} \tag{2.29}$$

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

The gravitational method is applied to the dual LP model of LP (2.28) which is LP (2.29). A strict interior feasible point $\mathbf{x}^0$ is defined in the first corresponding to $\mathbf{x}^0$ corresponds to $\mathbf{A}\mathbf{x}^0 > \mathbf{b}$. The process of the gravitational method is an iterative method which consists of two principle stages, i.e. the initialize stage and the iterative stage.

The initialize stage:

The ball of radius r, which has $\mathbf{x}^0$ as a center is created. Radius r is selected to correspond with

$$0 < r < \min\left\{ \frac{\mathbf{A}_{i:}^{\top}\mathbf{x}^0 - \mathbf{b}_i}{\|\mathbf{A}_{i:}\|} \;\middle|\; i = 1, 2, ..., m \right\},$$

where $\|\cdot\|$ is the Euclidean norm.

Note: If the chosen radius is sufficiently small, then the solution from the gravitational method will be an optimal solution of LP (2.29).

The iterative stage:

For the process in this stage, it is separated in three stages, i.e. the first stage is the direction finding, the second stage is finding the step length and the last stage is updating the center of the ball.

The first stage of finding the direction

The initial gravitational direction $\mathbf{d}^0$ is defined as $\dfrac{-\mathbf{c}}{\|\mathbf{c}\|}$.

Let $\mathbf{I}_{J(\mathbf{x}^k)}$ be the index set of touching constraints for the ball centered at $\mathbf{x}^k$.

Thus, $\mathbf{I}_{J(\mathbf{x}^k)} = \left\{ i \;\middle|\; \dfrac{\mathbf{A}_{i:}^{\top}\mathbf{x}^k - \mathbf{b}_i}{\|\mathbf{A}_{i:}\|} = r \right\}$.

The gravitational direction is determined by

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^{\top}\mathbf{d} \\ \text{subject to} \quad & \mathbf{A}_{\mathbf{I}_{J(\mathbf{x}^k)}}\mathbf{d} \geq \mathbf{0}, \\ & 1 - \mathbf{d}^{\top}\mathbf{d} \geq \mathbf{0}. \end{aligned} \tag{2.30}$$

However, model (2.30) is equivalent to the quadratic model.

$$\begin{aligned} \text{Minimize} \quad & (\mathbf{c}^{\top} - \eta^{\top}\mathbf{A}_{\mathbf{I}_{J(\mathbf{x}^k)}})(\mathbf{c}^{\top} - \eta^{\top}\mathbf{A}_{\mathbf{I}_{J(\mathbf{x}^k)}})^{\top} \\ \text{subject to} \quad & \eta \geq \mathbf{0}, \end{aligned} \tag{2.31}$$

where η is the column vector $(\eta_i \mid i \in \mathbf{I}_{J(\mathbf{x}^k)})$.

If η^* is an optimal solution of the quadratic model (2.31), then $\xi^\top = \mathbf{c}^\top - \eta^{*\top}\mathbf{A}_{\mathbf{I}_{J(\mathbf{x}^k)}}$. For case of $\xi \neq \mathbf{0}$, the direction of the gravitational method is defined as $\mathbf{d} = \dfrac{-\xi}{\|\xi\|}$. In another case, the ball can not move by the gravitational descent and the algorithm stops.

The second stage of finding the step length

After the direction of the gravitational method is found, the step length applied to move the ball is calculated. Since the ball needs to be inside the feasible region of LP (2.29) always. Thus, for all iterations, the ball with center x^k, it has to certify that

$$\frac{\mathbf{A}_{i:}^\top \mathbf{x}^k - \mathbf{b}_i}{\|\mathbf{A}_{i:}\|} \geq r \text{ for all } i = 1, 2, ..., m.$$

Let $\mathbf{I}_{JB(\mathbf{d}^k)}$ be the index set of blocking constraints in direction $\mathbf{d}^k$ which is defined as $\mathbf{I}_{JB(\mathbf{d}^k)} = \{i \mid \mathbf{A}_{i:}^\top \mathbf{d}^k < 0\}$.

If $\mathbf{I}_{JB(\mathbf{d}^k)} = \emptyset$, then the step length for the gravitational direction is ∞.

Thus, the solution of LP (2.29) is unbounded which can conclude that the primal LP model is infeasible.

If $\mathbf{I}_{JB(\mathbf{d}^k)} \neq \emptyset$ and $\mathbf{I}_{J(\mathbf{x}^k)} \cap \mathbf{I}_{JB(\mathbf{d}^k)} = \emptyset$, then the step length is calculated as

$$\theta = \min \left\{ \frac{\mathbf{A}_{i:}^\top \mathbf{x}^k - \mathbf{b}_i - r\|\mathbf{A}_{i:}\|}{-\mathbf{A}_{i:}^\top \mathbf{d}^k} \;\middle|\; i \in \mathbf{I}_{JB(\mathbf{d}^k)} \right\}.$$

The last stage of updating the center of the ball

The ball is moved along the direction of gravitational method $\mathbf{d}^k$ with step length θ which the center of the new ball is updated as follows:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \theta \mathbf{d}^k.$$

To apply the gravitational method on LP (2.29), a strictly interior feasible point must be identified in the first stage. If it is available, the gravitational method can start immediately. Otherwise, the artificial gravitational LP model is constructed by adding artificial variable x_{n+1} as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \mathbf{x} + M x_{n+1} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{e} x_{n+1} \geq \mathbf{b}, \\ & x_{n+1} \geq 0, \end{aligned} \tag{2.32}$$

where $\mathbf{e} = [1, 1, ..., 1]^\top \in \mathbf{R}^m$ and M is a large positive number.

Let $x^0_{n+1} > \max\{0, b_1, b_2, ..., b_m\}$. Then $[0, 0, ..., x^0_{n+1}]^\top$ is a strictly interior feasible point of artificial gravitational LP (2.32). Thus, LP (2.32) always has a solution. If $\mathbf{c} \neq \mathbf{0}$, the gravitational method is performed to find the solution of LP (2.32). Otherwise, every feasible solution of LP (2.29) is the optimal solution including the initial interior feasible solution which it makes the solution of LP (2.28) $\mathbf{y} = \mathbf{0}$ as the optimal solution.

Example 2.7. Consider a linear programming as follows:

$$\begin{aligned} \text{Maximize} \quad & y_1 + y_2 \\ \text{subject to} \quad & y_1 + 4y_2 \leq 12, \\ & y_1 - 2y_2 \leq 6, \\ & y_1 \geq 0, \\ & y_2 \geq 0. \end{aligned} \tag{2.33}$$

LP (2.33) is transformed to the standard form by adding the slack variables.

$$\begin{aligned} \text{Maximize} \quad & y_1 + y_2 \\ \text{subject to} \quad & y_1 + 4y_2 + s_1 = 12, \\ & y_1 - 2y_2 + s_2 = 6, \\ & y_1, y_2, s_1, s_2 \geq 0. \end{aligned} \tag{2.34}$$

The dual LP model of LP (2.34) can be written as

$$\begin{aligned} \text{Minimize} \quad & 12x_1 + 6x_2 \\ \text{subject to} \quad & x_1 + x_2 \quad \geq 1, \\ & 4x_1 - 2x_2 \quad \geq 1, \\ & x_1 \quad \geq 0, \\ & x_2 \quad \geq 0. \end{aligned} \tag{2.35}$$

From LP (2.35), $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 4 & -2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ and $\mathbf{c} = \begin{bmatrix} 12 \\ 6 \end{bmatrix}$.

Let $\mathbf{x}^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ be the strict interior point.

The initialize stage:

$$\min \left\{ \frac{\mathbf{A}_{i:}^{\top}\mathbf{x}^0 - \mathbf{b}_i}{\|\mathbf{A}_{i:}\|} \,\middle|\, i = 1, 2, ..., m \right\}$$

$$= \min \left\{ \frac{[1,1][1,1]^{\top} - 1}{\sqrt{1^2 + 1^2}}, \frac{[4,-2][1,1]^{\top} - 1}{\sqrt{4^2 + (-2)^2}}, \frac{[1,0][1,1]^{\top} - 0}{\sqrt{1^2 + 0^2}}, \frac{[0,1][1,1]^{\top} - 0}{\sqrt{0^2 + 1^2}} \right\}$$

=0.2236.

Thus, for this example, the radius of the ball is defined as 0.01.

The iterative stage:

Iteration 1:

The initial direction of the gravitational method is

$$\mathbf{d}^0 = \frac{-\mathbf{c}}{\|\mathbf{c}\|} = \frac{-\begin{bmatrix}12\\6\end{bmatrix}}{\sqrt{12^2+6^2}} = \begin{bmatrix}-0.8944\\-0.4472\end{bmatrix}.$$

Constraint No (i)	$\mathbf{A}_{i:}$	$\mathbf{A}_{i:}^\top \mathbf{d}^0$
1	$[1,1]^\top$	1.3416
2	$[4,-2]^\top$	-2.6832
3	$[1,0]^\top$	-0.8944
4	$[0,1]^\top$	-0.4472

Let $\mathbf{I}_{JB(\mathbf{d}^0)} = \{i \mid \mathbf{A}_{i:}^\top \mathbf{d}^0 < 0\} = \{2,3,4\}$.

Compute $\theta = \min\left\{ \dfrac{\mathbf{A}_{i:}^\top \mathbf{x}^0 - \mathbf{b}_i - r\|\mathbf{A}_{i:}\|}{-\mathbf{A}_{i:}^\top \mathbf{d}^0} \;\middle|\; i \in \mathbf{I}_{JB(\mathbf{d}^0)} \right\}$.

$= \min\{0.3560, 1.1068, 2.2137\} = 0.3560.$

Thus, $\mathbf{x}^1 = \mathbf{x}^0 + \theta\mathbf{d}^0 = \begin{bmatrix}1\\1\end{bmatrix} + 0.3560 \begin{bmatrix}-0.8944\\-0.4472\end{bmatrix} = \begin{bmatrix}0.6815\\0.8407\end{bmatrix}$.

Iteration 2:

$$\mathbf{I}_{J(\mathbf{x}^1)} = \left\{ i \;\middle|\; \frac{\mathbf{A}_{i:}^\top \mathbf{x}^1 - \mathbf{b}_i}{\|\mathbf{A}_{i:}\|} = r \right\} = \{2\}.$$

Thus, the direction of the gravitational method is calculated as follows:

$$\begin{aligned} &\text{Minimize} \quad ([12,6] - \eta_2[4,-2])([12,6] - \eta_2[4,-2])^\top \\ &\text{subject to} \quad \eta_2 \geq 0 \end{aligned} \tag{2.36}$$

For (2.36), solution η_2 is $\dfrac{72}{40}$.

Since $\xi^\top = \mathbf{c}^\top - [\eta_2]\mathbf{A}_{\mathbf{I}_{J(\mathbf{x}^1)}} = [12,6] - \left[\dfrac{72}{40}\right][4,-2] = [4.8, 9.6]$ is not zero vector,

the direction for this iteration $\mathbf{d}^1$ is $\dfrac{-\begin{bmatrix}4.8\\9.6\end{bmatrix}}{\left\|\begin{bmatrix}4.8\\9.6\end{bmatrix}\right\|} = \begin{bmatrix}-51.5188\\-103.0377\end{bmatrix}$.

Constraint No (i)	$\mathbf{A}_{i:}$	$\mathbf{A}_{i:}^\top \mathbf{d}^1$
1	$[1,1]^\top$	-154.5565
2	$[4,-2]^\top$	0
3	$[1,0]^\top$	-51.5188
4	$[0,1]^\top$	-103.0377

Let $\mathbf{I}_{JB(\mathbf{d}^1)} = \{i \mid \mathbf{A}_{i:}^\top \mathbf{d}^1 < 0\} = \{1,3,4\}$.

Compute $\theta = \min\left\{ \dfrac{\mathbf{A}_{i:}^\top \mathbf{x}^1 - \mathbf{b}_i - r\|\mathbf{A}_{i:}\|}{-\mathbf{A}_{i:}^\top \mathbf{d}^1} \;\middle|\; i \in \mathbf{I}_{JB(\mathbf{d}^1)} \right\}$.

$= \min\{0.0032, 0.0130, 0.0080\} = 0.0032$.

Thus, $\mathbf{x}^2 = \mathbf{x}^1 + \theta\mathbf{d}^1 = \begin{bmatrix}0.6815\\0.8407\end{bmatrix} + 0.0032\begin{bmatrix}-51.5188\\-103.0377\end{bmatrix} = \begin{bmatrix}0.5166\\0.5109\end{bmatrix}$.

Iteration 3:

$\mathbf{I}_{J(\mathbf{x}^2)} = \left\{ i \;\middle|\; \dfrac{\mathbf{A}_{i:}^\top \mathbf{x}^2 - \mathbf{b}_i}{\|\mathbf{A}_{i:}\|} = r \right\} = \{1,2\}$.

Thus, the direction of the gravitational method is calculated as follows:

$$\begin{aligned} \text{Minimize} \quad & \left([12,6] - [\eta_1,\eta_2]\begin{bmatrix}1 & 1\\ 4 & -2\end{bmatrix}\right)\left([12,6] - [\eta_1,\eta_2]\begin{bmatrix}1 & 1\\ 4 & -2\end{bmatrix}\right)^\top \\ \text{subject to} \quad & \eta_1, \eta_2 \geq 0 \end{aligned} \tag{2.37}$$

For (2.37), solution $\begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}$ is $\begin{bmatrix} 8 \\ 1 \end{bmatrix}$.

Since $\xi^\top = \mathbf{c}^\top - [\eta_1, \eta_2]\mathbf{A}_{\mathbf{I}_{J(\mathbf{x}^2)}} = [12, 6] - [8, 1]\begin{bmatrix} 1 & 1 \\ 4 & -2 \end{bmatrix} = [0, 0]$, this means that the ball cannot move further in the gravitational direction and hence it stops. As a result, the solution of LP (2.34) by the gravitational method is $[x_1, x_2]^\top = [0.5166, 0.5109]^\top$. However, the radius of the ball which is defined in this example is not sufficiently small. Therefore, the solution from the gravitational method is the optimal that is $[x_1, x_2]^\top = [0.5, 0.5]^\top$ which it can be shown in Figure 2.3.

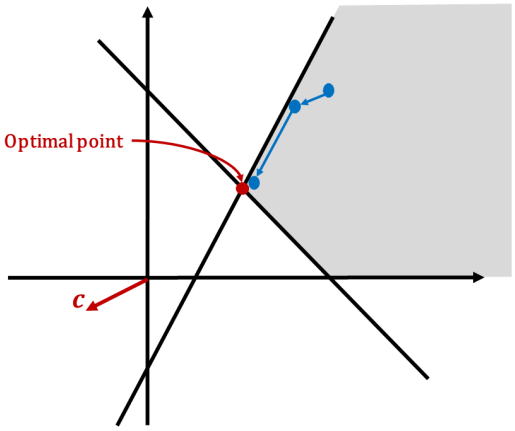


Figure 2.3: Example of the gravitational method.

2.6.4 The jump method

The idea of the jump method is inspired by the interior point method which uses the jump technique to improve the objective value of the existing extreme point. The jump method was first proposed by Yawila et al. [11] in 2016 called the simplex method with the objective jump which attempts to find the initial extreme point for the simplex method by using the objective jump. After the jump point is found then the artificial constraints are added to the LP model in order to create the artificial extreme point to perform the simplex method. However, adding the artificial constraints will expand the

size of the LP model. Therefore, the new jump method called the Preceding-jump simplex method is proposed by Kafakthong et al. [12] in 2019 which their method does not need to add artificial constraints into the LP model. Kafakthong's method attempts to find the direction for moving the jump point to the original extreme point by solving the right inverse. These two jump methods are explained in details as the following subsections.

2.6.4.1 Simplex method with objective jump

Consider a linear programming LP model as follows:

$$\begin{aligned}
\text{Maximize} \quad & c_1x_1 + c_2x_2 + ... + c_nx_n \\
\text{subject to} \quad & a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n && \leq b_1, \\
& a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n && \leq b_2, \\
& \vdots && \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n && \leq b_m, \\
& x_1, x_2, ..., x_n && \geq 0.
\end{aligned} \tag{2.38}$$

where $c_1, c_2, ..., c_n > 0$ and $b_1, b_2, ..., b_m \geq 0$. Let $Index = \{1, 2, ..., m\}$.

Step 1: Find the jump point.

Let $a_{ij}^{'} = \dfrac{a_{ij}}{c_j}$, for $i = 1, 2, ..., m$ and $j = 1, 2, ..., m$.

$$\mu^{'} = \min_{i \in Index} \left\{ \frac{b_i}{\sum_{j=1}^{n} a_{ij}^{'}} > 0 \;\middle|\; \sum_{j=1}^{n} a_{ij}^{'} > 0 \right\} \text{ and}$$

$$q = \operatorname*{argmin}_{i \in Index} \left\{ \frac{b_i}{\sum_{j=1}^{n} a_{ij}^{'}} > 0 \;\middle|\; \sum_{j=1}^{n} a_{ij}^{'} > 0 \right\}$$

The jump point is defined as $\mathbf{x}^J$ which it can be calculated as follows:

$x_i^J = \dfrac{\mu^{'}}{c_j}$ where x_i^J is the element of i^{th} of $\mathbf{x}^J$.

After that, q^{th} constraint is moved to the last constraint of the LP model, then indices of all constraints will be reordering from 1 to m.

Step 2: Generate the objective jump tableau as follows:

	s_m	s_{m+1}	$\cdots$	s_{m+n-1}	x_1	x_2	$\cdots$	x_n	s_1	$\cdots$	s_{m-1}	RHS
z	$\frac{1}{b_m}\sum_{j=1}^{n} c_j x_j^J$	$\sum_{j=1}^{n} c_j(\mathbf{J}^{-1})_{j2}$	$\cdots$	$\sum_{j=1}^{n} c_j(\mathbf{J}^{-1})_{jn}$	0	0	$\cdots$	0	0	$\cdots$	0	$\sum_{j=1}^{n} c_j x_j^J$
x_1	$\frac{x_1^J}{b_m}$	$(\mathbf{J}^{-1})_{12}$	$\cdots$	$(\mathbf{J}^{-1})_{1n}$	1	0	$\cdots$	0	0	$\cdots$	0	x_1^J
x_2	$\frac{x_2^J}{b_m}$	$(\mathbf{J}^{-1})_{22}$	$\cdots$	$(\mathbf{J}^{-1})_{2n}$	0	1	$\cdots$	0	0	$\cdots$	0	x_2^J
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_n	$\frac{x_n^J}{b_m}$	$(\mathbf{J}^{-1})_{n2}$	$\cdots$	$(\mathbf{J}^{-1})_{nn}$	0	0	$\cdots$	1	0	$\cdots$	0	x_n^J
s_1	$-\frac{1}{b_m}\sum_{j=1}^{n} a_{1j}x_j^J$	$-\sum_{k=1}^{n} a_{1k}(\mathbf{J}^{-1})_{k2}$	$\cdots$	$-\sum_{k=1}^{n} a_{1k}(\mathbf{J}^{-1})_{kn}$	0	0	$\cdots$	0	1	$\cdots$	0	$b_1 - \sum_{j=1}^{n} a_{1j}x_j^J$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
s_{m-1}	$-\frac{1}{b_m}\sum_{j=1}^{n} a_{m-1,j}x_j^J$	$-\sum_{k=1}^{n} a_{m-1,k}(\mathbf{J}^{-1})_{k2}$	$\cdots$	$-\sum_{k=1}^{n} a_{m-1,k}(\mathbf{J}^{-1})_{kn}$	0	0	$\cdots$	0	0	$\cdots$	1	$b_{m-1} - \sum_{j=1}^{n} a_{m-1,j}x_j^J$

where $\mathbf{J} = \begin{bmatrix} a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \\ c_1 & -c_2 & 0 & \cdots & 0 \\ c_1 & c_2 & -2c_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & c_3 & \cdots & -(n-1)c_n \end{bmatrix}_{n\times n}$

Step 3: Perform the simplex method by choosing the entering variable as s_{m+1}, $s_{m+2}, ..., s_{m+n-1}$, respectively. Note that the leaving variable is chosen by the minimum ratio test.

Step 4: Eliminate the rows and columns of the set of s_{m+1}, $s_{m+2}, ..., s_{m+n-1}$.

Step 5: Perform the standard simplex method to find the solution of the LP model.

Example 2.8. Consider a linear programming as follows:

$$\begin{aligned} \text{Maximize} \quad & x_1 + x_2 \\ \text{subject to} \quad & 3x_1 + x_2 \leq 18, \qquad (1) \\ & -3x_1 + 2x_2 \leq 6, \qquad (2) \\ & 3x_1 - 2x_2 \leq 8, \qquad (3) \\ & -5x_1 + x_2 \leq 2, \qquad (4) \\ & x_1, x_2 \geq 0. \end{aligned} \tag{2.39}$$

Step 1: Find the jump point.

Let $Index = \{1, 2, 3, 4\}$. Since $c_1 = c_2 = 1$, then $a_{ij}^{'} = a_{ij}$ for $i \in \{1, 2, 3, 4\}$ and $j \in \{1, 2\}$.

Let $\mu^{'} = \min\limits_{i \in Index} \left\{ \dfrac{b_i}{\sum_{j=1}^{n} a_{ij}^{'}} > 0 \;\middle|\; \sum_{j=1}^{n} a_{ij}^{'} > 0 \right\} = \min\limits_{\{1,3\}} \left\{ \dfrac{18}{4}, \dfrac{8}{1} \right\} = 4.5$ and $q = 1$.

Thus, the jump point is $[x_1^J, x_2^J]^\top = [4.5, 4.5]^\top$.

Rearrange the index of the constraints as follows:

$$
\begin{aligned}
\text{Maximize} \quad & x_1 + x_2 & & \\
\text{subject to} \quad & -3x_1 + 2x_2 & \leq 6, \quad & (1) \\
& 3x_1 - 2x_2 & \leq 8, \quad & (2) \\
& -5x_1 + x_2 & \leq 2, \quad & (3) \\
& 3x_1 + x_2 & \leq 18, \quad & (4) \\
& x_1, x_2 & \geq 0. &
\end{aligned}
\tag{2.40}
$$

Step 2: Generate the objective jump tableau.

Since $J = \begin{bmatrix} 3 & 1 \\ 1 & -1 \end{bmatrix}$, then $J^{-1} = \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & -0.75 \end{bmatrix}$.

Thus, the objective jump tableau is created as follows:

	s_4	s_5	x_1	x_2	s_1	s_2	s_3	RHS
z	1/2	-1/2	0	0	0	0	0	9
x_1	1/4	1/4	1	0	0	0	0	9/2
x_2	1/4	-3/4	0	1	0	0	0	9/2
s_1	1/4	9/4	0	0	1	0	0	21/2
s_2	-1/4	-9/4	0	0	0	1	0	7/2
s_3	1	2	0	0	0	0	1	20

Step 3: Perform the simplex method by choosing the entering variable as s_5.

	s_4	s_5	x_1	x_2	s_1	s_2	s_3	RHS
z	5/9	0	0	0	2/9	0	0	34/3
x_1	2/9	0	1	0	-1/9	0	0	10/3
x_2	1/3	0	0	1	1/3	0	0	8
s_5	1/9	1	0	0	4/9	0	0	14/3
s_2	0	0	0	0	1	1	0	14
s_3	7/9	0	0	0	-8/9	0	1	32/3

Step 4: Eliminate the rows and columns of s_5.

	s_4	x_1	x_2	s_1	s_2	s_3	RHS
z	5/9	0	0	2/9	0	0	34/3
x_1	2/9	1	0	-1/9	0	0	10/3
x_2	1/3	0	1	1/3	0	0	8
s_2	0	0	0	1	1	0	14
s_3	7/9	0	0	-8/9	0	1	32/3

Since this tableau is the optimal tableau. Therefore, the optimal solution is found with $[x_1, x_2]^\top = [10/3, 8]^\top$.

2.6.4.2 Preceding-jump simplex method

Consider a linear programming LP model as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.41}$$

where $\mathbf{c} \in \mathbf{R}^n$ with a positive vector, $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{A} \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$ is a non-positive vector.

The initial jump phase

Step 1: Define the initial feasible point and the set of binding constraints.

Let $\mathbf{x}_0$ be a feasible point for LP (2.41).

Note $\mathbf{x}_0 = \mathbf{0} \in \mathbf{R}^n$ is defined as an initial feasible point for LP (2.41).

Let FB be the set of the first binding constraints.

Since $\mathbf{x}_0$ binds non-negative constraints of $\mathbf{x}$. Therefore, the first element of FB is $\{m+1, m+2, ..., m+n\}$. Moreover, if there is some constraints which $\mathbf{x}_0$ is

binding $\mathbf{A}_{i:}^{T}\mathbf{x}_0 = b_i$ then the index of that constraint is added to FB.

Step 2: Find the initial jump point.

The gradient vector of the objective function $\mathbf{c}$ is defined as direction $\mathbf{d}_0$ for moving $\mathbf{x}_0$ to the initial jump point. Let $\mathbf{x}_1$ be the initial jump point and γ_0 be the index of the constraint which $\mathbf{x}_1$ is binding.

Thus, $\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0\mathbf{d}_0$ where $\alpha_0 = \min\limits_{i\in\{1,2,\ldots,m+n\}\setminus FB}\left\{\dfrac{b_i}{\mathbf{A}_{i:}^{\top}\mathbf{d}_0} \;\middle|\; b_i \geq 0 \text{ and } \mathbf{A}_{i:}^{\top}\mathbf{d}_0 > \mathbf{0}\right\}$

and $\gamma_0 = \operatorname{argmin}\limits_{i\in\{1,2,\ldots,m+n\}\setminus FB}\left\{\dfrac{b_i}{\mathbf{A}_{i:}^{\top}\mathbf{d}_0} \;\middle|\; b_i \geq 0 \text{ and } \mathbf{A}_{i:}^{\top}\mathbf{d}_0 > \mathbf{0}\right\}$.

The jump-to-vertex phase

Step 3: Let $\mathbf{x}_k$ be the current feasible jump point and V be the set of all visited constraints.

Step 4: Solve $\mathbf{x}_k^{'} = \mathbf{A}_{i:}^{+}b_i$ for $i \in V$ where $\mathbf{A}_{i:}^{+}$ is the right inverse of $\mathbf{A}_{i:}$.

Step 5: Create direction $\mathbf{d}_k$.

If $\mathbf{c}^{\top}(\mathbf{x}_k^{'} - \mathbf{x}_k) > 0$, $\mathbf{d}_k = \mathbf{x}_k^{'} - \mathbf{x}_k$. Otherwise, $\mathbf{d}_k = \mathbf{x}_k - \mathbf{x}_k^{'}$.

Step 6: Compute $\alpha_k = \min\limits_{i\in\{1,2,\ldots,m+n\}\setminus V}\left\{\dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}_k}{\mathbf{A}_{i:}^{\top}\mathbf{d}_k} \;\middle|\; b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}_k \geq \mathbf{0} \text{ and } \mathbf{A}_{i:}^{\top}\mathbf{d}_k > \mathbf{0}\right\}$.

Step 7: Find $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k\mathbf{d}_k$ and

$\gamma_k = \operatorname{argmin}\limits_{i\in\{1,2,\ldots,m+n\}\setminus V}\left\{\dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}_k}{\mathbf{A}_{i:}^{\top}\mathbf{d}_k} \;\middle|\; b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}_k \geq \mathbf{0} \text{ and } \mathbf{A}_{i:}^{\top}\mathbf{d}_k > \mathbf{0}\right\}$.

Step 8: Repeat Step 3 - Step 7 until the number of elements of V is equal to the number of variables.

After the jump-to-vertex phase terminates then the standard simplex method is performed to find the solution of the LP model by starting with the last point obtained from the jump-to-vertex phase.

Example 2.9. Consider a linear programming as follows:

$$\begin{array}{lll} \text{Maximize} & x_1 + x_2 & \\ \text{subject to} & 3x_1 + x_2 \leq 18, & (1) \\ & -3x_1 + 2x_2 \leq 6, & (2) \\ & 3x_1 - 2x_2 \leq 8, & (3) \\ & -5x_1 + x_2 \leq 2, & (4) \\ & x_1, x_2 \geq 0. & \end{array} \tag{2.42}$$

Transform LP (2.42) in the following form.

$$\begin{array}{lll} \text{Maximize} & x_1 + x_2 & \\ \text{subject to} & 3x_1 + x_2 \leq 18, & (1) \\ & -3x_1 + 2x_2 \leq 6, & (2) \\ & 3x_1 - 2x_2 \leq 8, & (3) \\ & -5x_1 + x_2 \leq 2, & (4) \\ & -x_1 \leq 0, & (5) \\ & -x_2 \leq 0. & (6) \end{array} \tag{2.43}$$

The initial jump phase

Step 1: Define the starting point, the starting direction and the set of the binding constraints.

Let $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{d}_0 = \mathbf{c} = [1, 1]^\top$. Hence, $FB = \{5, 6\}$.

Step 2: Find the initial jump point.

Constraint No (i)	$\mathbf{A}_{i:}$	$\mathbf{A}_{i:}^{\top}\mathbf{d}_0$
1	$[3,1]^{\top}$	4
2	$[-3,2]^{\top}$	-1
3	$[3,-2]^{\top}$	1
4	$[-5,1]^{\top}$	-4
5	$[-1,0]^{\top}$	-1
6	$[0,-1]^{\top}$	-1

Thus, α_0 is computed as follows:

$$\alpha_0 = \min_{i\in\{1,2,\ldots,m+n\}\setminus FB}\left\{\frac{b_i}{\mathbf{A}_{i:}^{\top}\mathbf{d}_0}\;\middle|\; b_i \geq 0 \text{ and } \mathbf{A}_{i:}^{\top}\mathbf{d}_0 > \mathbf{0}\right\} = \min_{\{1,3\}}\left\{\frac{18}{4}, \frac{8}{1}\right\} = 4.5.$$

Therefore, $\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0\mathbf{d}_0 = \begin{bmatrix}0\\0\end{bmatrix} + 4.5\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}4.5\\4.5\end{bmatrix}$ and $\gamma_0 = 1$.

The jump-to-vertex phase

Step 3: Let $\mathbf{x}_1$ be the current feasible jump point and V be the set of all visited constraints. Therefore, $V = \{1\}$.

Step 4: Solve $\mathbf{x}_1^{'} = \mathbf{A}_{1:}^{+}b_1 = [3,1]^{+}[18] = [5.4, 1.8]^{\top}$.

Step 5: Compute $\mathbf{x}_1^{'} - \mathbf{x}_1 = \begin{bmatrix}5.4\\1.8\end{bmatrix} - \begin{bmatrix}4.5\\4.5\end{bmatrix} = \begin{bmatrix}0.9\\-2.7\end{bmatrix}$.

Since $\mathbf{c}^{\top}(\mathbf{x}_1^{'} - \mathbf{x}_1) = [1,1]^{\top}\begin{bmatrix}0.9\\-2.7\end{bmatrix} = -1.8 < 0$. Thus, $\mathbf{d}_1 = \begin{bmatrix}-0.9\\2.7\end{bmatrix}$.

Step 6: Compute $\alpha_1 = \min_{i\in\{1,2,\ldots,6\}\setminus V}\left\{\frac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}_1}{\mathbf{A}_{i:}^{\top}\mathbf{d}_1}\;\middle|\; b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}_1 \geq \mathbf{0} \text{ and } \mathbf{A}_{i:}^{\top}\mathbf{d}_1 > \mathbf{0}\right\}$.

Constraint No (i)	$\mathbf{A}_{i:}$	b_i	$\mathbf{A}_{i:}^\top \mathbf{d}_1$	$b_i - \mathbf{A}_{i:}^\top \mathbf{x}_1$
1	$[3,1]^\top$	18	0	0
2	$[-3,2]^\top$	6	8.1	10.5
3	$[3,-2]^\top$	8	-8.1	3.5
4	$[-5,1]^\top$	2	7.2	20
5	$[-1,0]^\top$	0	0.9	4.5
6	$[0,-1]^\top$	0	-2.7	4.5

Thus, $\alpha_1 = \min\limits_{\{2,4,5\}} \left\{ \frac{10.5}{8.1}, \frac{20}{7.2}, \frac{4.5}{0.9} \right\} = 1.2962.$

Step 7: Therefore, $\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{d}_1 = \begin{bmatrix} 4.5 \\ 4.5 \end{bmatrix} + 1.2962 \begin{bmatrix} -0.9 \\ 2.7 \end{bmatrix} = \begin{bmatrix} 3.3334 \\ 7.9997 \end{bmatrix}$ and $\gamma_1 = 2$. Thus, $V = \{1, 2\}$.

Step 8: Since the number of the elements of V are equal to the number of variables. Thus, the extreme point of LP (2.42) is found with $\mathbf{x}_2 = \begin{bmatrix} 4.8888 \\ 3.3336 \end{bmatrix}$.

Next, the simplex tableau with the basic variables x_1, x_2, s_3, s_4 and the non-basic variables s_1, s_2 is created as follows:

	x_1	x_2	s_1	s_2	s_3	s_4	RHS
z	0	0	5/9	2/9	0	0	34/3
x_1	1	0	2/9	-1/9	0	0	10/3
x_2	0	1	1/3	1/3	0	0	8
s_3	0	0	0	1	1	0	14
s_4	0	0	7/9	-8/9	0	1	32/3

Since this tableau is the optimal tableau. Therefore, the optimal solution is found with $[x_1, x_2]^\top = [10/3, 8]^\top$.

2.7 Artificial-free simplex algorithm based on the non-acute constraint relaxation

The new method for solving an LP model without artificial variable is proposed by Boonperm and Sinapiromsaran called the artificial-free simplex algorithm based on the non-acute constraint relaxation (SNAR) [4]. The relation of the angle between the gradient vector of the objective function and the gradient vector of each constraint is applied to create the LP relaxation in order to reduce the number of constraints in calculating.

Consider a linear programming model as follows:

$$\begin{aligned} &\text{Minimize} \quad \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}, \end{aligned} \tag{2.44}$$

where $\mathbf{c} \in \mathbf{R}^n$ is a nonzero vector, $\mathbf{x}$ is the vector of n, $\mathbf{A}$ is the matrix of size $m \times n$ and b is the vector of m. Let $Index = \{1, 2, ..., m\}$.

Step 1: Split constraints into two groups.

$\mathbf{I}_{Pos} = \{i \in Index \mid \mathbf{A}_{i:}^\top \mathbf{c} > 0\}$, and $\mathbf{I}_{Neg} = \{i \in Index \mid \mathbf{A}_{i:}^\top \mathbf{c} \leq 0\}$,

which $\mathbf{I}_{Pos}$ is the set of all acute constraints and $\mathbf{I}_{Neg}$ is the set of all non-acute constraints.

Step 2: Construct the LP relaxation.

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}_{\mathbf{I}_{Pos}} \mathbf{x} \leq \mathbf{b}_{\mathbf{I}_{Pos}}, \end{aligned} \tag{2.45}$$

where $\mathbf{A}_{\mathbf{I}_{Pos}}$ is the submatrix of $\mathbf{A}$ that the row indices from $\mathbf{I}_{Pos}$ and $\mathbf{b}_{\mathbf{I}_{Pos}}$ is the column vector of $\mathbf{b}$ that corresponding to $\mathbf{I}_{Pos}$.

Step 3: Find the solution of the LP relaxation.

- If $\mathbf{b}_{\mathbf{I}_{Pos}} \geq \mathbf{0}$, the simplex method is applied to find the solution of the LP relaxation. Since, LP (2.45) is the unrestricted variable LP model. Therefore, the unrestricted variable ($\mathbf{x}$) is defined as the subtraction of two new restricted variables: $\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^-$ where $\mathbf{x}^+, \mathbf{x}^- \geq \mathbf{0}$. Thus, LP model (2.45) is transformed to

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^\top \mathbf{x}^+ - \mathbf{c}^\top \mathbf{x}^- \\ \text{subject to} \quad & \mathbf{A}_{\mathbf{I}_{Pos}} \mathbf{x}^+ - \mathbf{A}_{\mathbf{I}_{Pos}} \mathbf{x}^- \leq \mathbf{b}_{\mathbf{I}_{Pos}}, \\ & \mathbf{x}^+, \mathbf{x}^- \geq \mathbf{0}. \end{aligned} \tag{2.46}$$

After that, LP (2.46) is transformed to the standard form by adding the slack variable ($\mathbf{s}$).

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^\top \mathbf{x}^+ - \mathbf{c}^\top \mathbf{x}^- \\ \text{subject to} \quad & \mathbf{A}_{\mathbf{I}_{Pos}} \mathbf{x}^+ - \mathbf{A}_{\mathbf{I}_{Pos}} \mathbf{x}^- + \mathbf{s} = \mathbf{b}_{\mathbf{I}_{Pos}}, \\ & \mathbf{x}^+, \mathbf{x}^-, \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{2.47}$$

- If $\mathbf{b}_{\mathbf{I}_{Pos}} \not\geq \mathbf{0}$, the starting point ($\mathbf{x}'$) is calculated from $\mathbf{x}' = -\lambda \mathbf{c}$ where $\lambda = \max_{i \in \mathbf{I}_{Pos^-}} \left\{ \frac{b_i}{-\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$ and $\mathbf{I}_{Pos^-} = \{ i \in \mathbf{I}_{Pos} \mid b_i < 0 \}$. After that, the starting point $\mathbf{x}'$ is defined as the origin point of the transformed LP model as follows:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^\top \tilde{\mathbf{x}} + \mathbf{c}^\top \mathbf{x}' \\ \text{subject to} \quad & \mathbf{A}_{\mathbf{I}_{Pos}} \tilde{\mathbf{x}} \leq \mathbf{b}_{\mathbf{I}_{Pos}} - \mathbf{A}_{\mathbf{I}_{Pos}} \mathbf{x}', \end{aligned} \tag{2.48}$$

where $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}'$. Since LP (2.48) is the unrestricted variable LP model. Thus, $\tilde{\mathbf{x}}$ is defined as $\tilde{\mathbf{x}} = \mathbf{x}^+ - \mathbf{x}^-$ where $\mathbf{x}^+, \mathbf{x}^- \geq \mathbf{0}$. Therefore, LP (2.48) is written as

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^\top \mathbf{x}^+ - \mathbf{c}^\top \mathbf{x}^- + \mathbf{c}^\top \mathbf{x}' \\ \text{subject to} \quad & \mathbf{A}_{\mathbf{I}_{Pos}} \mathbf{x}^+ - \mathbf{A}_{\mathbf{I}_{Pos}} \mathbf{x}^- \leq \mathbf{b}_{\mathbf{I}_{Pos}} - \mathbf{A}_{\mathbf{I}_{Pos}} \mathbf{x}', \\ & \mathbf{x}^+, \mathbf{x}^- \geq \mathbf{0}. \end{aligned} \tag{2.49}$$

Then the slack variable ($\mathbf{s}$) is added to LP (2.49) to change it to the standard form. After that, the simplex method is performed.

Step 4: Reinsert the non-acute constraints.

If the solution of LP (2.45) is optimal, then all non-acute constraints from $\mathbf{I}_{Neg}$ will be added. Otherwise, a single non-acute constraint is added to the LP relaxation one by one. If the primal of the LP relaxation is feasible, SNAR applies the simplex method to find the solution. On the other hand, if the primal LP relaxation is infeasible, the technique from Pan [2] is applied to change dual infeasible to dual feasible. After that, the dual simplex method is performed.

Example 2.10. Consider a linear programming in the following form.

$$\begin{aligned} \text{Maximize} \quad & x_1 + x_2 \\ \text{subject to} \quad & 3x_1 + x_2 \leq 18, \\ & -3x_1 + 2x_2 \leq 6, \\ & 3x_1 - 2x_2 \leq 8, \\ & -5x_1 + x_2 \leq 2, \\ & x_1 \geq 0, \\ & x_2 \geq 0. \end{aligned} \tag{2.50}$$

Whereupon, LP (2.50) is transformed as follows:

$$
\begin{aligned}
\text{Maximize} \quad & x_1 + x_2 & & \\
\text{subject to} \quad & 3x_1 + x_2 & \leq 18, & \quad (1) \\
& -3x_1 + 2x_2 & \leq 6, & \quad (2) \\
& 3x_1 - 2x_2 & \leq 8, & \quad (3) \\
& -5x_1 + x_2 & \leq 2, & \quad (4) \\
& -x_1 & \leq 0, & \quad (5) \\
& -x_2 & \leq 0. & \quad (6)
\end{aligned}
\tag{2.51}
$$

Step 1: Separate the constraints into two groups:

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

Thus, $\mathbf{I}_{Pos} = \{1,3\}$ and $\mathbf{I}_{Neg} = \{2,4,5,6\}$.

Step 2: Create the LP relaxation with only acute constraints.

The relaxation LP model is

$$\begin{array}{lll} \text{Maximize} & x_1 + x_2 & \\ \text{subject to} & 3x_1 + x_2 \leq 18, & (1) \\ & 3x_1 - 2x_2 \leq 8. & (3) \end{array} \tag{2.52}$$

Step 3: Find the solution of the LP relaxation.

Since $\mathbf{b}_{\mathbf{I}_{Pos}} = [18, 8]^\top \geq \mathbf{0}$. Therefore, $\mathbf{x}^{(0)} = \mathbf{0}$ is a feasible point. Since LP (2.52) is an unrestricted variable model, x_1 is denoted by $x_1^+ - x_1^-$ where $x_1^+, x_1^- \geq 0$ and x_2 is denoted by $x_2^+ - x_2^-$ where $x_2^+, x_2^- \geq 0$. Thus, the LP relaxation is transformed as follows:

$$\begin{array}{lll} \text{Maximize} & x_1^+ - x_1^- + x_2^+ - x_2^- & \\ \text{subject to} & 3x_1^+ - 3x_1^- + x_2^+ - x_2^- \leq 18, & (1) \\ & 3x_1^+ - 3x_1^- - 2x_2^+ + 2x_2^- \leq 8, & (3) \\ & x_1^+, x_1^-, x_2^+, x_2^- \geq 0. & \end{array} \tag{2.53}$$

After that, s_1 and s_2 are added to LP (2.53) in order to make it to the standard form.

$$\begin{array}{lll} \text{Maximize} & x_1^+ - x_1^- + x_2^+ - x_2^- & \\ \text{subject to} & 3x_1^+ - 3x_1^- + x_2^+ - x_2^- + s_1 = 18, & (1) \\ & 3x_1^+ - 3x_1^- - 2x_2^+ + 2x_2^- + s_2 = 8, & (3) \\ & x_1^+, x_1^-, x_2^+, x_2^-, s_1, s_2 \geq 0. & \end{array} \tag{2.54}$$

The simplex method is performed with the initial tableau.

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	RHS
z	-1	1	-1	1	0	0	0
s_1	3	-3	1	-1	1	0	18
s_2	**3**	-3	-2	2	0	1	8

By Dantzig's pivot rule, x_1^+ enters the basis and s_2 leaves the basis. After pivoting, the second tableau is updated as follows:

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	RHS
z	0	-2	-5/3	5/3	0	1/3	8/3
s_1	0	0	**3**	-3	1	-1	10
x_1^+	1	-1	-2/3	2/3	0	1/3	8/3

The entering variable is x_2^+ and the leaving variable is s_1.

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	RHS
z	0	-2	0	0	5/9	- 2/9	74/9
x_2^+	0	0	1	-1	1/3	-1/3	10/3
x_1^+	1	-1	0	0	2/9	**1/9**	44/9

The entering variable is s_2 and the leaving variable is x_1^+.

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	RHS
z	2	-4	0	0	1	0	18
x_2^+	3	-3	1	-1	1	0	18
s_2	9	-9	0	0	2	1	44

After pivoting, the solution of this tableau is an unbounded solution. Thus, a single non-acute constraint is reinserted one by one to the tableau until the optimal solution is found.

Step 4: Reinsert the non-acute constraints.

The first constraint from $\mathbf{I}_{Neg}$ is added into the tableau as follows:

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	RHS
z	-2	2	0	0	-1	0	0	-18
x_2^+	3	-3	1	-1	1	0	0	18
s_2	9	-9	0	0	2	1	0	44
s_3	-9	**9**	0	0	-2	0	1	-30

For this tableau, x_1^- enters the basis and s_3 leaves the basis. After pivoting, the tableau is shown as

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	RHS
z	0	0	0	0	-5/9	0	-2/9	-34/3
x_2^+	0	0	1	-1	1/3	0	1/3	8
s_2	0	0	0	0	0	1	1	14
x_1^-	-1	1	0	0	-2/9	0	1/9	-10/3

Since the solution to this tableau is the optimal solution. Thus, the rest non-acute constraints are reinserted to the tableau.

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	0	0	0	0	-5/9	0	-2/9	0	0	0	-34/3
x_2^+	0	0	1	-1	1/3	0	1/3	0	0	0	8
s_2	0	0	0	0	0	1	1	0	0	0	14
x_1^-	-1	1	0	0	**-2/9**	0	1/9	0	0	0	-10/3
s_4	0	0	0	0	7/9	0	-8/9	1	0	0	32/3
s_5	0	0	0	0	2/9	0	-1/9	0	1	0	10/3
s_6	0	0	0	0	1/3	0	1/3	0	0	1	8

For this tableau, the primal model is infeasible and the dual model is feasible. Thus, the dual simplex is performed. The leaving variable is x_1^- and the entering variable is s_1.

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	0	0	0	0	-5/9	0	-2/9	0	0	0	-34/3
x_2^+	0	0	1	-1	1/3	0	1/3	0	0	0	8
s_2	0	0	0	0	0	1	1	0	0	0	14
x_1^+	1	-1	0	0	2/9	0	-1/9	0	0	0	10/3
s_4	0	0	0	0	7/9	0	-8/9	1	0	0	32/3
s_5	0	0	0	0	2/9	0	-1/9	0	1	0	10/3
s_6	0	0	0	0	1/3	0	1/3	0	0	1	8

Since this simplex tableau is the optimal tableau, the solution of this LP model is -34/3 with $[x_1, x_2]^\top = [10/3, 8]^\top$.

CHAPTER III

THE NEW TECHNIQUE FOR SOLVING THE UNRESTRICTED VARIABLE MODEL

An unrestricted variable LP model is a linear programming model that has unrestricted decision variables. The traditional method usually solves the unrestricted variable LP model by transforming into the standard form. However, for converting the unrestricted variable LP model to the standard form, the unrestricted variable (x) must be written as the subtraction of two new restricted variables as $x = x^+ - x^-$ where $x^+ \geq 0$ and $x^- \geq 0$. Therefore, the number of variables is increased twice for each unrestricted variable.

Let $\mathbf{I_R}$ be a set of the indices of restricted decision variables and $\mathbf{I_U}$ be a set of the indices of unrestricted decision variables.

Consider an unrestricted variable LP model in the form

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}_{\mathbf{I_R}}^{\top}\mathbf{x}_{\mathbf{I_R}} + \mathbf{c}_{\mathbf{I_U}}^{\top}\mathbf{x}_{\mathbf{I_U}} \\ \text{subject to} \quad & \mathbf{A}_{\mathbf{I_R}}\mathbf{x}_{\mathbf{I_R}} + \mathbf{A}_{\mathbf{I_U}}\mathbf{x}_{\mathbf{I_U}} = \mathbf{b}, \\ & \mathbf{x}_{\mathbf{I_R}} \geq \mathbf{0}, \mathbf{x}_{\mathbf{I_U}} \text{ unrestricted variable.} \end{aligned} \tag{3.1}$$

Unrestricted variable LP (3.1) is transformed into the standard form as follows: Let $\mathbf{x}_{\mathbf{I_U}} = \mathbf{x}_{\mathbf{I_U}}^{+} - \mathbf{x}_{\mathbf{I_U}}^{-}$ where $\mathbf{x}_{\mathbf{I_U}}^{+}, \mathbf{x}_{\mathbf{I_U}}^{-} \geq \mathbf{0}$.

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}_{\mathbf{I_R}}^{\top}\mathbf{x}_{\mathbf{I_R}} + \mathbf{c}_{\mathbf{I_U}}^{\top}\mathbf{x}_{\mathbf{I_U}}^{+} - \mathbf{c}_{\mathbf{I_U}}^{\top}\mathbf{x}_{\mathbf{I_U}}^{-} \\ \text{subject to} \quad & \mathbf{A}_{\mathbf{I_R}}\mathbf{x}_{\mathbf{I_R}} + \mathbf{A}_{\mathbf{I_U}}\mathbf{x}_{\mathbf{I_U}}^{+} - \mathbf{A}_{\mathbf{I_U}}\mathbf{x}_{\mathbf{I_U}}^{-} = \mathbf{b}, \\ & \mathbf{x}_{\mathbf{I_R}}, \mathbf{x}_{\mathbf{I_U}}^{+}, \mathbf{x}_{\mathbf{I_U}}^{-} \geq \mathbf{0}. \end{aligned} \tag{3.2}$$

After that, LP (3.2) in the simplex tableau must have each column of the unre-

stricted variable represented by two restricted variables where their signs are negative of each other. Consider $\mathbf{x}_j$, the column of the coefficient matrix is divided into two columns as positive $\mathbf{x}_j^+$ and negative $\mathbf{x}_j^-$.

$\mathbf{x}_j^+$	$\mathbf{x}_j^-$
$\mathbf{c}_j$	$-\mathbf{c}_j$
$\mathbf{A}_{:j}$	$-\mathbf{A}_{:j}$

Thus, the reduced cost of $\mathbf{x}_j^+$ and $\mathbf{x}_j^-$ are $\mathbf{c}_\mathbf{B}^\top B^{-1}\mathbf{A}_{:j}-\mathbf{c}_j = z_j-c_j$ and $\mathbf{c}_\mathbf{B}^\top B^{-1}(-\mathbf{A}_{:j})+\mathbf{c}_j = -z_j + c_j$, respectively. If $z_j - c_j$ is a negative value, $\mathbf{x}_j^+$ can be the entering variable. While if $z_j - c_j$ is a positive value, $\mathbf{x}_j^-$ can be the entering variable. Therefore, the entering variable of the new technique for solving the LP model can be chosen from the maximum of the absolute value of the reduced cost of the unrestricted variable and the negative value of the reduced cost of the restricted variable.

Let $f(x_i) = |\mathbf{c}_\mathbf{B}^\top\mathbf{B}^{-1}\mathbf{A}_{:i} - c_i|$ for $x_i \in \mathbf{I_U} \cup \mathbf{I}_\mathbf{R}^{'}$,

where $\mathbf{I}_\mathbf{R}^{'} = \left\{i \in \mathbf{I_R} \mid \mathbf{c}_\mathbf{B}^\top\mathbf{B}^{-1}\mathbf{A}_{:i} - c_i \leq 0\right\}$. If $\max\limits_{x_i \in \mathbf{I_U}\cup\mathbf{I}_\mathbf{R}^{'}} \{f(x_i)\} > 0$, the entering variable is x_r where $r = \operatorname{argmax}\limits_{x_i \in \mathbf{I_U}\cup\mathbf{I}_\mathbf{R}^{'}} \{f(x_i)\}$. Otherwise, the new technique of unrestricted variable LP model stops with the optimal solution.

After the entering variable is found, the leaving variable is calculated depending on the type of variables that it is an unrestricted variable or a restricted variable. Let $\mathbf{y}_r = \mathbf{B}^{-1}\mathbf{A}_{:r}$.

If $x_r \in \mathbf{I_R}$, the leaving variable is $x_{\mathbf{B}_\eta}$ where

$$\eta = \operatorname{argmin}\left\{ \frac{\bar{b}_i}{y_{r_i}} \;\middle|\; \bar{b}_i \geq 0 \text{ and } y_{r_i} > 0 \right\},$$

where y_{r_i} is the i^{th} element of $\mathbf{y}_r$.

If $x_r \in \mathbf{I_U}$ with the positive reduced cost, the leaving variable is $x_{\mathbf{B}_\eta}$ where

$$\eta = \text{argmin}\left\{ \frac{\bar{b}_i}{|y_{r_i}|} \;\middle|\; \bar{b}_i \geq 0 \text{ and } y_{r_i} > 0 \right\},$$

where y_{r_i} is the i^{th} element of $\mathbf{y}_r$.

If $x_r \in \mathbf{I_U}$ with the negative reduced cost, the leaving variable is $x_{\mathbf{B}_\eta}$ where

$$\eta = \text{argmin}\left\{ \frac{\bar{b}_i}{|y_{r_i}|} \;\middle|\; \bar{b}_i \geq 0 \text{ and } y_{r_i} < 0 \right\},$$

where y_{r_i} is the i^{th} element of $\mathbf{y}_r$.

After the entering variable and the leaving variable are found, the simplex tableau is updated which is slightly different from the traditionally simplex tableau update. The simplex tableau before the update can be displayed as follows:

	$x_{\mathbf{B}_1}$	$\cdots$	$x_{\mathbf{B}_\eta}$	$\cdots$	$x_{\mathbf{B}_m}$	$\cdots$	x_j	$\cdots$	x_r	$\cdots$	RHS
z	0	$\cdots$	0	$\cdots$	0	$\cdots$	$-(c_j - z_j)$	$\cdots$	$-(c_r - z_r)$	$\cdots$	$c_B^\top \bar{b}$
$x_{\mathbf{B}_1}$	1	$\cdots$	0	$\cdots$	0	$\cdots$	y_{j_1}	$\cdots$	y_{r_1}	$\cdots$	$\bar{b}_1$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{\mathbf{B}_\eta}$	0	$\cdots$	1	$\cdots$	0	$\cdots$	y_{j_η}	$\cdots$	y_{r_η}	$\cdots$	$\bar{b}_\eta$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{\mathbf{B}_m}$	0	$\cdots$	0	$\cdots$	1	$\cdots$	y_{j_m}	$\cdots$	y_{r_m}	$\cdots$	$\bar{b}_m$

Let x_r be the entering variable and $x_{\mathbf{B}_\eta}$ be the leaving variable. Therefore, the unrestricted simplex tableau is updated by the new technique which it can proceed as follows:

	$x_{\mathbf{B}_1}$	$\cdots$	$x_{\mathbf{B}_\eta}$	$\cdots$	$x_{\mathbf{B}_m}$	$\cdots$	x_j	$\cdots$	x_r	$\cdots$	RHS
z	0	$\cdots$	$\frac{(c_r - z_r)}{y_{r_\eta}}$	$\cdots$	0	$\cdots$	$-(c_j - z_j) + \frac{(c_r - z_r)y_{j_\eta}}{y_{r_\eta}}$	$\cdots$	0	$\cdots$	$c_B^\top \bar{b} + \frac{(c_r - z_r)\bar{b}_\eta}{y_{r_\eta}}$
$x_{\mathbf{B}_1}$	1	$\cdots$	$-\frac{y_{r_1}}{y_{r_\eta}}$	$\cdots$	0	$\cdots$	$y_{j_1} - \frac{y_{r_1}y_{j_\eta}}{y_{r_\eta}}$	$\cdots$	0	$\cdots$	$\bar{b}_1 - \frac{y_{r_1}\bar{b}_\eta}{y_{r_\eta}}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{\mathbf{B}_r}$	0	$\cdots$	$\frac{1}{\lvert y_{r_\eta}\rvert}$	$\cdots$	0	$\cdots$	$\frac{y_{j_\eta}}{\lvert y_{r_\eta}\rvert}$	$\cdots$	1	$\cdots$	$\frac{\bar{b}_\eta}{\lvert y_{r_\eta}\rvert}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$
$x_{\mathbf{B}_m}$	0	$\cdots$	$\frac{-y_{r_m}}{y_{r_\eta}}$	$\cdots$	1	$\cdots$	$y_{j_m} - \frac{y_{r_m}y_{j_\eta}}{y_{r_\eta}}$	$\cdots$	0	$\cdots$	$\bar{b}_m - \frac{y_{r_m}\bar{b}_\eta}{y_{r_\eta}}$

Example 3.1. Consider an unrestricted variable linear programming model:

$$\begin{aligned} \text{Maximize} \quad & -x_1 + 2x_2 + 3x_3 \\ \text{subject to} \quad & x_1 + 3x_2 + 2x_3 \leq 18, \\ & -2x_1 - 3x_2 \leq 6, \\ & x_1 - 5x_2 + x_3 \leq 15, \\ & -x_1 + 3x_2 \leq 6, \\ & x_3 \geq 0. \end{aligned} \tag{3.3}$$

LP (3.3) is transformed to the standard form by adding the slack variables as follows:

$$\begin{aligned} \text{Maximize} \quad & -x_1 + 2x_2 + 3x_3 \\ \text{subject to} \quad & x_1 + 3x_2 + 2x_3 + s_1 = 18, \\ & -2x_1 - 3x_2 + s_2 = 6, \\ & x_1 - 5x_2 + x_3 + s_3 = 15, \\ & -x_1 + 3x_2 + s_4 = 6, \\ & x_3, s_1, s_2, s_3, s_4 \geq 0. \end{aligned} \tag{3.4}$$

For this LP model, x_1, x_2 are unrestricted variables and x_3, s_1, s_2, s_3, s_4 are re-

stricted variables. The initial simplex tableau can be constructed as follows:

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	1	-2	-3	0	0	0	0	0
s_1	1	3	**2**	1	0	0	0	18
s_2	-2	-3	0	0	1	0	0	6
s_3	1	-5	1	0	0	1	0	15
s_4	-1	3	0	0	0	0	1	6

The maximum value between the absolute value of the reduced cost's unrestricted variable and positive value of the reduced cost's restricted variable is 3. Thus, the entering variable is x_3. Since x_3 is the restricted variable. The index of the leaving variables can be calculated from argmin $\left\{\frac{8}{2}, \frac{12}{3}, \frac{6}{1}\right\} = 1$. Thus, the leaving variable is s_1. After pivoting, the updated tableau can be presented as follows:

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	5/2	5/2	0	3/2	0	0	0	27
x_3	1/2	3/2	1	1/2	0	0	0	9
s_2	**-2**	-3	0	0	1	0	0	6
s_3	1/2	-13/2	0	-1/2	0	1	0	6
s_4	-1	3	0	0	0	0	1	6

For this tableau, unrestricted variable x_1 is the entering variable, and the reduced cost is the negative value. Therefore, the index of leaving variables can be calculated as argmin $\left\{\frac{6}{|-2|}, \frac{6}{|-1|}\right\} = 2$. Therefore, the leaving variable is s_2.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	0	-5/4	0	3/2	5/4	0	0	69/8
x_3	0	3/4	1	1/2	1/4	0	0	21/2
x_1	-1	-3/2	0	0	1/2	0	0	3
s_3	0	-29/4	0	-1/2	1/4	1	0	15/2
s_4	0	**9/2**	0	0	-1/2	0	1	3

The entering variable is x_2 which is unrestricted variable with the positive reduced cost. Thus, the leaving variable is s_4.

	x_1	x_2	x_3	s_1	s_2	s_3	s_4	RHS
z	0	0	0	3/2	10/9	0	2/7	106/3
x_3	0	0	1	1/2	1/3	0	-1/6	10
x_1	-1	0	0	0	1/3	0	1/3	4
s_3	0	0	0	-1/2	-5/9	1	8/5	37/3
x_2	0	1	0	0	-1/9	0	2/9	2/3

This tableau is the optimal tableau with the optimal value as 106/3 and the optimal solution $[x_1, x_2, x_3]^\top$ as $[-4, 2/3, 10]^\top$.

3.1 Results and experiments

In this section, the computational results of the new technique for solving the unrestricted variables LP problem are performed on the randomly generated linear programming problems of various sizes. The computational time between the new technique and

the Dantzig's rule of the simplex algorithm for the unrestricted variables problem are compared. The experiments were operated using the Intel(R) Core(TM) i7-3770 CPU@ 3.40 GHz processor with 8 GB RAM on Windows 10. All methods were written by NumPy library from Python.

3.1.1 The randomly generated problem

The randomly generated unrestricted variable linear programming test problems are maximization problems. Since the randomly generated problems are created to test the effectiveness of the new technique for solving the unrestricted variable. Therefore, to make it easier to have a starting point for performing the standard simplex method and the new technique, all variables from dual problems associated with the primal of unrestricted variable problems are generated according to the following criteria:

The cost vector ($\mathbf{c}$) of a dual problem is the right-hand side of the primal problem generated by $c_i \in [-9, 9]$. The matrix ($\mathbf{A}$) of the dual problem is created with $a_{ij} \in [-9, 9]$. To guarantee a feasible region of the primal, right-hand side vector ($\mathbf{b}$) is created by generating feasible solution ($\mathbf{x}$) with $x_i \in [0, 9]$. After that, $\mathbf{b}$ is created as $\mathbf{b} = \mathbf{Ax} + |\min(\mathbf{Ax})| + 1$.

The different size of m and n with the randomly generated problems are defined as following. Let $m \in \{100, 200, 300, 400, 500, 600\}$ and $n \in \left\{\frac{m}{5}, \frac{2m}{5}, \frac{3m}{5}, \frac{4m}{5}, \frac{5m}{5}\right\}$. For each size of the randomly generated LP problem, the wall-clock times are averaged from 5 LP problems.

3.1.2 Computational result

The average wall-clock time (in seconds) of the new technique for solving the unrestricted variables and the simplex method with Dantzig's pivot rule are demonstrated in Table 3.1. Row represents the number of constraints in the LP model, Col represents the number of variables in the LP model, IDPUR represents the new technique of solving the unrestricted variables, that is, the simplex method with improvement Dantzig's pivot rule for the unrestricted variable model, DPUR represents the simplex method with Dantzig's pivot rule for the unrestricted variable model, the boldface numbers identify the smallest average wall-clock time, and the number in parenthesis represents the standard deviations

of each size of the LP model.

Table 3.1: The comparison of IDPUR and DPUR using the average wall-clock time-randomly generated linear programming problems.

Row	Col	IDPUR	DPUR
100	20	**0.0295** (±0.0060)	0.0329 (±0.0106)
	40	**0.1526** (±0.0266)	0.1630 (±0.0475)
	60	**0.1956** (±0.0812)	0.2191 (±0.0718)
	80	**0.1526** (±0.0534)	0.1819 (±0.0571)
	100	**0.0924** (±0.0203)	0.0948 (±0.0159)
200	40	0.3658 (±0.1360)	**0.3427** (±0.1177)
	80	**0.8216** (±0.1995)	0.9295 (±0.1572)
	120	**0.6104** (±0.1910)	0.7979 (±0.2450)
	160	**0.5396** (±0.2442)	0.7418 (±0.1375)
	200	**0.6315** (±0.1248)	0.8941 (±0.1243)
300	60	0.9805 (±0.2120)	**0.9333** (±0.1112)
	120	**2.6352** (±1.1191)	2.8538 (±1.0607)
	180	**1.4727** (±0.5200)	1.9063 (±0.5959)
	240	**1.1579** (±0.1807)	1.7900 (±0.2273)
	300	**1.1642** (±0.1527)	2.2123 (±0.0620)
400	80	**2.6532** (±0.5038)	2.9222 (±0.4435)
	160	**3.5726** (±0.9648)	4.3390 (±1.3599)
	240	**2.0851** (±0.4609)	3.0290 (±0.5350)
	320	**2.3294** (±0.3196)	3.8877 (±0.3421)
	400	**1.9059** (±0.2849)	4.3740 (±0.3238)
500	100	**4.7147** (±0.5056)	5.4435 (±0.5705)

Continued on the next page

Table 3.1 – *Continued from the previous page*

Row	Col	IDPUR	DPUR
500	200	**6.9613** (±2.4625)	8.2117 (±2.4581)
	300	**3.4911** (±1.2604)	5.3059 (±1.4587)
	400	**3.3537** (±0.7215)	6.5730 (±0.8840)
	500	**3.5461** (±0.3594)	8.3984 (±0.4744)
600	120	**8.7293** (±0.4279)	9.8658 (±0.5491)
	240	**8.6296** (±3.2525)	10.6071 (±3.6216)
	360	**4.3597** (±0.7495)	7.3947 (±0.7521)
	480	**5.3092** (±1.0826)	10.6954 (±1.3462)
	600	**5.1970** (±0.6545)	13.7854 (±0.8306)
Average		**2.5947**	3.9642

Table 3.1 shows that most average time of the new technique is less than the simplex method with the Dantzig's pivot rule. For solving the unrestricted variables LP model by the standard simplex method with the Dantzig's pivot rule, it is necessary to change the unrestricted variable into two new distinct restricted variables. Therefore, the size of the LP model is extended with the number of unrestricted variables. While the new technique for solving the unrestricted variable does not need to replace it by two non-negative variables. Thus, the size of the LP model is smaller than the standard simplex method with the Dantzig's pivot rule which affects the time to find the solution. It clearly shows that the new technique for solving the unrestricted variable is more effective than the simplex method with the Dantzig's pivot rule.

3.2 Conclusion

Generally, most LP models assume that all variables are non-negative. However, there are some situations where some variables can be either negative value, zero value, or positive value. For example, the profit or loss variable in the assignment problem, the

scheduling problem, or the product-mix problem. It can hold any value if it is positive then it represents a profit and if it is negative then it indicates a loss. Another example, the temperature variable in the modeling of the chemical process can be the unrestricted variable.

Traditionally to solve the unrestricted variable LP model, all unrestricted variables are replaced by the difference of two new restricted variables. As a result, the size of the LP model is extended. Note these two new restricted variables will always have the opposite signs of the coefficients. Therefore, the new technique for solving the unrestricted variable model is proposed to find the solution of the unrestricted variable LP model without increasing the size of the LP problem.

The standard simplex method with the Dantzig's pivot rule is applied to test the effectiveness of the new technique by using the randomly generated LP problems. Since the solution and the number of iterations of both methods are equal for all LP models, the time complexity of both methods are the same. Thus, only the wall-clock time is shown in the experimental results. The new technique outperforms the standard simplex method for most LP model sizes except the LP model size of 200×40 and 300×60.

The average wall-clock time of both methods is rarely different. Although the new technique is calculated on the smaller simplex tableau than the standard simplex method, the process of finding the entering variable and the leaving variable, and updating the simplex tableau of the new technique is quite complicated. Moreover, the space requirement of both methods are the same. For the unrestricted variables LP model where m is the number of constraints, and n is the number of unrestricted variables, the simplex tableau for the standard simplex method has $(m+1)(m+2n+1) = m^2+2mn+2m+2n+1 = O(mn)$ space requirement. Likewise, the simplex method for the new technique has a $(m+1)(m+n+1) = m^2+mn+2m+n+1 = O(mn)$ space requirement.

Consider the small unrestricted variable LP model as follows:

$$\begin{aligned} \text{Maximize} \quad & 3x_1 - x_2 \\ \text{subject to} \quad & x_1 - x_2 + x_3 = 5 \\ & -x_1 - 4x_2 + x_4 = 8 \\ & x_1, x_2 \text{ unrestricted}, x_3, x_4 \geq 0, \end{aligned} \tag{3.5}$$

and

$$\begin{aligned} \text{Maximize} \quad & 3x_1 - x_2 \\ \text{subject to} \quad & x_1 - x_2 + x_3 = 5 \\ & -x_1 - 4x_2 + x_4 = 8 \\ & x_1, x_3, x_4 \geq 0, x_2 \text{ unrestricted}. \end{aligned} \tag{3.6}$$

For unrestricted variable LP (3.5), the simplex tableau for the standard simplex method is created by 3 rows and 7 columns with 21 elements. While the simplex tableau for the new technique is created by 3 rows and 5 columns with 15 elements. It shows that the new technique can reduce the number of columns from the standard simplex method up to n columns where n is the number of unrestricted variables and it can reduce the number of elements of the simplex tableau up to $(m+1)n$ where m is the number of constraints.

For unrestricted variable LP (3.6), the size of the simplex tableau of the standard simplex method consists of 3 rows and 6 columns with 18 elements and the size of the simplex tableau for the new technique consists of 3 rows and 5 columns with 15 elements. Although the new technique can reduce one column from the standard simplex method. However, the complexity of calculations for each iteration of the new technique can be time-consuming. Therefore, the new technique is suitable for the LP model with a lot of unrestricted variables.

From the advantages of the new technique, it is applied in AJSP which is proposed in Chapter 5. The new technique is used to solve the transformed LP model that all variables are unrestricted variables to reduce the total computational time.

CHAPTER IV

SELF-REGULATING ARTIFICIAL-FREE LINEAR PROGRAMMING SOLVER USING A JUMP AND SIMPLEX METHOD

The iterative jump method is proposed in the first section of this chapter. For each iteration of the iterative jump method, it moves an interior feasible point to a new feasible point along the direction of improvement of the objective value whereas it still maintains the feasibility of a new point. The process of the jump method can dodge some unnecessary extreme points. Therefore, the self-regulating artificial-free linear programming solver using a jump and simplex method (SAJS) is presented. It starts by creating the LP relaxation having the constraints that their gradient vector makes an acute angle with the gradient vector of the objective function. After that, the initial starting point of the iterative jump method is searched. Next, the iterative jump method is performed on the LP relaxation. After it terminates, the last jump point obtained from the iterative jump method is relocated to the origin point of the new LP model called the transformed LP model and feasible non-acute constraints of the last jump point are reinserted. If there are no non-acute constraints remaining, then the simplex method can start to find the solution, immediately. Otherwise, the rest of the non-acute constraints are reinserted. After that, the dual simplex is performed when the dual solution is feasible. But if the dual solution is infeasible, the technique of Pan [2] is applied before performing the dual simplex.

4.1 The iterative jump method

The process of the iterative jump method is moving an initial feasible point along a feasible direction that improves the objective values. For each iteration, the new point derived from the move is called the improved feasible point. It will be continuously updated until it meets the stopping criteria. The iterative jump method is divided into two phases, namely the initial jump phase and the iterative jump phase. Consider an LP

model in the following form:

$$\begin{aligned} &\text{Maximize} && \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} && \mathbf{A}\mathbf{x} \leq \mathbf{b}. \end{aligned} \tag{4.1}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{m \times n}$. Let $Index = \{1, 2, \ldots, m\}$.

For the initial jump phase, the initial feasible point ($\mathbf{x}^{(0)}$) is moved along the feasible direction of the gradient vector of the objective function ($\mathbf{c}$) with the step size (λ). The process of the initial jump phase is explained as follows:

Let $\mathbf{x}^{(0)}$ be a feasible point that $\mathbf{b} - \mathbf{A}\mathbf{x}^{(0)} \geq \mathbf{0}$ and $\mathbf{v} = \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$. Thus, the new jump point $\mathbf{x}^{(1)}$ is calculated as follows:

$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda \mathbf{v}$ where $\lambda = \min\limits_{i \in Index} \left\{ \dfrac{b_i - \mathbf{A}_{i:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{i:}^\top \mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^\top \mathbf{v} > 0 \right\}$, and $\gamma^{(1)}$ is the index of the constraint which $\mathbf{x}^{(1)}$ is binding by $\gamma^{(1)} = \operatorname{argmin}\limits_{i \in Index} \left\{ \dfrac{b_i - \mathbf{A}_{i:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{i:}^\top \mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^\top \mathbf{v} > 0 \right\}$.

After $\mathbf{x}_1$ is obtained, then $\mathbf{x}_1$ is binding on the constraint $\gamma^{(1)}$ which it can be easily proven as the following.

Since $\lambda = \dfrac{b_{\gamma^{(1)}} - \mathbf{A}_{\gamma^{(1)}:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{\gamma^{(1)}:}^\top \mathbf{v}}$, then $\mathbf{A}_{\gamma^{(1)}:}^\top \mathbf{x}_1 = \mathbf{A}_{\gamma^{(1)}:}^\top (\mathbf{x}^{(0)} + \lambda \mathbf{v}) = \mathbf{A}_{\gamma^{(1)}:}^\top \mathbf{x}^{(0)} + \lambda \mathbf{A}_{\gamma^{(1)}:}^\top \mathbf{v} = \mathbf{A}_{\gamma^{(1)}:}^\top \mathbf{x}^{(0)} + \left(\dfrac{b_{\gamma^{(1)}} - \mathbf{A}_{\gamma^{(1)}:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{\gamma^{(1)}:}^\top \mathbf{v}} \right) \mathbf{A}_{\gamma^{(1)}:}^\top \mathbf{v} = b_{\gamma^{(1)}}$.

For the process of the initial jump phase, it has been proven in Theorem 4.1 that this direction will make the better objective value of the improved feasible point, and the step size will maintain the feasibility of the improved feasible point.

Theorem 4.1. Given LP (4.1) with $\mathbf{c} \neq \mathbf{0}$. Let $\mathbf{x}^{(0)}$ be a feasible point and $\mathbf{v} = \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$. Then $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda \mathbf{v}$ is feasible which $\lambda = \min \left\{ \dfrac{b_i - \mathbf{A}_{i:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{i:}^\top \mathbf{v}} \;\middle|\; i \in Index \text{ and } \mathbf{A}_{i:}^\top \mathbf{v} > 0 \right\}$ and $\mathbf{c}^\top \mathbf{x}^{(0)} \leq \mathbf{c}^\top \mathbf{x}^{(1)}$.

Proof. Let $\mathbf{x}^{(0)}$ be a feasible point for LP (4.1) and $\mathbf{v} = \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$ where $\mathbf{c} \neq \mathbf{0}$.
Let $\lambda = \min \left\{ \dfrac{b_i - \mathbf{A}_{i:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{i:}^\top \mathbf{v}} \;\middle|\; i \in Index \text{ and } \mathbf{A}_{i:}^\top \mathbf{v} > 0 \right\}$ and $\mathbf{x}^1 = \mathbf{x}^0 + \lambda \mathbf{v}$.

Since $\mathbf{x}^{(0)}$ is the feasible point, $b_u - \mathbf{A}_{u:}^\top \mathbf{x}^{(0)} \geq 0$ for all $u \in Index$.

For $u \in Index$ with $\mathbf{A}_{u:}^\top \mathbf{v} > 0$,

$$\begin{aligned}
\lambda &\leq \frac{b_u - \mathbf{A}_{u:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{u:}^\top \mathbf{v}} \\
\lambda(\mathbf{A}_{u:}^\top \mathbf{v}) &\leq b_u - \mathbf{A}_{u:}^\top \mathbf{x}^{(0)} \\
\mathbf{A}_{u:}^\top \left(\mathbf{x}^{(0)} + \lambda \mathbf{v}\right) &\leq b_u \\
\mathbf{A}_{u:}^\top \mathbf{x}^{(1)} &\leq b_u.
\end{aligned}$$

For $u \in Index$ with $\mathbf{A}_{u:}^\top \mathbf{v} < 0$,

$$\begin{aligned}
\frac{b_u - \mathbf{A}_{u:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{u:}^\top \mathbf{v}} &< \lambda \\
\lambda(\mathbf{A}_{u:}^\top \mathbf{v}) &< b_u - \mathbf{A}_{u:}^\top \mathbf{x}^{(0)} \\
\mathbf{A}_{u:}^\top \left(\mathbf{x}^{(0)} + \lambda \mathbf{v}\right) &< b_u \\
\mathbf{A}_{u:}^\top \mathbf{x}^{(1)} &< b_u.
\end{aligned}$$

For $u \in Index$ with $\mathbf{A}_{u:}^\top \mathbf{v} = 0$,

$$\mathbf{A}_{u:}^\top \mathbf{x}^{(1)} = \mathbf{A}_{u:}^\top \left(\mathbf{x}^{(0)} + \lambda \mathbf{v}\right) = \mathbf{A}_{u:}^\top \mathbf{x}^{(0)} + \lambda \mathbf{A}_{u:}^\top \mathbf{v} = \mathbf{A}_{u:}^\top \mathbf{x}^{(0)} \leq \mathbf{b}_u.$$

Therefore, $\mathbf{A}\mathbf{x}^{(1)} \leq \mathbf{b}$. Hence, $\mathbf{x}^{(1)}$ is the feasible point.

Next, it can be shown that $\mathbf{c}^\top \mathbf{x}^{(0)} \leq \mathbf{c}^\top \mathbf{x}^{(1)}$.

Since $\mathbf{v} = \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$,

$$\begin{aligned}
\mathbf{c}^\top \mathbf{x}^{(1)} &= \mathbf{c}^\top (\mathbf{x}^{(0)} + \lambda \mathbf{v}) \\
&= \mathbf{c}^\top \mathbf{x}^{(0)} + \lambda \mathbf{c}^\top \mathbf{v} \\
&= \mathbf{c}^\top \mathbf{x}^{(0)} + \lambda \mathbf{c}^\top \left(\frac{\mathbf{c}}{\|\mathbf{c}\|}\right) \\
&= \mathbf{c}^\top \mathbf{x}^{(0)} + \lambda \|\mathbf{c}\|.
\end{aligned}$$

Since $\lambda \geq 0$ and $\|\mathbf{c}\| \geq 0$, $\mathbf{c}^\top \mathbf{x}^{(1)} \geq \mathbf{c}^\top \mathbf{x}^{(0)}$. □

After the initial jump phase terminates, the iterative jump phase will start by the improved feasible point ($\mathbf{x}^{(1)}$) in the initial jump phase. The direction of the iterative jump phase is calculated from the gradient vector of the objective function and the gradient vector of the constraint that the current improved feasible point is binding. Therefore, the process in the iterative jump phase for each iteration can be calculated as follows:

Let $\mathbf{v} = -\dfrac{\mathbf{A}_{\gamma^{(k)}:}}{\|\mathbf{A}_{\gamma^{(k)}:}\|} + \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$ where $\gamma^{(k)}$ is the index of the constraint that $\mathbf{x}^{(k)}$ is binding.

Thus, $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda\mathbf{v}$ where $\lambda = \min\limits_{i \in Index} \left\{ \dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^{\top}\mathbf{v} > 0 \right\}$ and $\gamma^{(k+1)} = \operatorname{argmin}\limits_{i \in Index} \left\{ \dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^{\top}\mathbf{v} > 0 \right\}$ for $k = 1, 2, 3, ...$

Likewise, the new improved feasible point is moved by the direction of the iterative jump phase will have better objective values. Moreover, the iterative jump phase will still maintain the feasibility of the new improved feasible point which will be proven in Theorem 4.2.

Theorem 4.2. Given LP (4.1) with $\mathbf{c} \neq \mathbf{0}$. Let $\mathbf{x}^{(k)}$ be a feasible point which lies on γ^{th} constraint of $\mathbf{A}$ and $\mathbf{v} = \dfrac{-\mathbf{A}_{\gamma:}}{\|\mathbf{A}_{\gamma:}\|} + \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$. Then $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda\mathbf{v}$ is feasible which $\lambda = \min \left\{ \dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}} \;\middle|\; i \in Index \text{ and } \mathbf{A}_{i:}^{\top}\mathbf{v} > 0 \right\}$ and $\mathbf{c}^{\top}\mathbf{x}^{(k)} \leq \mathbf{c}^{\top}\mathbf{x}^{(k+1)}$.

Proof. Let $\mathbf{x}^{(k)}$ be a feasible point for LP (4.1) which lies on γ^{th} constraint and $\mathbf{v} = \dfrac{-\mathbf{A}_{\gamma:}}{\|\mathbf{A}_{\gamma:}\|} + \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$ where $\mathbf{c} \neq \mathbf{0}$.

Let $\lambda = \min \left\{ \dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}} \;\middle|\; i \in Index \text{ and } \mathbf{A}_{i:}^{\top}\mathbf{v} > 0 \right\}$ and $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda\mathbf{v}$.

Since $\mathbf{x}^{(k)}$ is the feasible point, $b_u - \mathbf{A}_{u:}^{\top}\mathbf{x}^{(k)} \geq 0$ for all $u \in Index$.

For $u \in Index$ with $\mathbf{A}_{u:}^{\top}\mathbf{v} > 0$,

$$\begin{aligned}
\lambda &\leq \frac{b_u - \mathbf{A}_{u:}^{\top}\mathbf{x}^{(k)}}{\mathbf{A}_{u:}^{\top}\mathbf{v}} \\
\lambda(\mathbf{A}_{u:}^{\top}\mathbf{v}) &\leq b_u - \mathbf{A}_{u:}^{\top}\mathbf{x}^{(k)} \\
\mathbf{A}_{u:}^{\top}\left(\mathbf{x}^{(k)} + \lambda\mathbf{v}\right) &\leq b_u \\
\mathbf{A}_{u:}^{\top}\mathbf{x}^{(k+1)} &\leq b_u.
\end{aligned}$$

For $u \in Index$ with $\mathbf{A}_{u:}^{\top}\mathbf{v} < 0$,

$$\begin{aligned}
\frac{b_u - \mathbf{A}_{u:}^{\top}\mathbf{x}^{(k)}}{\mathbf{A}_{u:}^{\top}\mathbf{v}} &< \lambda \\
\lambda(\mathbf{A}_{u:}^{\top}\mathbf{v}) &< b_u - \mathbf{A}_{u:}^{\top}\mathbf{x}^{(k)} \\
\mathbf{A}_{u:}^{\top}\left(\mathbf{x}^{(k)} + \lambda\mathbf{v}\right) &< b_u \\
\mathbf{A}_{u:}^{\top}\mathbf{x}^{(k+1)} &< b_u.
\end{aligned}$$

For $u \in Index$ with $\mathbf{A}_{u:}^{\top}\mathbf{v} = 0$,

$$\mathbf{A}_{u:}^{\top}\mathbf{x}^{(k+1)} = \mathbf{A}_{u:}^{\top}\left(\mathbf{x}^{(k)} + \lambda\mathbf{v}\right) = \mathbf{A}_{u:}^{\top}\mathbf{x}^{(k)} + \lambda\mathbf{A}_{u:}^{\top}\mathbf{v} = \mathbf{A}_{u:}^{\top}\mathbf{x}^{(k)} \leq \mathbf{b}_u.$$

Therefore, $\mathbf{A}\mathbf{x}^{(k+1)} \leq \mathbf{b}$. Hence, $\mathbf{x}^{(k+1)}$ is the feasible point.
Next, it can be shown that $\mathbf{c}^{\top}\mathbf{x}^{(k)} \leq \mathbf{c}^{\top}\mathbf{x}^{(k+1)}$.
Consider,

$$\begin{aligned}
\mathbf{c}^{\top}\mathbf{x}^{(k+1)} &= \mathbf{c}^{\top}(\mathbf{x}^{(k)} + \lambda\mathbf{v}) \\
&= \mathbf{c}^{\top}\mathbf{x}^{(k)} + \lambda\mathbf{c}^{\top}\mathbf{v} \\
&= \mathbf{c}^{\top}\mathbf{x}^{(k)} + \lambda\mathbf{c}^{\top}\left(\frac{-\mathbf{A}_{\gamma:}}{\|\mathbf{A}_{\gamma:}\|} + \frac{\mathbf{c}}{\|\mathbf{c}\|}\right) \\
&= \mathbf{c}^{\top}\mathbf{x}^{(k)} + \frac{-\lambda\mathbf{c}^{\top}\mathbf{A}_{\gamma:}}{\|\mathbf{A}_{\gamma:}\|} + \frac{\lambda\|\mathbf{c}\|^2}{\|\mathbf{c}\|} \\
&= \mathbf{c}^{\top}\mathbf{x}^{(k)} + \frac{-\lambda\|\mathbf{A}_{\gamma:}\|\|\mathbf{c}\|cos(\theta)}{\|\mathbf{A}_{\gamma:}\|} + \lambda\|\mathbf{c}\| \\
&= \mathbf{c}^{\top}\mathbf{x}^{(k)} + \lambda\|\mathbf{c}\|(1 - cos(\theta)).
\end{aligned}$$

where θ is the angle between the gradient vector of $\mathbf{A}_{\gamma:}$ and the gradient vector of $\mathbf{c}$.
Since $\lambda \geq 0$, $\|\mathbf{c}\| > 0$ and $1 - cos(\theta) > 0$, thus, $\mathbf{c}^{\top}\mathbf{x}^{(k+1)} \geq \mathbf{c}^{\top}\mathbf{x}^{(k)}$. □

The algorithm of the iterative jump method are summarized in Algorithm 1. After Algorithm 1 terminates, the last jump point ($\hat{\mathbf{x}}$) from the iterative jump method is discovered.

In the next step, the simplex method is applied to find the exact solution. Since $\hat{\mathbf{x}}$ is not an extreme point, the transformed LP model is created by relocating $\hat{\mathbf{x}}$ to the

Algorithm 1 The iterative jump method.

Input: $\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{x}^{(0)}, \gamma^{(0)}, Index, Tol$
Output: $\hat{\mathbf{x}}$

Compute $\mathbf{v} = \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$ where $\mathbf{c} \neq 0$.
Set $\alpha = M$, where M be a large constant.
for i $\in$ Index **do**
    if $\mathbf{A}_{i:}^{\top}\mathbf{v} > 0$ and $b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(0)} > 0$ **then**
        Compute $Dist = \dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(0)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}}$.
        if $Dist < \alpha$ **then**
            Define $\alpha = Dist$.
            Define $\gamma^{(1)} = i$.
if $\alpha \neq M$ **then**
    Compute $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha\mathbf{v}$.
else
    break
Set $k = 1$, $\Delta z^{(2)} = \Delta z^{(1)} = 1$ where $\Delta z^{(2)}$ and $\Delta z^{(1)}$ are the initial values of consecutive difference.
while $\dfrac{\Delta z^{(k+1)}}{\Delta z^{(k)}} > Tol$ **do**
    Compute $\mathbf{v} = \dfrac{-\mathbf{A}_{\gamma^{(k)}:}}{\|\mathbf{A}_{\gamma^{(k)}:}\|} + \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$, where $\mathbf{c} \neq 0$.
    Construct $Index' = Index \setminus \{\gamma^{(k)}\}$.
    Set $\alpha = M$, where M be a large constant.
    for i $\in Index'$ **do**
        if $\mathbf{A}_{i:}^{\top}\mathbf{v} > 0$ and $b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)} > 0$ **then**
            Compute $Dist = \dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}}$.
            if $Dist < \alpha$ **then**
                Define $\alpha = Dist$.
                Define $\gamma^{(k+1)} = i$.
    if $\alpha \neq M$ **then**
        Compute $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{v}$.
        Compute $\Delta z^{(k+1)} = \mathbf{c}\mathbf{x}^{(k+1)} - \mathbf{c}\mathbf{x}^{(k)}$.
        Compute $k = k + 1$.
    else
        break
Set $\hat{\mathbf{x}} = \mathbf{x}^{(k+1)}$

origin point. Let $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$ for $\mathbf{x}$ in LP (4.1). Consequently, the transformed LP model is created as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \tilde{\mathbf{x}} + \mathbf{c}^\top \hat{\mathbf{x}} \\ \text{subject to} \quad & \mathbf{A}\tilde{\mathbf{x}} \leq \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}. \end{aligned} \tag{4.2}$$

Whereupon the slack variable is added to LP (4.2).

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \tilde{\mathbf{x}} + \mathbf{c}^\top \hat{\mathbf{x}} \\ \text{subject to} \quad & \mathbf{A}\tilde{\mathbf{x}} + \mathbf{s} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}, \\ & \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{4.3}$$

However, LP (4.3) is an unrestricted variable LP model, $\tilde{\mathbf{x}}$ is defined as the subtraction of two restricted variables as $\tilde{\mathbf{x}} = \mathbf{x}^+ - \mathbf{x}^-$ where $\mathbf{x}^+, \mathbf{x}^- \geq \mathbf{0}$. Therefore, the standard form of LP (4.3) is constructed as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \mathbf{x}^+ - \mathbf{c}^\top \mathbf{x}^- + \mathbf{c}^\top \hat{\mathbf{x}} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x}^+ - \mathbf{A}\mathbf{x}^- + \mathbf{s} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}, \\ & \mathbf{x}^+, \mathbf{x}^-, \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{4.4}$$

Then the simplex method can start to find the solution, immediately. Lastly, the solution is restored to the solution of original LP (4.1).

The iterative jump method requires an initial feasible point before start, in which finding that point is quite complicated. Therefore, SAJS is proposed in the next section. It applies the iterative jump on the relaxation model since the LP relaxation can guarantee the existence of the feasible points.

4.2 Self-regulating artificial-free linear programming solver using a jump and simplex method

From previous work (Section 2.7), it can solve the LP model without artificial

variables by creating the LP relaxation. Therefore, SAJS is proposed in this section. Firstly, SAJS starts by creating the LP relaxation with only acute constraints. The acute constraints are the constraints that their gradient vector makes an acute angle with the gradient vector of the objective function. The rest of the constraints are the non-acute constraints. Secondly, the initial jump point of the iterative jump method is calculated from the LP relaxation. The iterative jump method is performed on the LP relaxation until it satisfies the stopping criterion. Thirdly, after the iterative jump method terminates, the last jump point is relocated to the origin point of the new LP model called the transformed LP model. Fourthly, the non-acute constraints that the last jump point satisfies are reinserted into the transformed LP model called the full transformed LP model. As a result, the current origin point is still a feasible point for this LP model. However, if the remaining non-acute constraints do not exist, then the simplex method is performed to find the solution, immediately. Otherwise, the remaining non-acute constraints are reinserted to the full transformed LP model. Since the current origin point is infeasible for this LP model, the perturbation technique by Pan [2] is applied to change the infeasible dual solution to a feasible one before performing the dual simplex.

Consider a linear program in the following form:

$$\begin{aligned} &\text{Maximize} \quad \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}. \end{aligned} \tag{4.5}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{m \times n}$. Let $Index = \{1, 2, \ldots, m\}$.

The steps of SAJS are summarized as follows:

Step 1: All constraints are separated into two groups: the group of acute constraints

$$\mathbf{I}_{Acute} = \left\{ i \in Index \mid \mathbf{A}_{i:}^\top \mathbf{c} > 0 \right\},$$

and the group of non-acute constraints

$$\mathbf{I}_{NonAcute} = \left\{ i \in Index \mid \mathbf{A}_{i:}^\top \mathbf{c} \leq 0 \right\}.$$

Step 2: The LP relaxation is constructed with acute constraints as follows:

$$\begin{aligned} &\text{Maximize} && \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} && \mathbf{A}_{\mathbf{I}_{Acute}} \mathbf{x} \leq \mathbf{b}_{\mathbf{I}_{Acute}}. \end{aligned} \tag{4.6}$$

Step 3: The initial jump point of the iterative jump method is found.

Let $\mathbf{I}^{-}_{Acute} = \{j \in \mathbf{I}_{Acute} \mid b_j < 0\}$.

If $\mathbf{b}_{\mathbf{I}_{Acute}} \geq \mathbf{0}$, then $\mathbf{x}^{(0)} = \lambda \mathbf{c}$ where $\lambda = \min\limits_{i \in \mathbf{I}_{Acute}} \left\{ \dfrac{b_i}{\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$ and

$\gamma^{(0)} = \operatorname{argmin}\limits_{i \in \mathbf{I}_{Acute}} \left\{ \dfrac{b_i}{\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$.

Otherwise, $\mathbf{x}^{(0)} = -\lambda \mathbf{c}$ where $\lambda = \max\limits_{i \in \mathbf{I}^{-}_{Acute}} \left\{ \dfrac{b_i}{-\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$ and

$\gamma^{(0)} = \operatorname{argmax}\limits_{i \in \mathbf{I}^{-}_{Acute}} \left\{ \dfrac{b_i}{-\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$.

From both cases, $\mathbf{x}^{(0)}$ is feasible. For case of $\mathbf{b}_{\mathbf{I}_{Acute}} \geq \mathbf{0}$, it is proven in [4]. Another case is proven in Theorem 4.3.

Theorem 4.3. Given LP (4.5) and $\mathbf{I}_{Acute} = \left\{ i \in Index | \mathbf{A}_{i:}^\top \mathbf{c} > 0 \right\}$. If $\mathbf{b}_{\mathbf{I}_{Acute}} \geq \mathbf{0}$ and $\lambda = \min\limits_{i \in \mathbf{I}_{Acute}} \left\{ \dfrac{b_i}{\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$, then $\mathbf{x}^{(0)} = \lambda \mathbf{c}$ is feasible.

Proof. Suppose $\mathbf{b}_{\mathbf{I}_{Acute}} \geq \mathbf{0}$ and $\lambda = \min\limits_{i \in \mathbf{I}_{Acute}} \left\{ \dfrac{b_i}{\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$. Since $b_i \geq 0$ and $\mathbf{A}_{i:}^\top \mathbf{c} > 0$ for all $i \in \mathbf{I}_{Acute}$, $\lambda \geq 0$. Therefore, $\lambda \leq \dfrac{b_h}{\mathbf{A}_{h:}^\top \mathbf{c}}$, for all $h \in \mathbf{I}_{Acute}$. Hence, $\lambda \left(\mathbf{A}_{h:}^\top \mathbf{c} \right) \leq b_h$. Choose $\mathbf{x}^{(0)} = \lambda \mathbf{c}$, it gets $\mathbf{A}_{h:}^\top \mathbf{x}^{(0)} \leq b_h$ for all $h \in \mathbf{I}_{Acute}$. Thus, $\mathbf{A}_{\mathbf{I}_{Acute}} \mathbf{x}^{(0)} \leq \mathbf{b}_{\mathbf{I}_{Acute}}$. □

Step 4: The iterative jump method is performed to find the improved feasible point by Algorithm 2. The inputs of the algorithm are $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{x}^{(0)}$, $\gamma^{(0)}$, $\mathbf{I}_{Acute}$ and ε. The stopping parameter (ε) is defined as the acceptable ratio improvement of two consecutive differences of the objective values. This is discussed in section 4.3.3.

Algorithm 2 The algorithm of the iterative jump method for SAJS.

Input: $\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{x}^{(0)}, \gamma^{(0)}, \mathbf{I}_{Acute}, \varepsilon$

Output: $\hat{\mathbf{x}}$

Set $k = 0$, $\Delta z^{(1)} = \Delta z^{(0)} = 1$ where $\Delta z^{(1)}$ and $\Delta z^{(0)}$ are the initial values of the consecutive difference.

while $\dfrac{\Delta z^{(k+1)}}{\Delta z^{(k)}} > \varepsilon$ **do**

Compute $\mathbf{v} = \dfrac{-\mathbf{A}_{\gamma^{(k)}:}}{\|\mathbf{A}_{\gamma^{(k)}:}\|} + \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$, where $\mathbf{c} \neq 0$.

Construct $\mathbf{I}'_{Acute} = \mathbf{I}_{Acute} \setminus \{\gamma^{(k)}\}$.

Set $\alpha = M$, where M be a large constant.

for i $\in \mathbf{I}'_{Acute}$ **do**

if $\mathbf{A}_{i:}^{\top}\mathbf{v} > 0$ and $b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)} > 0$ **then**

Compute $Dist = \dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}}$.

if $Dist < \alpha$ **then**

Define $\alpha = Dist$.

Define $\gamma^{(k+1)} = i$.

if $\alpha \neq M$ **then**

Compute $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{v}$.

Compute $\Delta z^{(k+1)} = \mathbf{c}\mathbf{x}^{(k+1)} - \mathbf{c}\mathbf{x}^{(k)}$.

Compute $k = k + 1$.

else

break

Set $\hat{\mathbf{x}} = \mathbf{x}^{(k+1)}$

After Algorithm 2 terminates, the last improved feasible point ($\hat{\mathbf{x}}$) is found.

Step 5: The last improved feasible point ($\hat{\mathbf{x}}$) is relocated to the origin point of the transformed LP model.

Let $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$. Therefore, the transformed LP model is created as

$$\begin{aligned} &\text{Maximize} \quad \mathbf{c}^\top \tilde{\mathbf{x}} + \mathbf{c}^\top \hat{\mathbf{x}} \\ &\text{subject to} \quad \mathbf{A}_{\mathbf{I}_{Acute}} \tilde{\mathbf{x}} \leq \mathbf{b}_{\mathbf{I}_{Acute}} - \mathbf{A}_{\mathbf{I}_{Acute}} \hat{\mathbf{x}}. \end{aligned} \tag{4.7}$$

Step 6: The transformed LP model will be converted into the standard form.

First, the slack variable ($\mathbf{s}$) is added to the transformed LP model.

$$\begin{aligned} &\text{Maximize} \quad \mathbf{c}^\top \tilde{\mathbf{x}} + \mathbf{c}^\top \hat{\mathbf{x}} \\ &\text{subject to} \quad \mathbf{A}_{\mathbf{I}_{Acute}} \tilde{\mathbf{x}} + \mathbf{s} = \mathbf{b}_{\mathbf{I}_{Acute}} - \mathbf{A}_{\mathbf{I}_{Acute}} \hat{\mathbf{x}}, \\ &\qquad\qquad \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{4.8}$$

After that, the unrestricted variable ($\tilde{\mathbf{x}}$) is changed as the subtraction of two restricted variables. Let $\tilde{\mathbf{x}} = \mathbf{x}^+ - \mathbf{x}^-$ where $\mathbf{x}^+ , \mathbf{x}^- \geq \mathbf{0}$.

$$\begin{aligned} &\text{Maximize} \quad \mathbf{c}^\top \mathbf{x}^+ - \mathbf{c}^\top \mathbf{x}^- + \mathbf{c}^\top \hat{\mathbf{x}} \\ &\text{subject to} \quad \mathbf{A}_{\mathbf{I}_{Acute}} \mathbf{x}^+ - \mathbf{A}_{\mathbf{I}_{Acute}} \mathbf{x}^- + \mathbf{s} = \mathbf{b}_{\mathbf{I}_{Acute}} - \mathbf{A}_{\mathbf{I}_{Acute}} \hat{\mathbf{x}}, \\ &\qquad\qquad \mathbf{x}^+, \mathbf{x}^-, \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{4.9}$$

Step 7: The initial simplex tableau is constructed by assigning the initial basic feasible solution as the slack variable.

	$\mathbf{x}^+$	$\mathbf{x}^-$	$\mathbf{s}$	RHS
z	$-\mathbf{c}^\top$	$-(-\mathbf{c}^\top)$	$\mathbf{0}$	$\mathbf{c}^\top \hat{\mathbf{x}}$
$\mathbf{s}$	$\mathbf{A}_{\mathbf{I}_{Acute}}$	$-\mathbf{A}_{\mathbf{I}_{Acute}}$	$\mathbf{I}$	$\mathbf{b}_{\mathbf{I}_{Acute}} - \mathbf{A}_{\mathbf{I}_{Acute}} \hat{\mathbf{x}}$

Step 8: The non-acute constraints that are satisfied by $\hat{\mathbf{x}}$ are reinserted to the initial simplex tableau.

Let $\mathbf{I}^-_{NonAcute} = \left\{ i \in \mathbf{I}_{NonAcute} \mid b_i - \mathbf{A}_{i:}^\top \hat{\mathbf{x}} < 0 \right\}$ and

$\mathbf{I}^{+}_{NonAcute} = \{i \in \mathbf{I}_{NonAcute} \mid b_i - \mathbf{A}^{\top}_{i:}\hat{\mathbf{x}} \geq 0\}$.

Thus, if the set of $\mathbf{I}^{+}_{NonAcute}$ is not empty, the constraints in this set are added into the simplex tableau as follows:

	$\mathbf{x}^{+}$	$\mathbf{x}^{-}$	$\mathbf{s}$	$\mathbf{s}_1$	RHS
z	$-\mathbf{c}^{\top}$	$-(-\mathbf{c}^{\top})$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{c}^{\top}\hat{\mathbf{x}}$
$\mathbf{s}$	$\mathbf{A}_{\mathbf{I}_{Acute}}$	$-\mathbf{A}_{\mathbf{I}_{Acute}}$	$\mathbf{I}$	$\mathbf{0}$	$\mathbf{b}_{\mathbf{I}_{Acute}} - \mathbf{A}_{\mathbf{I}_{Acute}}\hat{\mathbf{x}}$
$\mathbf{s}_1$	$\mathbf{A}_{\mathbf{I}^{+}_{NonAcute}}$	$-\mathbf{A}_{\mathbf{I}^{+}_{NonAcute}}$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{b}_{\mathbf{I}^{+}_{NonAcute}} - \mathbf{A}_{\mathbf{I}^{+}_{NonAcute}}\hat{\mathbf{x}}$

Step 9: The rest of the non-acute constraints are reinserted to the simplex tableau. However, if $\mathbf{I}^{-}_{NonAcute}$ is empty, the simplex method is performed to find the solution of the transformed LP model, immediately. After that, the solution will be converted to the solution of the original LP model. Otherwise, the constraints in $\mathbf{I}^{-}_{NonAcute}$ are reinserted to the simplex tableau and then it goes to Step 10.

	$\mathbf{x}^{+}$	$\mathbf{x}^{-}$	$\mathbf{s}$	$\mathbf{s}_1$	$\mathbf{s}_2$	RHS
z	$-\mathbf{c}^{\top}$	$-(-\mathbf{c}^{\top})$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{c}^{\top}\hat{\mathbf{x}}$
$\mathbf{s}$	$\mathbf{A}_{\mathbf{I}_{Acute}}$	$-\mathbf{A}_{\mathbf{I}_{Acute}}$	$\mathbf{I}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{b}_{\mathbf{I}_{Acute}} - \mathbf{A}_{\mathbf{I}_{Acute}}\hat{\mathbf{x}}$
$\mathbf{s}_1$	$\mathbf{A}_{\mathbf{I}^{+}_{NonAcute}}$	$-\mathbf{A}_{\mathbf{I}^{+}_{NonAcute}}$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{0}$	$\mathbf{b}_{\mathbf{I}^{+}_{NonAcute}} - \mathbf{A}_{\mathbf{I}^{+}_{NonAcute}}\hat{\mathbf{x}}$
$\mathbf{s}_2$	$\mathbf{A}_{\mathbf{I}^{-}_{NonAcute}}$	$-\mathbf{A}_{\mathbf{I}^{-}_{NonAcute}}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{b}_{\mathbf{I}^{-}_{NonAcute}} - \mathbf{A}_{\mathbf{I}^{-}_{NonAcute}}\hat{\mathbf{x}}$

Step 10: For the current simplex tableau, the primal solution and the dual solution are always infeasible since the value of the reduced cost in column $\mathbf{x}^{+}$ and $\mathbf{x}^{-}$ are always opposite. Thus, the perturbation technique of Pan [2] is applied to make the dual solution feasible.

Let $(\mathbf{x}^+)^+$ be the variable from $\mathbf{x}^+$ with the positive reduced cost,

$(\mathbf{x}^+)^-$ be the variable from $\mathbf{x}^+$ with the negative reduced cost,

$(\mathbf{x}^-)^+$ be the variable from $\mathbf{x}^-$ with the positive reduced cost,

$(\mathbf{x}^-)^-$ be the variable from $\mathbf{x}^-$ with the negative reduced cost,

$(\mathbf{A}^+)^+$ be the column vector from $\mathbf{A}$ which corresponds to $\mathbf{x}^+$ with the positive reduced cost,

$(\mathbf{A}^+)^-$ be the column vector from $\mathbf{A}$ which corresponds to $\mathbf{x}^+$ with the negative reduced cost,

$(\mathbf{A}^-)^+$ be the column vector from $\mathbf{A}$ which corresponds to $\mathbf{x}^-$ with the positive reduced cost,

$(\mathbf{A}^-)^-$ be the column vector from $\mathbf{A}$ which corresponds to $\mathbf{x}^-$ with the negative reduced cost.

The values of the negative reduced cost in the simplex tableau are changed to positive which make the dual solution feasible. The positive value used in this dissertation is set to 1. Therefore, the perturbation simplex tableau is changed as follows:

	$(\mathbf{x}^+)^+$	$(\mathbf{x}^+)^-$	$(\mathbf{x}^-)^+$	$(\mathbf{x}^-)^-$	$\mathbf{s}$	$\mathbf{s}_1$	$\mathbf{s}_2$	RHS
z	$-\mathbf{c}^\top$	$-(\mathbf{1})$	$\mathbf{c}^\top$	$-(\mathbf{1})$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{c}^\top\hat{\mathbf{x}}$
$\mathbf{s}$	$(\mathbf{A}^+)^+_{\mathbf{I}_{Acute}}$	$(\mathbf{A}^+)^-_{\mathbf{I}_{Acute}}$	$(\mathbf{A}^-)^+_{\mathbf{I}_{Acute}}$	$(\mathbf{A}^-)^-_{\mathbf{I}_{Acute}}$	$\mathbf{I}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{b}_{\mathbf{I}_{Acute}} - \mathbf{A}_{\mathbf{I}_{Acute}}\hat{\mathbf{x}}$
$\mathbf{s}_1$	$(\mathbf{A}^+)^+_{\mathbf{I}^+_{NonAcute}}$	$(\mathbf{A}^+)^-_{\mathbf{I}^+_{NonAcute}}$	$(\mathbf{A}^-)^+_{\mathbf{I}^+_{NonAcute}}$	$(\mathbf{A}^-)^-_{\mathbf{I}^+_{NonAcute}}$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{0}$	$\mathbf{b}_{\mathbf{I}^+_{NonAcute}} - \mathbf{A}_{\mathbf{I}^+_{NonAcute}}\hat{\mathbf{x}}$
$\mathbf{s}_2$	$(\mathbf{A}^+)^+_{\mathbf{I}^-_{NonAcute}}$	$(\mathbf{A}^+)^-_{\mathbf{I}^-_{NonAcute}}$	$(\mathbf{A}^-)^+_{\mathbf{I}^-_{NonAcute}}$	$(\mathbf{A}^-)^-_{\mathbf{I}^-_{NonAcute}}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{b}_{\mathbf{I}^-_{NonAcute}} - \mathbf{A}_{\mathbf{I}^-_{NonAcute}}\hat{\mathbf{x}}$

Next, the dual simplex is performed to determine the feasible solution of the simplex tableau. After that, the original reduced costs of the transformed LP model are restored. If the dual solution is feasible, then the current solution is converted to the solution of the original LP model. Otherwise, the simplex method is performed to find the solution and that solution is converted to the solution of the original

LP model similarly. However, if the solution of the transformed LP relaxation is unbounded, then the solution of the original LP model is infeasible.

Next, the flowchart of SAJS, comprised of ten steps, is shown in Figure 4.1 - 4.3.

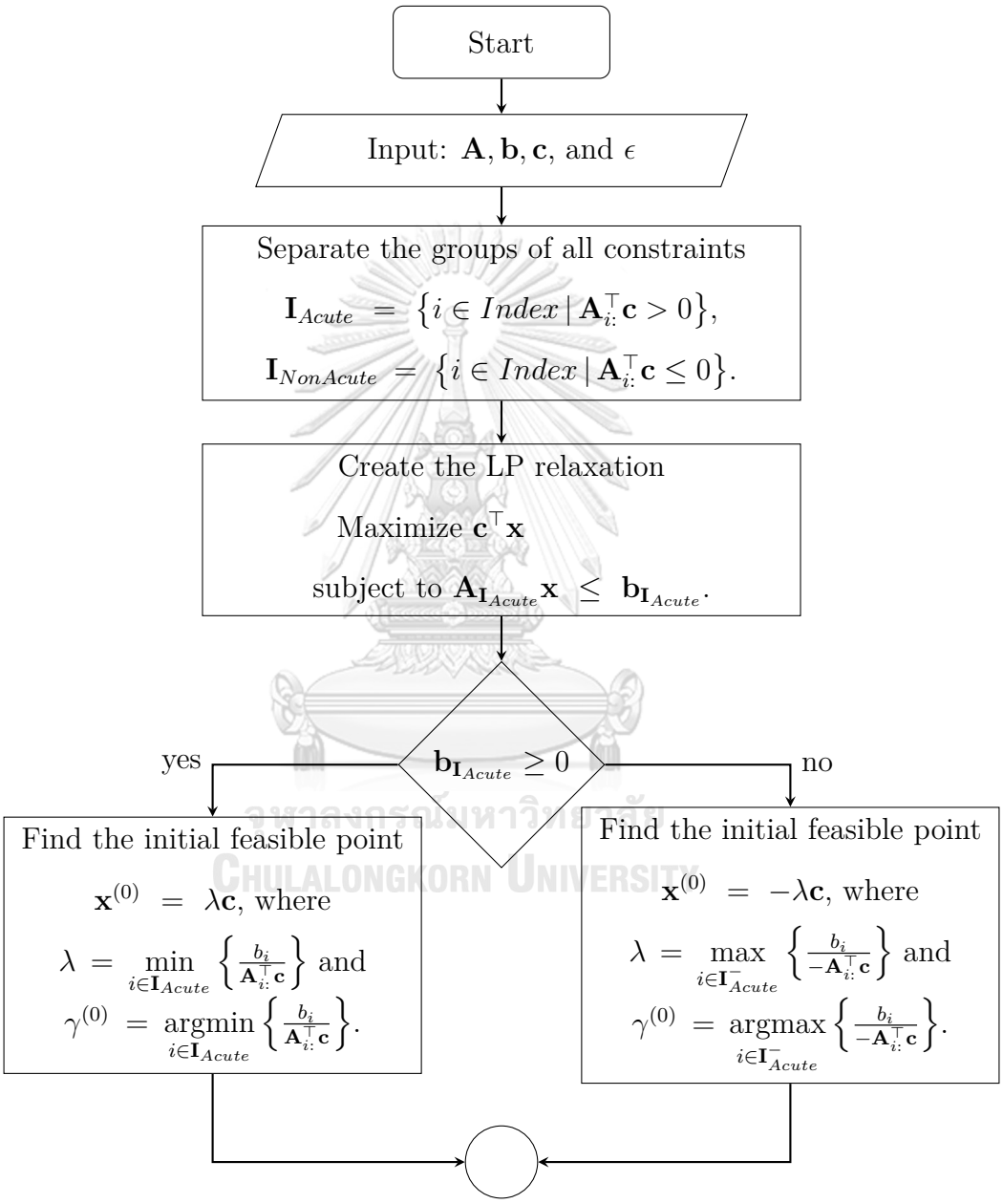


Figure 4.1: The flowchart of SAJS.

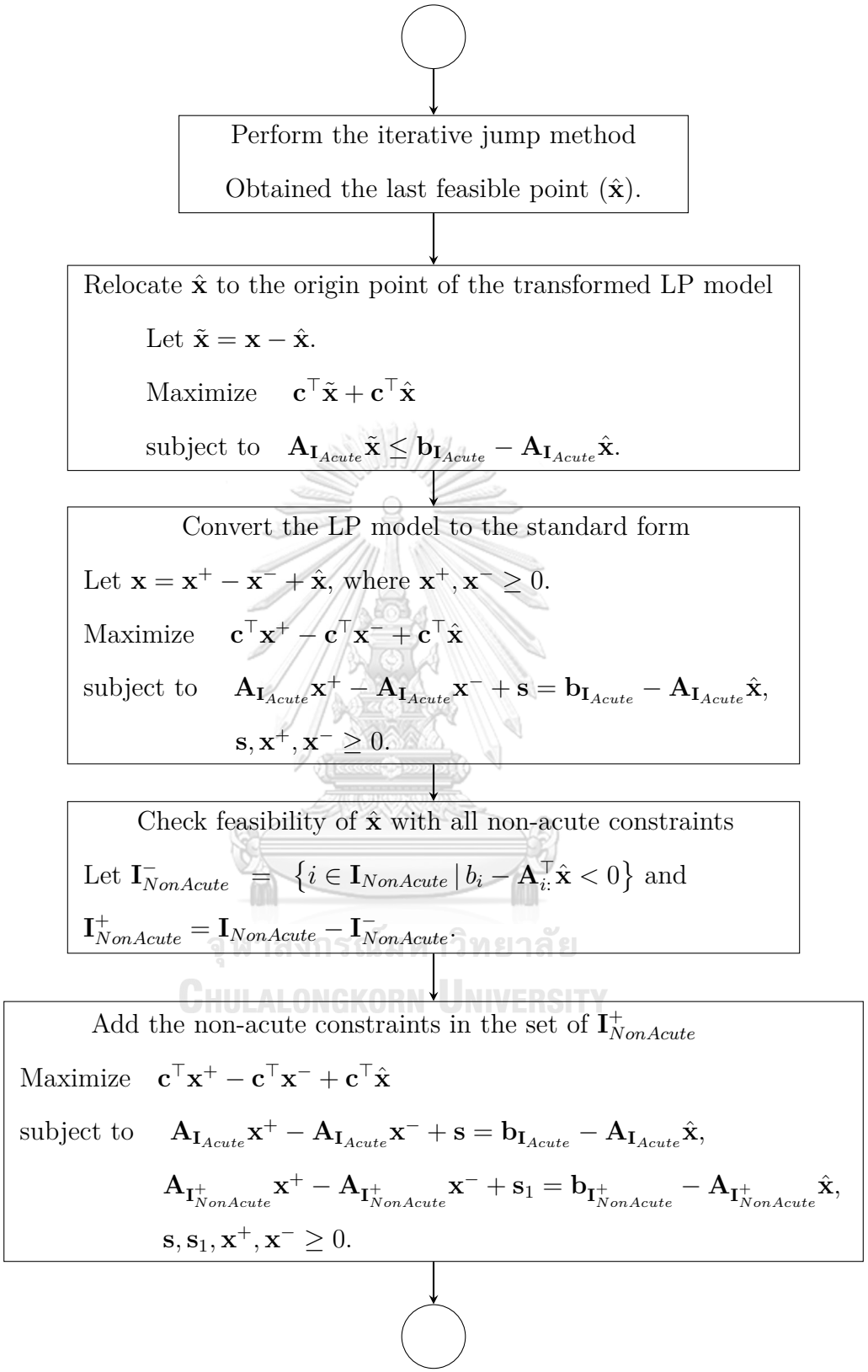


Figure 4.2: The flowchart of SAJS (Con.).

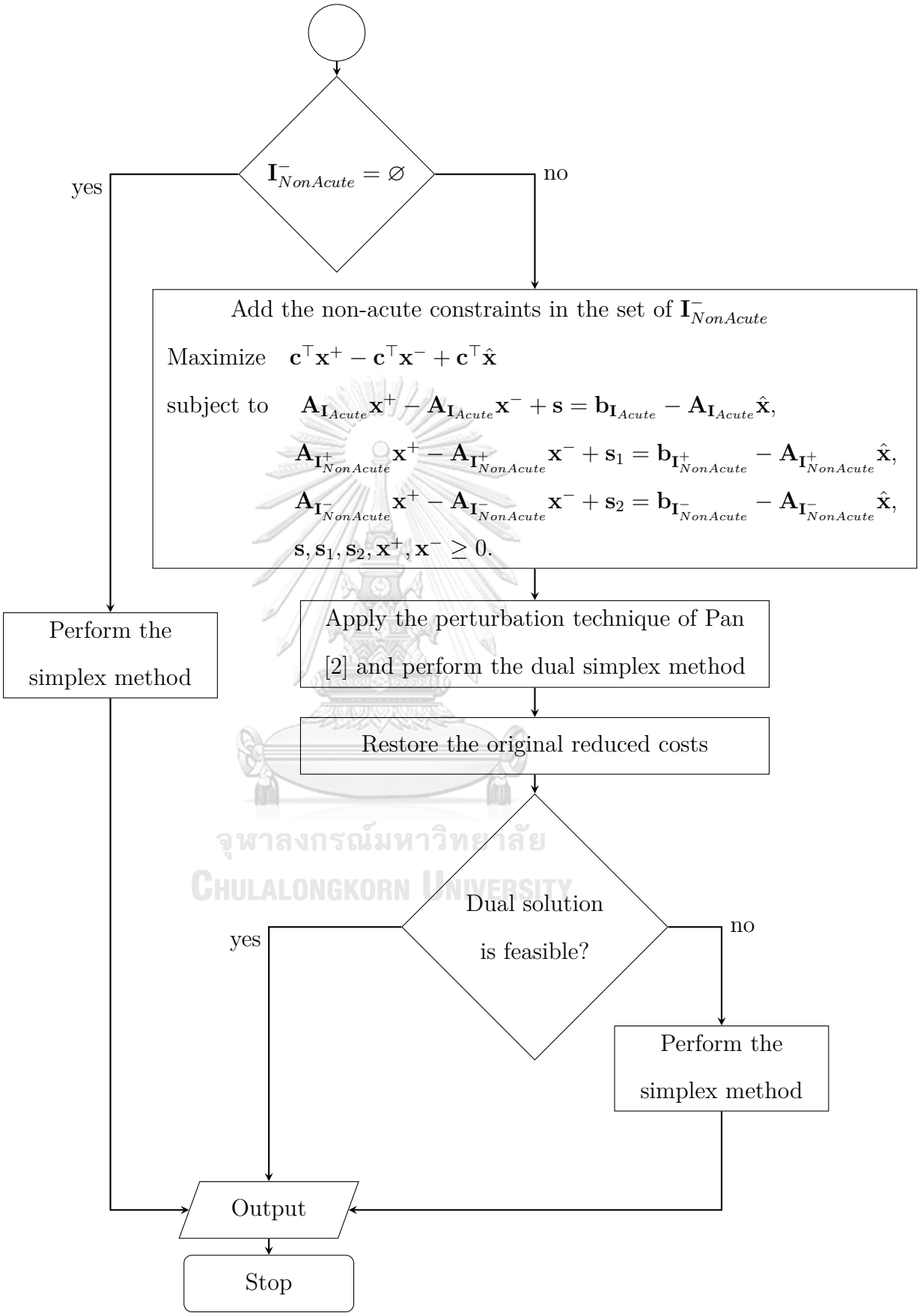


Figure 4.3: The flowchart of SAJS (Con.).

Example 4.1. Consider a linear programming model as follows:

$$\begin{aligned}
\text{Maximize} \quad & x_1 + x_2 \\
\text{subject to} \quad & x_1 + 5x_2 \leq 36, && (1) \\
& -2x_1 + 4x_2 \leq 3, && (2) \\
& 2x_1 - 3x_2 \leq 18, && (3) \\
& -x_1 + 5x_2 \leq 10, && (4) \\
& 9x_1 - 2x_2 \geq 15, && (5) \\
& x_2 \geq 1. && (6)
\end{aligned} \tag{4.10}$$

Transform the LP model in the following form.

$$\begin{aligned}
\text{Maximize} \quad & x_1 + x_2 \\
\text{subject to} \quad & x_1 + 5x_2 \leq 36, && (1) \\
& -2x_1 + 4x_2 \leq 3, && (2) \\
& 2x_1 - 3x_2 \leq 18, && (3) \\
& -x_1 + 5x_2 \leq 10, && (4) \\
& -9x_1 + 2x_2 \leq -15, && (5) \\
& -x_2 \leq -1. && (6)
\end{aligned} \tag{4.11}$$

Step 1: Separate constraints into two groups:

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

Thus, $\mathbf{I}_{Acute} = \{1,2,4\}$ and $\mathbf{I}_{NonAcute} = \{3,5,6\}$.

Step 2: Create the LP relaxation with the acute constraints.

The LP relaxation is created as follows:

$$\begin{aligned} \text{Maximize} \quad & x_1 + x_2 \\ \text{subject to} \quad & x_1 + 5x_2 \quad \leq 36, \qquad (1) \\ & -2x_1 + 4x_2 \quad \leq 3, \qquad (2) \\ & -x_1 + 5x_2 \quad \leq 10. \qquad (4) \end{aligned} \tag{4.12}$$

Step 3: Find the initial jump point of the iterative jump method.

Since $\mathbf{b}_{\mathbf{I}_{Acute}} = [36,3,10]^{\top} \geq \mathbf{0}$. Thus, λ is computed as follows:

$\lambda = \min\limits_{i \in \mathbf{I}_{Acute}} \left\{ \dfrac{b_i}{\mathbf{A}_{i:}^{\top}\mathbf{c}} \right\} = \min\limits_{i \in \{1,2,4\}} \left\{ \dfrac{36}{6}, \dfrac{3}{2}, \dfrac{10}{4} \right\} = 1.5.$

Therefore, $\mathbf{x}^{(0)} = \lambda\mathbf{c} = [1.5,1.5]^{\top}$ and $\gamma^{(0)} = 2$.

Step 4: Perform the iterative jump method by Algorithm 2.

Iteration 1:

Compute $\mathbf{v} = \dfrac{-\mathbf{A}_{2:}}{\|\mathbf{A}_{2:}\|} + \dfrac{\mathbf{c}}{\|\mathbf{c}\|} = \dfrac{-[-2,4]^{\top}}{\|[-2,4]^{\top}\|} + \dfrac{[1,1]^{\top}}{\|[1,1]^{\top}\|} = [1.1543, -0.1873]^{\top}$.

Constraint No. (i)	$\mathbf{A}_{i:}$	$b_{i:}$	$\mathbf{A}_{i:}^{\top}\mathbf{v}$	$b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(0)}$
1	$[1,5]^{\top}$	36	0.2178	27
4	$[-1,5]^{\top}$	10	-2.0908	4

Compute $\lambda = \min\limits_{i \in \mathbf{I}_{Acute} \setminus \{\gamma^{(0)}\}} \left\{ \dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(0)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^{\top}\mathbf{v} > 0 \right\} = \min\limits_{i \in \{1\}} \left\{ \dfrac{27}{0.2178} \right\} = 123.9669$
and $\gamma^{(1)} = \operatorname{argmin}\limits_{i \in \mathbf{I}_{Acute} \setminus \{\gamma^{(0)}\}} \left\{ \dfrac{b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(0)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^{\top}\mathbf{v} > 0 \right\} = 1.$

Thus, $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda\mathbf{v} = \begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix} + 123.9669 \begin{bmatrix} 1.1543 \\ -0.1873 \end{bmatrix} = \begin{bmatrix} 144.5949 \\ -21.7190 \end{bmatrix}$.

Iteration 2:

Compute $\mathbf{v} = \dfrac{-\mathbf{A}_{1:}}{\|\mathbf{A}_{1:}\|} + \dfrac{\mathbf{c}}{\|\mathbf{c}\|} = \dfrac{-[1,5]^{\top}}{\|[1,5]^{\top}\|} + \dfrac{[1,1]^{\top}}{\|[1,1]^{\top}\|} = [0.5109, -0.2734]^{\top}$.

Constraint No. (i)	$\mathbf{A}_{i:}$	$b_{i:}$	$\mathbf{A}_{i:}^{\top}\mathbf{v}$	$b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(1)}$
2	$[-2,4]^{\top}$	3	-2.1154	379.0658
4	$[-1,5]^{\top}$	10	-2.0908	263.1899

Since $\mathbf{A}_{i:}^{\top}\mathbf{v} < 0$ for all $i \in \{2,4\}$. So, the iterative jump method terminates with the last jump point $[\hat{x}_1, \hat{x}_2]^{\top} = [144.5949, -21.7190]^{\top}$ and the solution of the LP relaxation is unbounded.

Step 5: Relocate the last jump point $(\hat{\mathbf{x}})$ to the origin point of the new transformed LP model.

Let $\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix}$. Therefore, the transformed LP model is created as

follows:

$$\begin{array}{lll} \text{Maximize} & \tilde{x}_1 + \tilde{x}_2 + 122.8759 & \\ \text{subject to} & \tilde{x}_1 + 5\tilde{x}_2 \leq 0, & (1) \\ & -2\tilde{x}_1 + 4\tilde{x}_2 \leq 379.0658, & (2) \\ & -\tilde{x}_1 + 5\tilde{x}_2 \leq 263.1899, & (4) \end{array} \tag{4.13}$$

Step 6: Add the slack variables into the transformed LP model.

$$\begin{array}{lll} \text{Maximize} & \tilde{x}_1 + \tilde{x}_2 + 122.8759 & \\ \text{subject to} & \tilde{x}_1 + 5\tilde{x}_2 + s_1 = 0, & (1) \\ & -2\tilde{x}_1 + 4\tilde{x}_2 + s_2 = 379.0658, & (2) \\ & -\tilde{x}_1 + 5\tilde{x}_2 + s_3 = 263.1899, & (4) \\ & s_1, s_2, s_3 \geq 0. & \end{array} \tag{4.14}$$

The current transformed LP model is an unrestricted variable LP model.

Let $\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} = \begin{bmatrix} x_1^+ \\ x_2^+ \end{bmatrix} - \begin{bmatrix} x_1^- \\ x_2^- \end{bmatrix}$ where $\begin{bmatrix} x_1^+ \\ x_2^+ \end{bmatrix}, \begin{bmatrix} x_1^- \\ x_2^- \end{bmatrix} \geq \mathbf{0}$.

$$\begin{array}{lll} \text{Maximize} & x_1^+ - x_1^- + x_2^+ - x_2^- + 122.8759 & \\ \text{subject to} & x_1^+ - x_1^- + 5x_2^+ - 5x_2^- + s_1 = 0, & (1) \\ & -2x_1^+ + 2x_1^- + 4x_2^+ - 4x_2^- + s_2 = 379.0658, & (2) \\ & -x_1^+ + x_1^- + 5x_2^+ - 5x_2^- + s_3 = 263.1899, & (4) \\ & x_1^+, x_1^-, x_2^+, x_2^-, s_1, s_2, s_3 \geq 0. & \end{array} \tag{4.15}$$

Step 7: Create the initial simplex tableau as follows:

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	RHS
z	-1	1	-1	1	0	0	0	122.8759
s_1	1	-1	5	-5	1	0	0	0
s_2	-2	2	4	-4	0	1	0	379.0658
s_3	-1	1	5	-5	0	0	1	263.1899

Step 8: Reinsert the non-acute constraints satisfy the current solution.

Constraint No. (i)	$\mathbf{A}_{i:}$	$b_{i:}$	$b_i - \mathbf{A}_{i:}^\top \mathbf{x}^{(0)}$
3	$[2, -3]^\top$	18	-336.3468
5	$[-9, 2]^\top$	-15	1329.7921
6	$[0, -1]^\top$	-1	-22.7190

Let $\mathbf{I}^-_{NonAcute} = \{i \in \mathbf{I}_{NonAcute} \mid b_i - \mathbf{A}_{i:}^\top \hat{\mathbf{x}} < 0\} = \{3, 6\}$

and $\mathbf{I}^+_{NonAcute} = \{i \in \mathbf{I}_{NonAcute} \mid b_i - \mathbf{A}_{i:}^\top \hat{\mathbf{x}} \geq 0\} = \{5\}$.

The constraints in the set of $\mathbf{I}_{NonAucte^+}$ are reinserted into the initial simplex tableau.

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	s_4	RHS
z	-1	1	-1	1	0	0	0	0	122.8759
s_1	1	-1	5	-5	1	0	0	0	0
s_2	-2	2	4	-4	0	1	0	0	379.0658
s_3	-1	1	5	-5	0	0	1	0	263.1899
s_4	-9	9	2	-2	0	0	0	1	1329.7921

Step 9: Reinsert the constraints in the set of $\mathbf{I}^{-}_{NonAcute}$ into the simplex tableau.

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	-1	1	-1	1	0	0	0	0	0	0	122.8759
s_1	1	-1	5	-5	1	0	0	0	0	0	0
s_2	-2	2	4	-4	0	1	0	0	0	0	379.0658
s_3	-1	1	5	-5	0	0	1	0	0	0	263.1899
s_4	-9	9	2	-2	0	0	0	1	0	0	1329.7921
s_5	2	-2	-3	3	0	0	0	0	1	0	-336.3468
s_6	0	0	-1	1	0	0	0	0	0	1	-22.719

Step 10: Perturb the reduced costs of the simplex tableau to make the dual solution feasible and perform the dual simplex method.

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	-1	-1	-1	-1	0	0	0	0	0	0	122.8759
s_1	1	-1	5	-5	1	0	0	0	0	0	0
s_2	-2	2	4	-4	0	1	0	0	0	0	379.0658
s_3	-1	1	5	-5	0	0	1	0	0	0	263.1899
s_4	-9	9	2	-2	0	0	0	1	0	0	1329.7921
s_5	2	-2	$\mathbf{-3}$	3	0	0	0	0	1	0	-336.3468
s_6	0	0	-1	1	0	0	0	0	0	1	-22.719

By the dual simplex method, x_2^+ is the entering variable and s_5 is the leaving

variable. After pivoting, the updated simplex tableau can be written as follows:

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	-1.6667	-0.3333	0	-2	0	0	0	0	-0.3333	0	235
s_1	4.3333	$\mathbf{-4.3333}$	0	0	1	0	0	0	1.6667	0	-560.5714
s_2	0.6667	-0.6667	0	0	0	1	0	0	1.3333	0	-69.4
s_3	2.3333	-2.3333	0	0	0	0	1	0	1.6667	0	-297.4
s_4	-7.6667	7.6667	0	0	0	0	0	1	0.6667	0	1105.5556
x_2^+	-0.6667	0.6667	1	-1	0	0	0	0	-0.3333	0	112.1111
s_6	-0.6667	0.6667	0	0	0	0	0	0	-0.3333	1	89.4

The entering variable is x_1^- and the leaving variable is s_1. After pivoting, the simplex tableau is represented as follows:

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	-2	0	0	-2	0	0	0	0	-0.5	0	278.1111
x_1^-	-1	1	0	0	-0.25	0	0	0	-0.375	0	129.375
s_2	0	0	0	0	-0.1429	1	0	0	1	0	16.8571
s_3	0	0	0	0	-0.5	0	1	0	0.7778	0	4.5
s_4	0	0	0	0	1.7778	0	0	1	3.6	0	113.75
x_2^+	0	0	1	-1	0.1667	0	0	0	0	0	25.875
s_6	0	0	0	0	0.1667	0	0	0	0	1	3.1429

For the current simplex tableau, it is the optimal tableau. Thus, the original

reduced cost is reinserted to the simplex tableau.

	x_1^+	x_1^-	x_2^+	x_2^-	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	0	0	0	0	0.375	0	0	0	0.3333	0	19.375
x_1^-	-1	1	0	0	-0.25	0	0	0	-0.375	0	129.375
s_2	0	0	0	0	-0.1429	1	0	0	1	0	16.8571
s_3	0	0	0	0	-0.5	0	1	0	0.7778	0	4.5
s_4	0	0	0	0	1.7778	0	0	1	3.6	0	113.75
x_2^+	0	0	1	-1	0.1667	0	0	0	0	0	25.875
s_6	0	0	0	0	0.1667	0	0	0	0	1	3.1429

Since this simplex tableau is the optimal tableau, the solution is restored to the solution of the original LP model as follows:

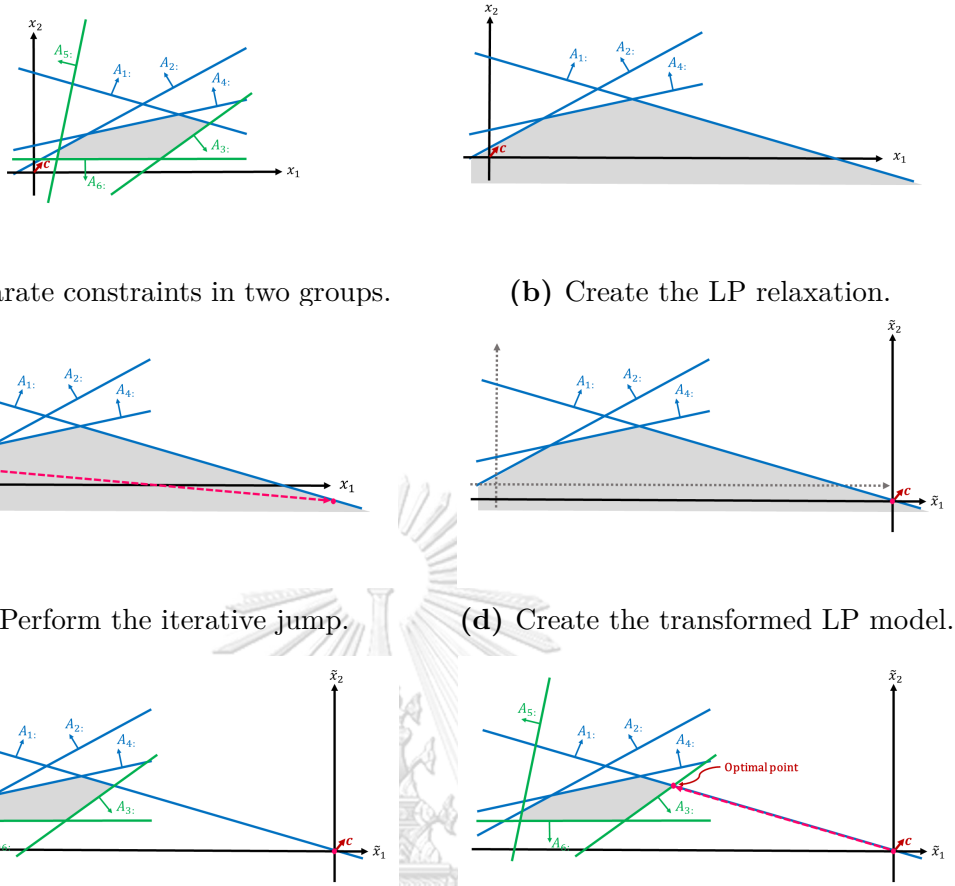
$$\begin{aligned}[\tilde{x}_1, \tilde{x}_2]^\top &= [x_1^+, x_2^+]^\top - [x_1^-, x_2^-]^\top \\ &= [0, 207/8]^\top - [1035/8, 0]^\top \\ &= [-129.3750, 25.8750]^\top.\end{aligned}$$

Thus, the solution is

$$\begin{aligned}[x_1, x_2]^\top &= [\tilde{x}_1, \tilde{x}_2]^\top + [\hat{x}_1, \hat{x}_2]^\top \\ &= [-129.3750, 25.8750]^\top + [144.5949, -21.7190]^\top \\ &= [15.2308, 4.1538]^\top.\end{aligned}$$

with the objective value is 19.3846.

The geometric views of Example 4.10 are shown in Figure 4.4(a)-4.4(f).



(a) Separate constraints in two groups.

(b) Create the LP relaxation.

(c) Perform the iterative jump.

(d) Create the transformed LP model.

(e) Reinsert the non-acute constraints.

(f) Perform the dual simplex method.

Figure 4.4: Geometric views of SAJS performing on Example 4.1.

4.3 Results and experiments

In this section, the problems used to test the effectiveness of the self-regulating artificial-free linear programming solver using a jump and simplex method (SAJS) are explained. The tested problems are divided into two collections: the randomly generated problems and the Netlib problems. After that, the computational results of the wall-clock time of SAJS, SNAR, and the two-phase simplex method (TP) are shown and summarized. The experiments were operated by an Intel(R) Core(TM) i7-3770 CPU@ 3.40 GHz processor with 8 GB RAM on Windows 10. All methods were written by

NumPy library in Python.

4.3.1 The randomly generated problem

The maximized LP model is constructed as follows:

- the number of constraints (m) is higher than the number of variables (n),
- the objective vector ($\mathbf{c}$) is generated as the vector of ones,
- the generated rows of $\mathbf{A}$ are divided into two groups:
  1. for $i = 1, 2, ..., t$, where $t = \left\lceil \frac{3}{4}n \right\rceil$, all coefficients in this group of $\mathbf{A}$ are uniformly random from $[-3, 9]$,
  2. for $i = t+1, t+2, ..., m$, all coefficients in this group of $\mathbf{A}$ are uniformly random from $[-9, 3]$,

  Note: For rows in the first group of $\mathbf{A}$, it has a high probability that the created constraint makes an acute angle with the objective vector. Meanwhile, the second group has a high probability that the created constraint makes an obtuse angle with the objective vector.
- the right-hand side vector is created as follows:

  first, $\mathbf{x}$ where $x_j \in [-9, 9]$, $j = 1, 2, ..., n$ is generated to guarantee a nonempty feasible region. After that, the right-hand side vector is established as $b_i = \mathbf{A}_{i:}^{\top}\mathbf{x}$ where $i = 1, 2, ..., n$ and $b_i = \mathbf{A}_{i:}^{\top}\mathbf{x} + 1$ where $i = n+1, n+2, ..., m$.

The different sizes of m and n with the randomly generated problems are defined as following. Let $m > n$, $m \in \{100, 200, 300, 400, 500, 1000, 2000\}$ and $n \in \left\{ \frac{m}{10}, \frac{2m}{10}, \frac{3m}{10}, \frac{4m}{10} \right\}$. For each size of the randomly generated, the wall-clock times are averaged from 10 LP problems.

4.3.2 The Netlib problem

The Netlib LP test sets used to test the effectiveness of all methods are in the Mathematical Programming System (MPS) format. Each section of the MPS file is separated into a header which consists of a single word. Types of the headers are defined as follows:

Table 4.1: Description of the header in the MPS file.

Header	Description of each header
NAME	The name of the LP problem.
ROWS	The type and name of each constraint.
COLUMNS	The name of each variable including the coefficient of the constraints and the objective function.
RHS	The name of the right-hand side vector including the values of each constraint.
ENDDATA	The end of reading of the MPS file.

In the section of ROWS, the type of constraints are defined by a single letter as

- N : the objective function
- G : the greater than or equal to constraint
- L : the less than or equal to constraint
- E : the equality constraint.

In the header of COLUMNS and ROWS, the non-zero value of coefficients of the objective function and constraints are represented in the MPS file.

The next demonstrated example will show how the LP model in the MPS format is translated to the traditional LP model.

```
NAME          Example4.1
ROWS
 N  OBJ
 G  R1
 E  R2
 L  R3
 L  R4
 L  R5
 G  R6
COLUMNS
    x1   R2        3
    x1   R3       -3
    x1   R4        2
    x1   OBJ      -3
    x2   R1        2
    x2   R2       -3
    x2   R3        2
    x2   OBJ       4
    x3   R1        3
    x3   R2        6
```

```
    x3        R4      -1
    x3        R5       1
    x3        OBJ     -5
    x4        R2      -1
    x4        R3      -4
    x4        R4       4
    x4        R5       1
    x4        OBJ      3
RHS
    RHS1      R1       8
    RHS1      R2      20
    RHS1      R3      -6
    RHS1      R4      18
    RHS1      R5      10
    RHS1      R6       2
ENDDATA
```


The equivalent LP model of the MPS file is shown in the following.

Example 4.2.

$$\begin{array}{lll} \text{OBJ}: \text{ Minimize} & -3x_1 + 4x_2 - 5x_3 + 3x_4 & \\ \text{R1}: \text{ subject to} & 2x_2 + 3x_3 & \geq 8, \\ \text{R2}: & 3x_1 - 3x_2 + 6x_3 - x_4 & = 20, \\ \text{R3}: & -3x_1 + 2x_2 - 4x_4 & \leq -6, \\ \text{R4}: & 2x_1 - x_3 + 4x_4 & \leq 18, \\ \text{R5}: & x_3 & \leq 10, \\ \text{R6}: & x_4 & \geq 2, \\ & x_1, x_2, x_3, x_4 & \geq 0. \end{array} \tag{4.16}$$

Since the format LP model used in this dissertation corresponds with LP (4.5). Thus, the equality constraint

$$3x_1 - 3x_2 + 6x_3 - x_4 = 20 \tag{4.17}$$

is transformed as

$$x_1 = \frac{20 + 3x_2 - 6x_3 + x_4}{3} \tag{4.18}$$

After that, for all constraints of LP (4.16), x_1 is represented by $\frac{20 + 3x_2 - 6x_3 + x_4}{3}$. Thus, the LP model is written as follows:

$$
\begin{aligned}
\text{Minimize} \quad & -3x_2 + 6x_3 - x_4 - 20 \\
\text{subject to} \quad & 2x_2 + 3x_3 && \geq 8, \\
& -x_2 + 6x_3 - 5x_4 && \leq 14, \\
& 2x_2 - 5x_3 + \frac{14}{3}x_4 && \leq \frac{14}{3}, \\
& x_3 && \leq 10, \\
& x_4 && \geq 2, \\
& 3x_2 - 6x_3 + x_4 && \geq -20, \\
& x_2, x_3, x_4 && \geq 0.
\end{aligned} \tag{4.19}
$$

Next, the greater than or equal to constraints will be multiplied by -1 to convert them into the less than or equal to constraints.

$$
\begin{aligned}
\text{Minimize} \quad & -3x_2 + 6x_3 - x_4 - 20 \\
\text{subject to} \quad & -2x_2 - 3x_3 && \leq -8, \\
& -x_2 + 6x_3 - 5x_4 && \leq 14, \\
& 2x_2 - 5x_3 + \frac{14}{3}x_4 && \leq \frac{14}{3}, \\
& x_3 && \leq 10, \\
& -x_4 && \leq -2, \\
& -3x_2 + 6x_3 - x_4 && \leq 20, \\
& -x_2 && \leq 0, \\
& -x_3 && \leq 0, \\
& -x_4 && \leq 0.
\end{aligned} \tag{4.20}
$$

4.3.3 Computational result

The iterative jump method will terminate when it satisfied the stopping criterion. The stopping criterion introduced in this dissertation is the stopping criterion involved the

improvement of the objective values having the stopping parameter, ε, which is defined as the least ratio improvement of two consecutive differences of the objective values. Thus, for each i^{th} jump, if $\frac{\mathbf{c}^\top \mathbf{x}_{i+2} - \mathbf{c}^\top \mathbf{x}_{i+1}}{\mathbf{c}^\top \mathbf{x}_{i+1} - \mathbf{c}^\top \mathbf{x}_i}$ is greater than ε then the iterative jump method will continue. The best ε will be the one that gives the fastest average wall-clock times of SAJS which are tested and are shown in Table 4.2.

Table 4.2: SAJS performances varying ε.

Row	Col	The average wall-clock time of SAJS (sec.)				
		ε=0.20	ε=0.30	ε=0.40	ε=0.50	ε=0.60
100	10	0.0156 (±0.0079)	0.0112 (±0.0085)	**0.0087** (±0.0078)	0.0237 (±0.0245)	0.0294 (±0.0340)
	20	0.0353 (±0.0214)	**0.0293** (±0.0132)	0.0393 (±0.0203)	0.0359 (±0.0171)	0.0372 (±0.0139)
	30	**0.0517** (±0.0167)	0.0646 (±0.0162)	0.0621 (±0.0260)	0.0734 (±0.0342)	0.0702 (±0.0328)
	40	0.1023 (±0.0592)	0.0751 (±0.0227)	0.0814 (±0.0197)	**0.0716** (±0.0202)	0.0789 (±0.0295)
200	20	0.0852 (±0.0357)	0.0716 (±0.0244)	**0.0653** (±0.0188)	0.0825 (±0.0268)	0.0728 (±0.0126)
	40	**0.1738** (±0.0360)	0.2257 (±0.0608)	0.2539 (±0.0627)	0.2575 (±0.0601)	0.2276 (±0.0434)
	60	0.4161 (±0.0919)	0.3907 (±0.1089)	**0.3455** (±0.0720)	0.3925 (±0.0949)	0.4129 (±0.0823)
	80	0.5384 (±0.0961)	0.5634 (±0.1182)	0.5604 (±0.1117)	**0.5037** (±0.1514)	0.5707 (±0.0707)
300	30	**0.2441** (±0.0845)	0.2647 (±0.0879)	0.2937 (±0.0982)	0.3070 (±0.0921)	0.2505 (±0.0692)
	60	0.7219 (±0.1121)	0.8509 (±0.0983)	0.7412 (±0.1450)	0.8312 (±0.0815)	0.7731 (±0.1108)
	90	1.1185 (±0.1387)	1.0895 (±0.1556)	1.0973 (±0.1313)	1.0587 (±0.1574)	**1.0333** (±0.1558)
	120	1.7548 (±0.2300)	1.6880 (±0.2283)	1.7038 (±0.2158)	**1.6714** (±0.1753)	1.7073 (±0.2093)

Continued on the next page

Table 4.2 – *Continued from the previous page*

Row	Col	The average wall-clock time of SAJS (sec.)				
		ε=0.20	ε=0.30	ε=0.40	ε=0.50	ε=0.60
400	40	0.6008 (±0.1032)	**0.5355** (±0.0953)	0.5505 (±0.1447)	0.5381 (±0.0593)	0.5634 (±0.1350)
	80	1.6020 (±0.2390)	1.5954 (±0.2490)	**1.5347** (±0.2348)	1.6173 (±0.2130)	1.6007 (±0.2099)
	120	2.9433 (±0.3600)	2.8127 (±0.3218)	2.8303 (±0.3240)	**2.7863** (±0.3223)	2.9393 (±0.5035)
	160	4.2761 (±0.6735)	**4.2352** (±0.5976)	4.3220 (±0.6476)	4.3217 (±0.5776)	4.2812 (±0.5769)
500	50	1.1332 (±0.1485)	1.1474 (±0.1681)	**1.1290** (±0.1874)	1.2073 (±0.1734)	1.2129 (±0.1870)
	100	3.3854 (±0.2225)	3.3023 (±0.2934)	3.2889 (±0.2554)	3.2297 (±0.3564)	**3.1328** (±0.3320)
	150	5.4433 (±0.5061)	**5.2168** (±0.5792)	5.4445 (±0.3721)	5.3581 (±0.3541)	5.4255 (±0.4713)
	200	8.0838 (±1.0386)	8.1042 (±1.1673)	**7.8459** (±1.1333)	8.0047 (±1.3826)	7.9496 (±1.4953)
Average		1.6363	1.6137	**1.6099**	1.6186	1.6185

In Table 4.2, it demonstrates the average wall-clock time (in seconds) varying $\varepsilon = 0.20, 0.30, 0.40, 0.50, 0.60$ of SAJS. For Table 4.2, Row represents the number of constraints in the LP model, Col represents the number of variables in the LP model, the boldface numbers identify the smallest average wall-clock time and the number in parenthesis represents the standard deviations of each size of the LP model. Both methods are tested with the different sizes of the LP models as $m = \{100, 200, 300, 400, 500\}$ and $n = \left\{\frac{m}{10}, \frac{2m}{10}, \frac{3m}{10}, \frac{4m}{10}\right\}$. The results of the average wall-clock times in SAJS for each size problem are not significantly different in each ε. However, SAJS with $\varepsilon = 0.40$ takes the least total average wall-clock time. Therefore, for this dissertation, SAJS uses $\varepsilon = 0.40$ to compare with TP and SNAR with 280 randomly generated problems presented in Table 4.3.

Table 4.3: The comparison of the average wall-clock time of SAJS with ε=0.40, TP, and SNAR on randomly generated linear programming problems.

Row	Col	SAJS with ε=0.40	TP	SNAR
100	10	**0.0087** (±0.0078)	0.0545 (±0.0391)	0.0328 (±0.0295)
	20	**0.0393** (±0.0203)	0.0754 (±0.0254)	0.0811 (±0.0350)
	30	**0.0621** (±0.0260)	0.0859 (±0.0339)	0.1769 (±0.0778)
	40	**0.0814** (±0.0197)	0.1232 (±0.0413)	0.3361 (±0.1292)
200	20	**0.0653** (±0.0188)	0.2587 (±0.1502)	0.1060 (±0.0382)
	40	**0.2539** (±0.0627)	0.4503 (±0.1196)	0.4667 (±0.1568)
	60	**0.3455** (±0.0720)	0.4563 (±0.0778)	1.4911 (±0.4156)
	80	**0.5604** (±0.1117)	0.5722 (±0.1070)	2.9059 (±0.5067)
300	30	**0.2937** (±0.0982)	1.0987 (±0.2552)	0.4030 (±0.2304)
	60	**0.7412** (±0.1450)	1.5624 (±0.2246)	1.5629 (±0.4752)
	90	**1.0973** (±0.1313)	1.8698 (±0.4564)	6.4747 (±2.3881)
	120	**1.7038** (±0.2158)	2.5036 (±0.3773)	16.8580 (±5.4162)
400	40	**0.5505** (±0.1447)	1.8357 (±0.7608)	0.8795 (±0.2560)

Continued on the next page

Table 4.3 – *Continued from the previous page*

Row	Col	SAJS with ε=0.40	TP	SNAR
400	80	**1.5347** (±0.2348)	3.9229 (±1.0108)	4.5473 (±1.0408)
	120	**2.8303** (±0.3240)	4.2499 (±0.6664)	31.1020 (±16.5569)
	160	**4.3220** (±0.6476)	6.3061 (±1.0764)	96.1301 (±14.6135)
500	50	**1.1290** (±0.1874)	4.2233 (±1.6131)	1.7442 (±0.2062)
	100	**3.2889** (±0.2554)	6.0040 (±1.4478)	10.1424 (±4.4312)
	150	**5.4445** (±0.3721)	8.4096 (±1.2648)	88.5504 (±43.5178)
	200	**7.8459** (±1.1333)	11.9502 (±1.3608)	248.8169 (±35.2017)
1000	100	**9.6681** (±1.0846)	41.8906 (±16.9485)	13.5965 (±2.3317)
	200	**30.5525** (±4.3661)	67.8041 (±19.6872)	345.9712 (±76.8162)
	300	**58.5633** (±4.5964)	100.1301 (±18.2994)	1,325.4713 (±355.2482)
	400	**86.3726** (±14.0708)	119.9710 (±12.2241)	4,005.7372 (±510.1724)
2000	200	**98.4510** (±10.9009)	490.0733 (±129.8028)	255.0905 (±60.6867)
	400	**356.8204** (±40.8287)	628.8453 (±171.6103)	4,675.7620 (±1419.6281)
	600	**732.6358** (±83.2131)	1,020.2260 (±161.6943)	24,843.2105 (±12659.5754)
	800	**983.3027** (±72.5009)	1,322.7826 (±140.8523)	54,348.7565 (±9224.8811)
Average		**85.3059**	137.4191	3225.9430

In Table 4.3, Row represents the number of constraints in the LP model, Col represents the number of variables in the LP model, the boldface numbers identify the smallest average wall-clock time and the number in parenthesis represents the standard deviations of each size of the LP model. Table 4.3 shows that the wall-clock time of SAJS with $\varepsilon = 0.40$ outperforms TP and SNAR for all randomly generated LP models. Since artificial variables are added to the LP model for the TP method. Thus, its size is expanded which affects the solution time. While SNAR reinserts the non-acute constraint one by one into the LP relaxation when the solution of the LP relaxation is unbounded. As a

result, it requires longer computational time to find a solution. Furthermore, Figure 4.5 to Figure 4.11 present the comparison of the wall-clock time of SAJS with $\varepsilon = 0.40$, TP and SNAR for each size of the number of constraints from $\{100, 200, 300, 400, 500, 1000, 2000\}$.

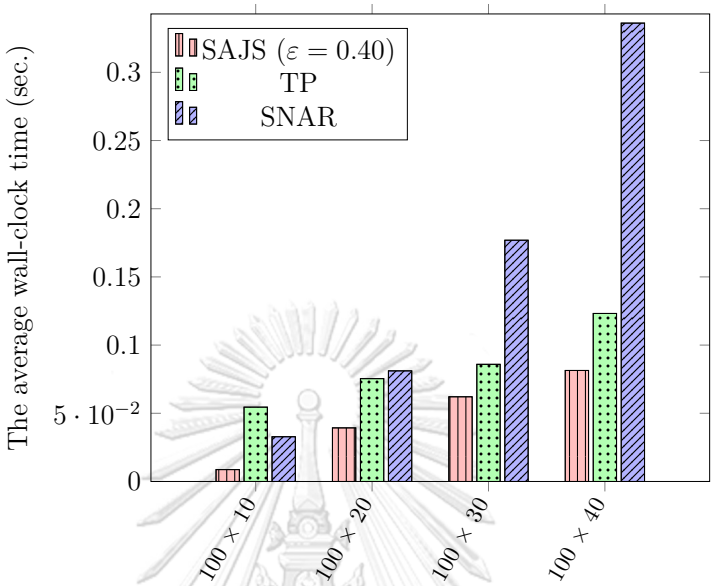


Figure 4.5: The comparison of SAJS, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 100 constraints.

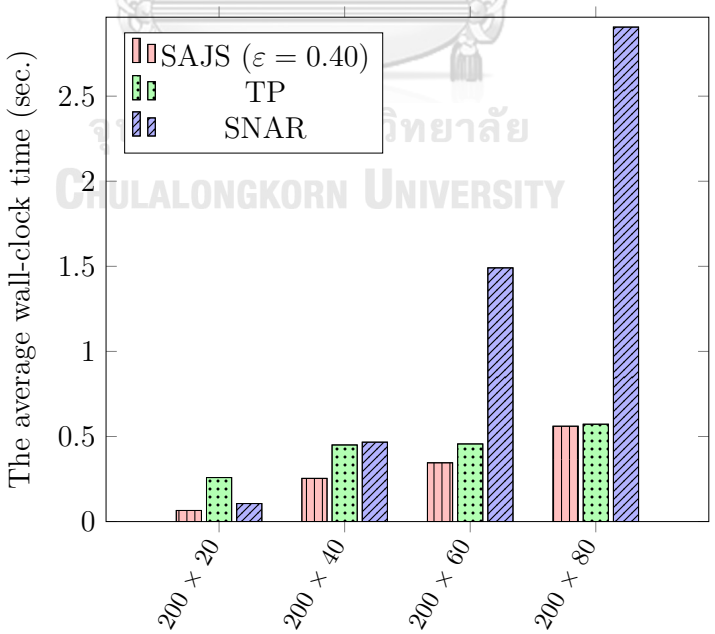


Figure 4.6: The comparison of SAJS, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 200 constraints.

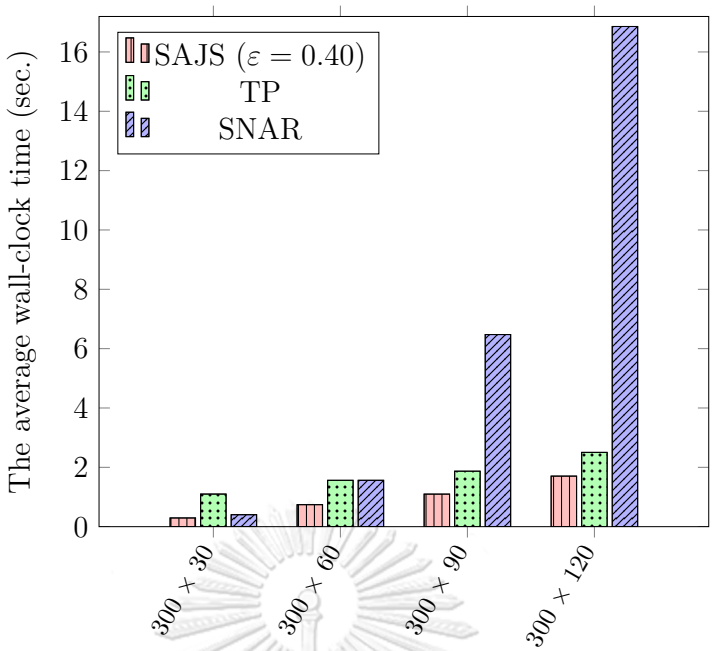


Figure 4.7: The comparison of SAJS, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 300 constraints.

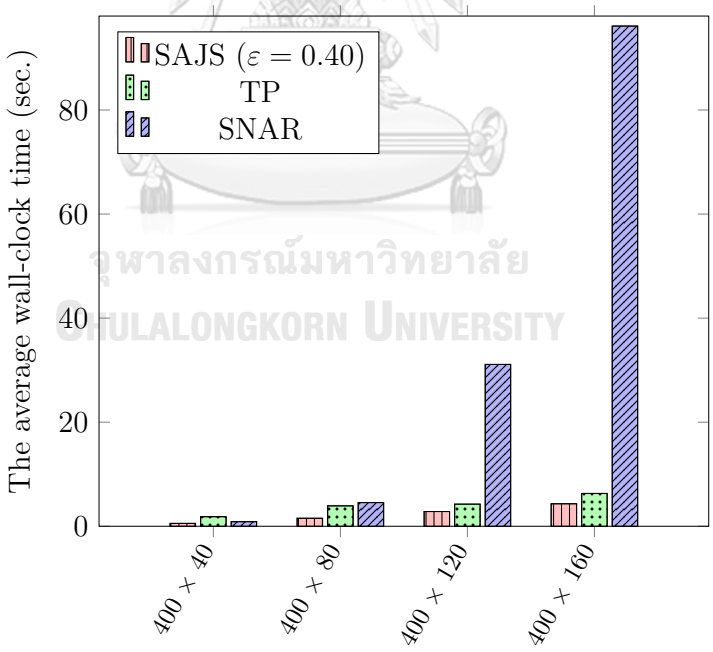


Figure 4.8: The comparison of SAJS, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 400 constraints.

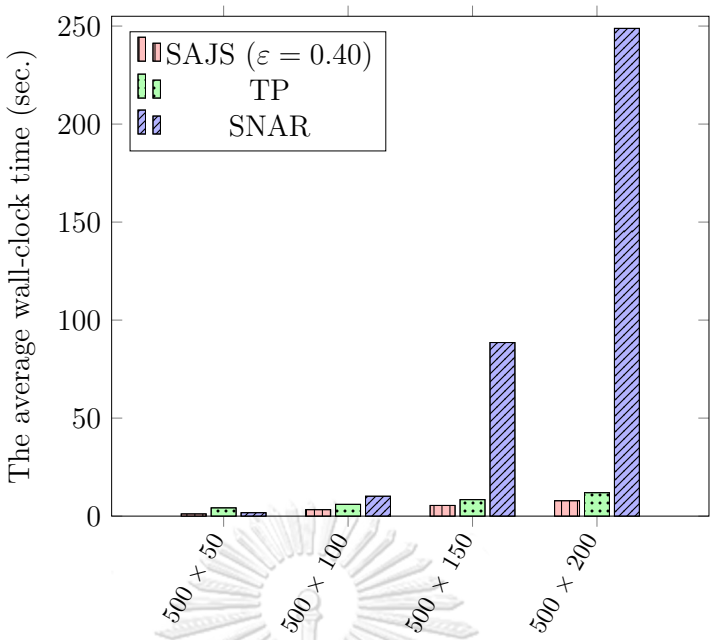


Figure 4.9: The comparison of SAJS, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 500 constraints.

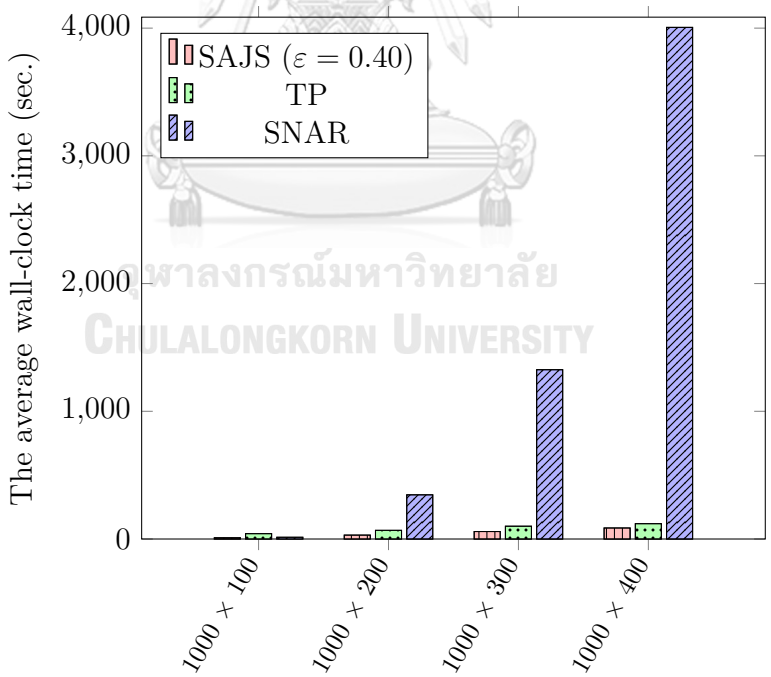


Figure 4.10: The comparison of SAJS, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 1000 constraints.

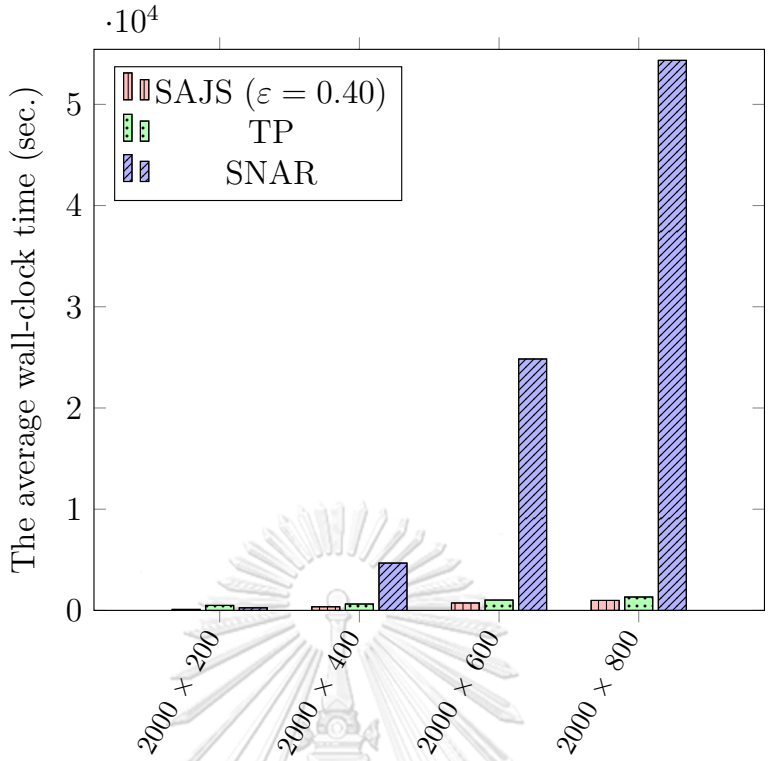


Figure 4.11: The comparison of SAJS, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 2000 constraints.

From Figure 4.5 - Figure 4.11, it shows that SNAR has poor performance more than other methods especially the LP model with a large number of variables. While other methods provide similar performance.

Table 4.4: The comparison of the average wall-clock time of SAJS with $\varepsilon = 0.40$, TP, and SNAR on Netlib problems.

	#Acute	#NonAcute	Row	Col	SAJS with ε=0.40	TP	SNAR
AFIRO	13	38	51	24	**0.0070**	0.0090	0.0160
SC50B	10	68	78	28	**0.0090**	0.0137	0.0269
SC50A	5	73	78	28	0.0156	**0.0156**	0.0312
BLEND	67	71	138	82	0.1007	0.2234	**0.0625**
STOCFOR1	42	72	114	40	**0.0625**	0.1189	0.0937
SC105	63	99	162	66	0.3226	0.2323	**0.2290**
SCAGR7	10	153	163	58	**0.0439**	0.0954	0.3854
SHARE2B	75	90	165	48	**0.0828**	0.1526	0.3788
ADLITTLE	68	117	185	56	0.3255	**0.1983**	0.4397
SHARE1B	145	171	316	142	**1.7278**	2.5567	3.1531
SC205	84	169	253	136	**0.9219**	1.8421	1.5883
BEACONFD	19	298	317	112	**0.4648**	0.6144	2.6241
BRANDY	135	160	295	122	0.4721	**0.4358**	2.1742

Continued on the next page

Table 4.4 – *Continued from the previous page*

	#Acute	#NonAcute	Row	Col	SAJS with ε=0.40	TP	SNAR
ISRAEL	124	242	366	213	**1.6921**	2.5204	8.0066
SCORPION	7	323	330	110	**1.6356**	2.4394	15.1751
LOTFI	454	306	760	683	**13.0909**	57.3044	49.6100
AGG	240	232	472	249	**5.6064**	8.7660	165.9386
BANDM	361	254	615	127	**0.9824**	1.5254	105.5491
E226	276	220	496	108	**0.4608**	1.1479	137.5253
SCAGR25	219	253	472	167	**2.1669**	6.9685	290.3034
SCFXM1	463	197	660	360	**6.1334**	8.3432	320.4382
AGG2	210	391	601	271	**4.1853**	8.6766	444.9718
AGG3	513	245	758	242	**2.7424**	5.5820	437.4691
SCTAP1	445	313	758	242	**3.3793**	6.7133	9.5161
DEGEN2	350	409	759	315	**15.3564**	31.5566	333.8480
SCSD1	285	386	671	200	**3.9966**	10.0661	916.5727

Continued on the next page

Table 4.4 – *Continued from the previous page*

	#Acute	#NonAcute	Row	Col	SAJS with ε=0.40	TP	SNAR
SCFXM2	808	542	1350	1203	**76.3067**	308.8540	4199.4493
SCRS8	764	511	1275	785	**36.3433**	109.6057	1979.8150
SCFXM3	753	795	1548	1146	**52.6607**	157.6138	NA
SCSD6	400	802	1202	542	**30.2915**	95.9760	NA
SHIP04S	1111	1097	2208	1806	**159.7317**	562.2637	16490.8526
25FV47	1722	1028	2750	2353	**506.4883**	2391.8577	NA
STOCFOR2	724	1154	1878	1056	**568.5233**	2931.9326	8929.0809
DEGEN3	630	1173	1803	813	**71.2765**	267.8126	NA
SCTAP2	1206	1327	2533	1755	**166.4448**	706.7056	303.7931
SHIP04L	1773	727	2500	1410	**443.0405**	767.0421	58380.0571
SHIP08S	1180	1426	2606	1103	**681.2449**	2253.1495	NA
SHIP12S	1295	1683	2978	1827	**240.8176**	877.3988	NA
SCTAP3	2298	2131	4429	3651	1257.8763	6013.0127	**1245.7888**

Continued on the next page

Table 4.4 – *Continued from the previous page*

	#Acute	#NonAcute	Row	Col	SAJS with ε=0.40	TP	SNAR
SCSD8	2359	981	3340	1860	**791.3759**	2179.0948	NA
SHIP08L	1773	1272	3045	888	**155.5084**	977.6429	NA
Average					**129.3638**	506.2946	

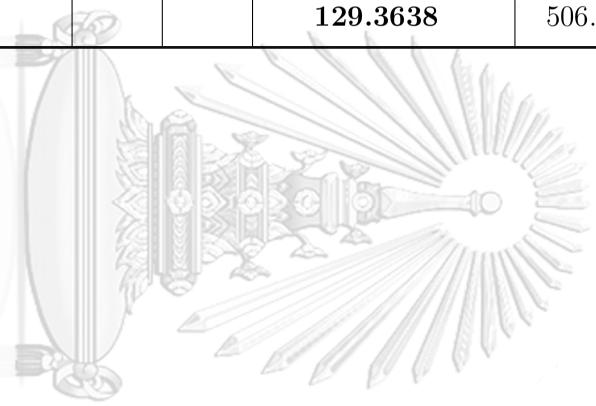


Table 4.4 shows the comparison of SAJS with $\varepsilon = 0.40$, TP and SNAR with 41 standard problems from Netlib. In Table 4.4, #Acute represents the number of acute constraints, #NonAcute represents the number of non-acute constraints, Row represents the number of constraints in the LP model, Col represents the number of variables in the LP model, the boldface numbers identify the smallest wall-clock time. In this dissertation, the maximum time limit of computation is defined as one day (86,400 seconds). If the wall-clock time of computation of the Netlib problem exceeds the time limit, it is represented by NA. The Netlib problems which SNAR exceeds the time limit in Table 4.4 are SCSD6, SCSD8, 25FV47, SCFXM3, SHIP08S, DEGEN3, SHIP12S and SHIP08L.

The results from Table 4.4 are indicated that SAJS with $\varepsilon = 0.40$ gives the minimum average all-clock time for the Netlib problems. Most Netlib problems, SAJS is more effective than TP and SNAR except SC50A, BLEND, SC105, ADLITTLE, BRANDY and SCTAP3. For SC50A, ADLITTLE and BRANDY, TP takes the least wall-clock time. However, the wall-clock times of SAJS are slightly different from TP for these problems. Similarly, BLEND, SC105, and SCTAP3, SNAR takes the least wall-clock time which is only slightly different from SAJS. In addition, Table 4.4 shows that the size of the LP problem influences the wall-clock solution time. For the small-size LP model (the number of variables $\times$ the number of constraints less than 30,000), from AFIRO to ADLITTLE, the performance of SAJS, TP, and SNAR are similar. When the size of the LP model is medium (the number of variables $\times$ the number of constraints is greater than 30,000 but less than 200,000), from SHARE1B to AGG3, the efficiency of SNAR is poor except BRANDY, SCTAP1, SCTAP2, and SCTAP3 which is similar in performance to SAJS. For a large-size LP model (the number of variables $\times$ the number of constraints is greater than 200,000), from SCTAP1 to SHIP08L, SAJS outperforms TP and SNAR except for SCTAP3.

4.4 Conclusion

The self-regulating artificial-free method for solving a linear program, namely SAJS is proposed in this chapter. It consists of three phases: phase 1 is the creation of the LP relaxation having only the acute constraints, phase 2 is the iterative jump performing on the LP relaxation, and phase 3 is the reinsertion the non-acute constraints. The LP relaxation constructed in phase 1 can guarantee the existence of the feasible point. For

the iterative jump method performed in phase 2, it is applied to improve the current feasible point. In phase 3, the non-acute constraints are reinserted to the LP relaxation in order to find the feasible solution of the original LP model.

The improvement of solving the LP model is still an ongoing research. Usually, the simplex method begins with a basic feasible solution and continuously updates the basic feasible solution until the optimal solution is found. In other words, it starts at the initial extreme point and moves that extreme point to adjacent one until the optimal point is found. Therefore, if the LP model consists of enormous constraints, it may visit many extreme points that may affect the time to find the solution. Therefore, the iterative jump method applied in SAJS is introduced to improve the feasible solution by avoiding some extreme points which can reduce the computational time significantly. For the LP model, if the origin point is feasible, the simplex method can start immediately. Otherwise, the artificial variables are added to the LP model then the two-phase simplex method or the big-M simplex method is performed which it increases the size of the LP model. For all above reason, SAJS is presented to improve solving the LP model by using the iterative jump without artificial variables. SAJS does not use artificial variables so it will not increase the problem size. Moreover, SAJS applies the iterative jump method on the LP relaxation having smaller number of extreme points. Likewise, the number of constraints performed by the iterative jump method are smaller than the original LP model that it can reduce the computational time for each iteration.

The two-phase simplex method and SNAR are used to test the effectiveness of SAJS by randomly generated problems and Netlib problems. For all randomly generated problems, SAJS outperforms the two-phase simplex method and SNAR. Since the two-phase simplex method needs to add artificial variables which enlarges the size of the LP model. Thus, it takes a long time to solve the LP model while both SAJS and SNAR do not need artificial variables. However, for most randomly generated problem, SNAR is inferior to those from both SAJS and the two-phase simplex method. Since if the solution of the LP relaxation is unbounded, a single non-acute constraint is reinserted to the LP relaxation one by one which takes a long time. Except for the LP model with few variables where $n = \frac{m}{10}$, SNAR is more effectively than the two-phase simplex method.

To verify the effectiveness of SAJS, Netlib problems are used. The results show

that the average wall-clock time of SAJS is less than both the two-phase method and SNAR significantly. Moreover, the nonparametric Wilcoxon test verified the effectiveness of SAJS. For the Wilcoxon signed-rank test, the p-value of the difference between SAJS and TP is equal to 1.4758×10^{-7} and the p-value of the difference between SAJS and SNAR is equal to 5.9125×10^{-6}. Thus, SAJS statistically significantly outperforms both TP and SNAR.

SAJS is the method for solving the LP model by relaxing all non-acute constraints in order to reduce the number of constraints in performing the iterative jump method. However, applying the iterative jump on the LP relaxation may make the last jump point obtained from the iterative jump method infeasible for the original LP model. As a result, the primal solution of the transformed LP model, which is created by setting the last jump point to the origin point, is infeasible. Moreover, the dual solution of the transformed LP model is infeasible too. Thus, the technique of Pan [2] is applied to change the dual solution is feasible before performing the dual simplex method. From the reasons mentioned above, It makes SAJS complicated and time-consuming to manage. Therefore, AJSP is introduced in order to find the solution of the LP model without artificial variables and without removing constraints by applying the jump technique proposed in Chapter 5.

CHAPTER V

ARTIFICIAL-FREE LINEAR PROGRAMMING USING A JUMP AND THE SIMPLEX METHOD BY STARTING WITH PERTURBED CONSTRAINTS

The artificial-free linear programming using a jump and the simplex method by starting with perturbed constraints called AJSP starts by creating the perturbation LP model that will be suitable to perform the iterative jump method. The perturbation LP model initially keeps all acute constraints which guarantees to have a feasible point. After the initial feasible point is identified, it checked the consistency with all constraints. If that point is feasible for all constraints, then the original LP model is defined as the perturbation LP model. Otherwise, each constraint in which the initial feasible point does not satisfy is disturbed and the original constraints are replaced by the perturbed constraints. This LP model is called the perturbation LP model. After the perturbation LP model is created, the iterative jump method is performed at this initial feasible point. For each iteration of the iterative jump method, the new jump point is checked for consistency with all original constraints. If there is a constraint which the new jump point satisfies that constraint is restored to the original constraint. The process of the iterative jump method is executed until it reaches the stopping criterion. Then, the perturbation LP model will be converted to the transformed LP model. After that, the new technique of solving unrestricted variable problems proposed by Visuthirattanamanee et al. [20] is applied to find the solution of the transformed LP model. After the solution is found, the rest of original constraints are restored. The current solution is the optimal solution of the original LP model when the primal solution is feasible. Otherwise, the dual simplex method is performed to find the solution of the original LP model.

5.1 Artificial-free linear programming using a jump and the simplex method by starting with perturbed constraints

Consider a linear program in the following form:

$$\begin{aligned} &\text{Maximize} \quad \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}. \end{aligned} \tag{5.1}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{m \times n}$. Let $Index = \{1, 2, \ldots, m\}$.

The process of AJSP can be summarized in six steps as follows:

Step 1: Let $\mathbf{I}_{Acute}$ be a set of the acute constraints which can be calculated as follows:

$$\mathbf{I}_{Acute} = \left\{ i \in Index \mid \mathbf{A}_{i:}^\top \mathbf{c} > 0 \right\}.$$

Step 2: The initial feasible point is defined by considering only the acute constraints. Let $\mathbf{I}_{Acute}^{-} = \left\{ i \in \mathbf{I}_{Acute} \mid b_i < 0 \right\}$.

If $\mathbf{b}_{\mathbf{I}_{Acute}} \geq \mathbf{0}$, the initial feasible point ($\mathbf{x}^{(0)}$) is defined as

$\mathbf{x}^{(0)} = \lambda \mathbf{c}$ where $\lambda = \min\limits_{i \in \mathbf{I}_{Acute}} \left\{ \dfrac{b_i}{\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$ and $\gamma^{(0)} = \operatorname{argmin}\limits_{i \in \mathbf{I}_{Acute}} \left\{ \dfrac{b_i}{\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$.

Otherwise, the initial feasible point ($\mathbf{x}^{(0)}$) is defined as

$\mathbf{x}^{(0)} = -\lambda \mathbf{c}$ where $\lambda = \max\limits_{i \in \mathbf{I}_{Acute}^{-}} \left\{ \dfrac{b_i}{-\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$ and $\gamma^{(0)} = \operatorname{argmax}\limits_{i \in \mathbf{I}_{Acute}^{-}} \left\{ \dfrac{b_i}{-\mathbf{A}_{i:}^\top \mathbf{c}} \right\}$.

Step 3: All constraints are checked consistency with $\mathbf{x}^{(0)}$. Let $\mathbf{t} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$. If $\mathbf{t} \geq \mathbf{0}$, then the current LP model is defined as the perturbation LP model immediately and $\mathbf{x}^{(0)}$ is set as the initial feasible point of the iterative jump method. Otherwise, the constraints, which $\mathbf{x}^{(0)}$ does not satisfy, are perturbed before performing the iterative jump method as follows:

For $i \in Index$, $\bar{b}_i = b_i - \beta$ where $\beta = \begin{cases} 0 & \text{if } t_i \geq 0, \\ t_i - 1 & \text{if } t_i < 0. \end{cases}$

Then, the perturbation LP model is constructed as

$$\begin{aligned} &\text{Maximize} \quad \mathbf{c}^\top \mathbf{x} \\ &\text{subject to} \quad \mathbf{A}\mathbf{x} \leq \bar{\mathbf{b}}. \end{aligned} \tag{5.2}$$

Step 4: The iterative jump method is performed by Algorithm 3. The inputs of the algorithm are $\mathbf{A}$, $\mathbf{b}$, $\bar{\mathbf{b}}$, $\mathbf{c}$, $\mathbf{x}^{(0)}$, $\gamma^{(0)}$, *Index* and ε. The stopping parameter (ε) is defined as the least ratio improvement of two consecutive differences of the objective values which the user can accept. This is discussed in section 5.2.1. For each iteration of the iterative jump method, the new jump point needs to check the consistency of all constraints. If there is a constraint which the new jump point satisfies, that constraint is restored to the original constraint.

Algorithm 3 The algorithm of the iterative jump method for AJSP

Input: $\mathbf{A}, \mathbf{b}, \bar{\mathbf{b}}, \mathbf{c}, \mathbf{x}^{(0)}, \gamma^{(0)}, Index, Tol$

Output: $\hat{\mathbf{x}}$

Set $k = 0$, $\Delta z^{(1)} = \Delta z^{(0)} = 1$ where $\Delta z^{(1)}$ and $\Delta z^{(0)}$ are the initial values of the consecutive difference.
Set $temp = \{i \in Index \mid b_i - \mathbf{A}_{i:} \cdot \mathbf{x}^{(0)} < 0\}$
while $\dfrac{\Delta z^{(k+1)}}{\Delta z^{(k)}} > Tol$ **do**
    Compute $\mathbf{v} = \dfrac{-\mathbf{A}_{\gamma^{(k)}:}}{\|\mathbf{A}_{\gamma^{(k)}:}\|} + \dfrac{\mathbf{c}}{\|\mathbf{c}\|}$, where $\mathbf{c} \neq 0$.
    Construct $Index' = Index \setminus \{\gamma^{(k)}\}$.
    Set $\alpha = M$, where M be a large constant.
    for i $\in Index'$ **do**
        if $\mathbf{A}_{i:}^{\top}\mathbf{v} > 0$ and $\bar{b}_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)} > 0$ **then**
            Compute $Dist = \dfrac{\bar{b}_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(k)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}}$.
            if $Dist < \alpha$ **then**
                Define $\alpha = Dist$.
                Define $\gamma^{(k+1)} = i$.
    if $\alpha \neq M$ **then**
        Compute $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{v}$.
        Compute $\Delta z^{(k+1)} = \mathbf{c}\mathbf{x}^{(k+1)} - \mathbf{c}\mathbf{x}^{(k)}$.
        Compute $k = k + 1$.
    else
        break
    if $temp \neq \emptyset$ **then**
        for $i \in Index$ **do**
            if $b_i - \mathbf{A}_{i:} \cdot \mathbf{x}^{(k)} \geq 0$ **then**
                Set $\bar{b}_i = b_i$
                Set $temp = temp \setminus \{i\}$
Set $\hat{\mathbf{x}} = \mathbf{x}^{(k+1)}$

Step 5: After Algorithm 3 terminates, the last jump point ($\hat{\mathbf{x}}$) is relocated to the origin point of the transformed LP model. Let $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$.

Thus, the transformed LP model can be written as.

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \tilde{\mathbf{x}} + \mathbf{c}^\top \hat{\mathbf{x}} \\ \text{subject to} \quad & \mathbf{A}\tilde{\mathbf{x}} \leq \bar{\mathbf{b}} - \mathbf{A}\hat{\mathbf{x}}. \end{aligned} \tag{5.3}$$

After that, the slack variable is added to the transformed LP model.

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^\top \tilde{\mathbf{x}} + \mathbf{c}^\top \hat{\mathbf{x}} \\ \text{subject to} \quad & \mathbf{A}\tilde{\mathbf{x}} + \mathbf{s} = \bar{\mathbf{b}} - \mathbf{A}\hat{\mathbf{x}}, \\ & \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{5.4}$$

Since the current transformed LP model is an unrestricted variable problem, the new technique of solving the unrestricted variable by Visuthirattanamanee et al. [20] is applied to find the solution without adding the artificial variables.

Step 6: After the solution is found, the original constraints are reinserted to the transformed LP model. However, if the primal solution of the current LP model is feasible, then the current solution of the transformed LP model will be the solution of the original model as well. Otherwise, the current simplex tableau is restored to the standard tableau of the simplex method and then the dual simplex method is performed. After that, the solution is restored to the original LP model.

Next, the flowchart of AJSP, comprised of six steps, is shown in Figure 5.1 - 5.3.

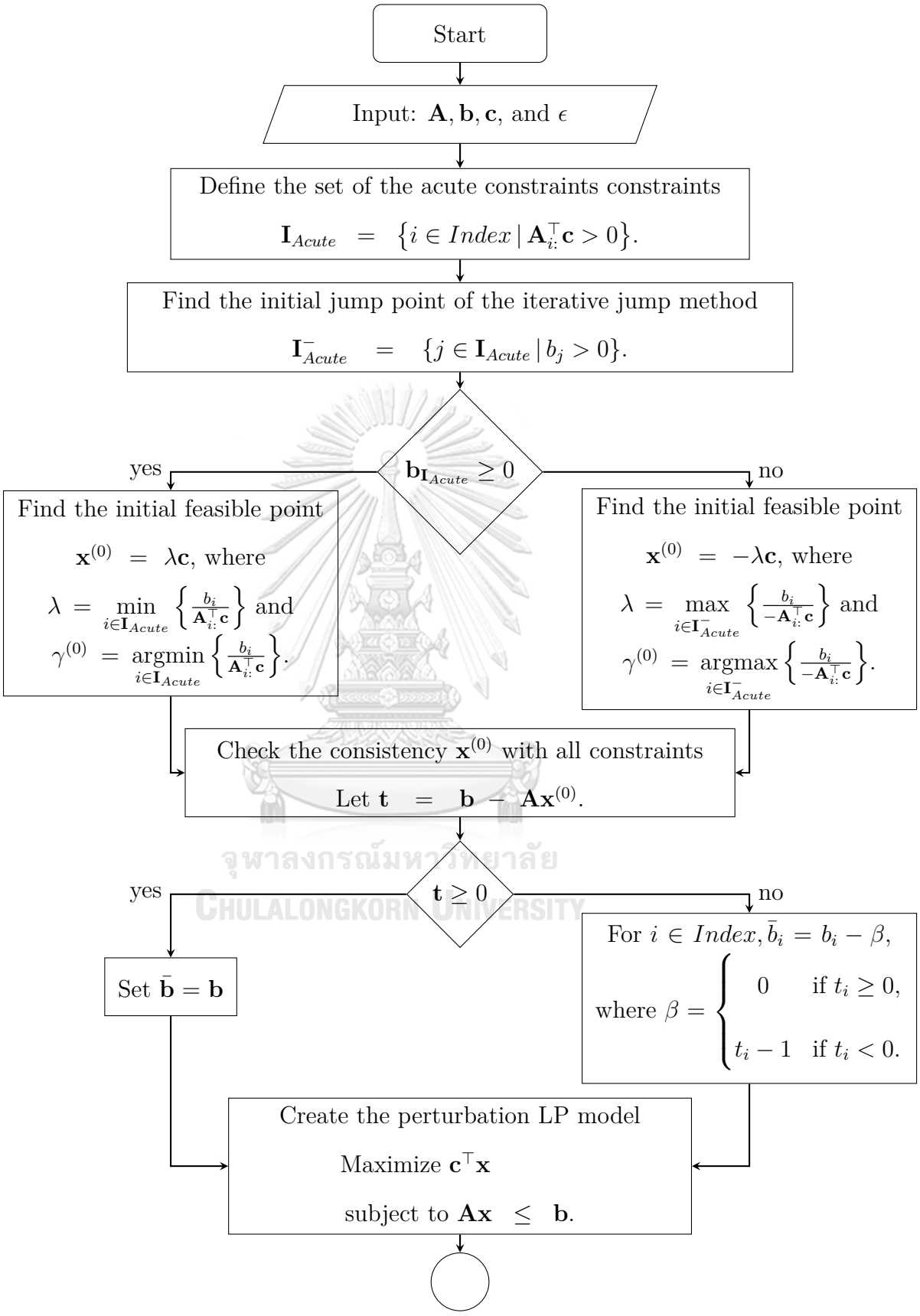


Figure 5.1: The flowchart of AJSP.

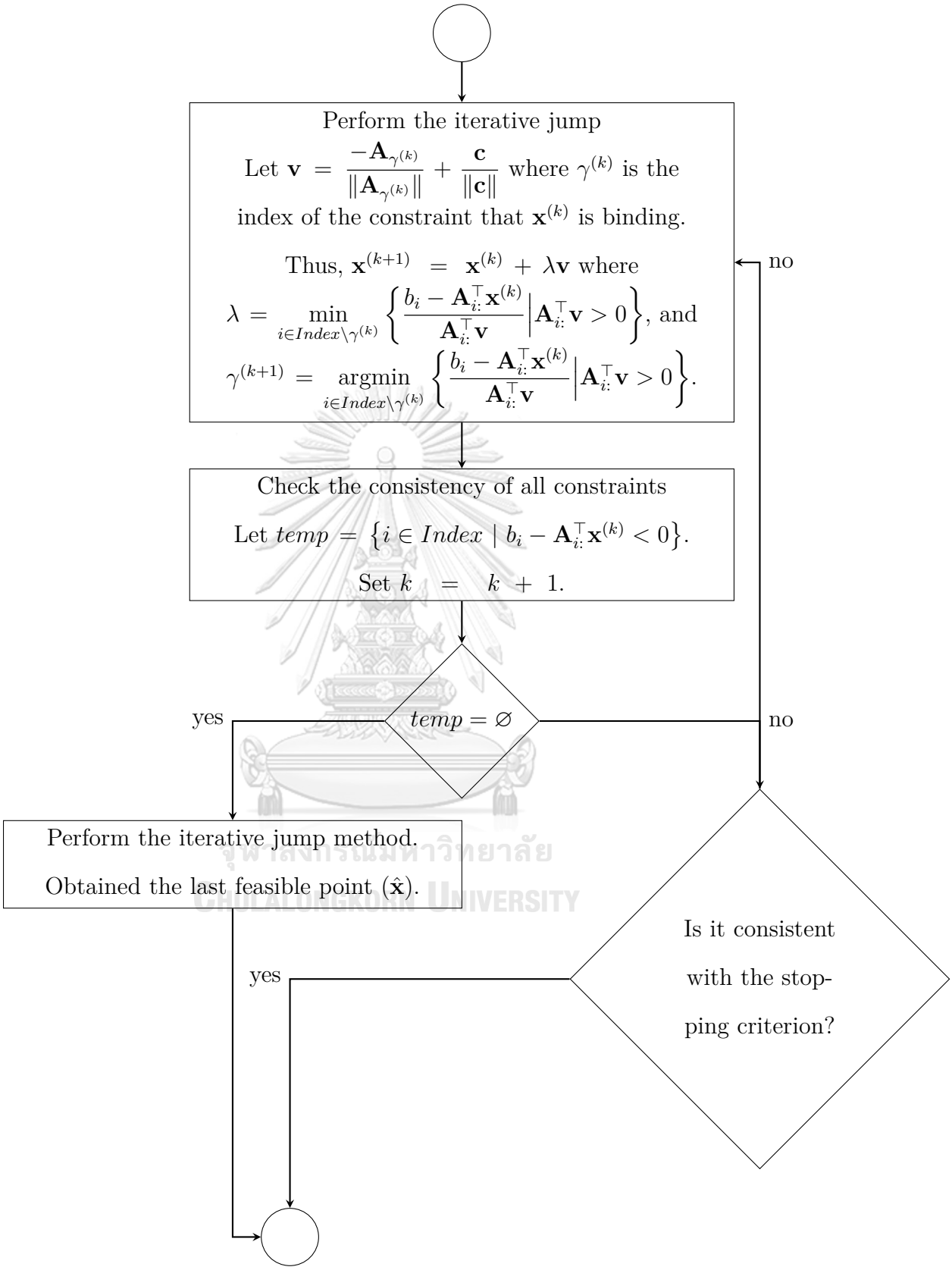


Figure 5.2: The flowchart of AJSP (Con.).

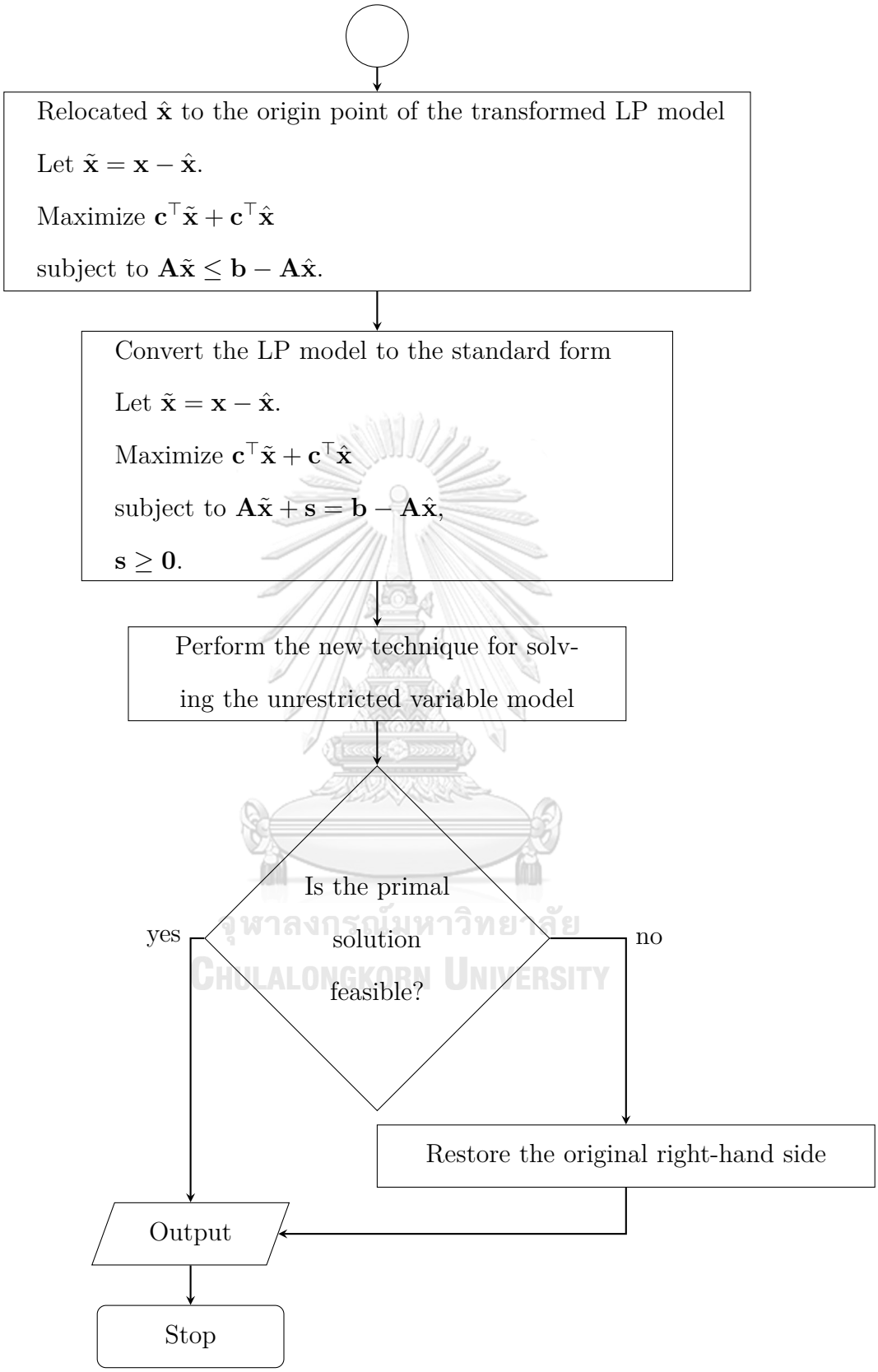


Figure 5.3: The flowchart of AJSP (Con.).

Example 5.1. Consider a linear programming model as follows:

$$
\begin{aligned}
\text{Maximize} \quad & x_1 + x_2 \\
\text{subject to} \quad & x_1 + 5x_2 \leq 36, && (1) \\
& -2x_1 + 4x_2 \leq 3, && (2) \\
& 2x_1 - 3x_2 \leq 18, && (3) \\
& -x_1 + 5x_2 \leq 10, && (4) \\
& -9x_1 + 2x_2 \leq -15, && (5) \\
& -x_2 \leq -1. && (6)
\end{aligned}
\tag{5.5}
$$

Step 1: Separate the constraints into two groups:

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

Thus, $\mathbf{I}_{Acute} = \{1, 2, 4\}$ and $\mathbf{I}_{NonAcute} = \{3, 5, 6\}$.

Step 2: Find the initial feasible point.

Since $\mathbf{b}_{\mathbf{I}_{Acute}} = [36, 3, 10]^{\top} \geq \mathbf{0}$. Thus, λ is computed as follows:

Constraint No. (i)	$\mathbf{A}_{i:}$	$\mathbf{A}_{i:}^{\top}\mathbf{c}$
1	$[1,5]^{\top}$	6
2	$[-2,4]^{\top}$	2
4	$[-1,5]^{\top}$	4

$$\lambda = \min_{i \in \mathbf{I}_{Acute}} \left\{ \frac{b_i}{\mathbf{A}_{i:}^{\top}\mathbf{c}} \right\} = \min_{i \in \{1,2,4\}} \left\{ \frac{36}{6}, \frac{3}{2}, \frac{10}{4} \right\} = 1.5.$$

Therefore, $\mathbf{x}^{(0)} = \lambda \mathbf{c} = [1.5, 1.5]^{\top}$ and $\gamma^{(0)} = 2$.

Step 3: Create the perturbation LP model.

$$\text{Let } \mathbf{t} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)} = \begin{bmatrix} 36 \\ 3 \\ 18 \\ 10 \\ -15 \\ -1 \end{bmatrix} - \begin{bmatrix} 1 & 5 \\ -2 & 4 \\ 2 & -3 \\ -1 & 5 \\ -9 & 2 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix}$$

$$= [27, 0, 19.5, 4, -4.5, 0.5]^{\top}.$$

Let $temp = \{i \in Index \mid b_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(0)} < 0\} = \{5\}$.

$$\text{Compute } \bar{\mathbf{b}}, \text{ for } i \in Index,\ \bar{b}_i = b_i - \beta \text{ where } \beta = \begin{cases} 0 & \text{if } t_i \geq 0, \\ t_i - 1 & \text{if } t_i < 0. \end{cases}$$

$$\text{Thus, } \bar{\mathbf{b}} = \begin{bmatrix} 36 \\ 3 \\ 18 \\ 10 \\ -15 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -5.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 36 \\ 3 \\ 18 \\ 10 \\ -9.5 \\ -1 \end{bmatrix}.$$

Thus, the perturbation LP model is constructed as

$$\begin{aligned}
\text{Maximize} \quad & x_1 + x_2 \\
\text{subject to} \quad & x_1 + 5x_2 \leq 36, && (1) \\
& -2x_1 + 4x_2 \leq 3, && (2) \\
& 2x_1 - 3x_2 \leq 18, && (3) \\
& -x_1 + 5x_2 \leq 10, && (4) \\
& -9x_1 + 2x_2 \leq -9.5, && (5) \\
& -x_2 \leq -1. && (6)
\end{aligned} \tag{5.6}$$

Step 4: Perform the iterative jump method by Algorithm 3.

Iteration 1:

Compute $\mathbf{v} = \dfrac{-\mathbf{A}_{2:}}{\|\mathbf{A}_{2:}\|} + \dfrac{\mathbf{c}}{\|\mathbf{c}\|} = \dfrac{-[-2,4]^\top}{\|[-2,4]^\top\|} + \dfrac{[1,1]^\top}{\|[1,1]^\top\|} = [1.1543, -0.1873]^\top$.

Constraint No. (i)	$\mathbf{A}_{i:}$	$\bar{b}_i$	$\mathbf{A}_{i:}^\top \mathbf{v}$	$\bar{b}_i - \mathbf{A}_{i:}^\top \mathbf{x}^{(0)}$
1	$[1,5]^\top$	36	0.2178	27
3	$[2,-3]^\top$	18	2.8705	19.5
4	$[-1,5]^\top$	10	−2.0908	4
5	$[-9,2]^\top$	−9.5	−10.7633	1
6	$[0,-1]^\top$	−1	0.1873	0.5

Compute $\lambda = \min\limits_{i \in Index \setminus \{\gamma^{(0)}\}} \left\{ \dfrac{\bar{b}_i - \mathbf{A}_{i:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{i:}^\top \mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^\top \mathbf{v} > 0 \right\}$

$= \min\limits_{i \in \{1,3,6\}} \left\{ \dfrac{27}{0.2178}, \dfrac{19.5}{2.8705}, \dfrac{0.5}{0.1873} \right\} = 2.6695$

and $\gamma^{(1)} = \underset{i \in \mathbf{I}_{Acute} \setminus \{\gamma^{(0)}\}}{\operatorname{argmin}} \left\{ \dfrac{\bar{b}_i - \mathbf{A}_{i:}^\top \mathbf{x}^{(0)}}{\mathbf{A}_{i:}^\top \mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^\top \mathbf{v} > 0 \right\} = 6.$

Thus, $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda \mathbf{v} = \begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix} + 2.6695 \begin{bmatrix} 1.1543 \\ -0.1873 \end{bmatrix} = \begin{bmatrix} 4.5814 \\ 1.0000 \end{bmatrix}$.

Check the consistency of $\mathbf{x}^{(1)}$ with the perturbed constraints. Compute $temp = \left\{ i \in temp \mid b_i - \mathbf{A}_{i:}^{\top} \mathbf{x}^{(1)} \right\} = \{24.2326\}$. Thus, $\bar{\mathbf{b}}_5$ is restored as -15 and the set of $temp = temp \setminus \{5\} = \emptyset$.

Iteration 2:

Compute $\mathbf{v} = \frac{-\mathbf{A}_{6:}}{\|\mathbf{A}_{6:}\|} + \frac{\mathbf{c}}{\|\mathbf{c}\|} = \frac{-[0,-1]^{\top}}{\|[0,-1]^{\top}\|} + \frac{[1,1]^{\top}}{\|[1,1]^{\top}\|} = [0.7071, 1.7071]^{\top}$.

Constraint No. (i)	$\mathbf{A}_{i:}$	$\bar{b}_i$	$\mathbf{A}_{i:}^{\top}\mathbf{v}$	$\bar{b}_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(1)}$
1	$[1,5]^{\top}$	36	9.2426	26.4188
2	$[-2,4]^{\top}$	3	5.4142	8.1622
3	$[2,-3]^{\top}$	18	-3.7071	11.8377
4	$[-1,5]^{\top}$	10	7.8284	9.5811
5	$[-9,2]^{\top}$	-15	-2.9497	24.2302

Compute $\lambda = \min_{i \in Index \setminus \{\gamma^{(1)}\}} \left\{ \frac{\bar{b}_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(1)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^{\top}\mathbf{v} > 0 \right\}$

$= \min_{i \in \{1,2,4\}} \left\{ \frac{26.4188}{9.2426}, \frac{8.1622}{5.4142}, \frac{9.5811}{7.8284} \right\} = 1.2238$

and $\gamma^{(2)} = \underset{i \in \mathbf{I}_{Acute} \setminus \{\gamma^{(1)}\}}{\operatorname{argmin}} \left\{ \frac{\bar{b}_i - \mathbf{A}_{i:}^{\top}\mathbf{x}^{(1)}}{\mathbf{A}_{i:}^{\top}\mathbf{v}} \;\middle|\; \mathbf{A}_{i:}^{\top}\mathbf{v} > 0 \right\} = 4.$

Thus, $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \lambda \mathbf{v} = \begin{bmatrix} 4.5814 \\ 1.0000 \end{bmatrix} + 1.2238 \begin{bmatrix} 0.7071 \\ 1.7071 \end{bmatrix} = \begin{bmatrix} 5.4465 \\ 3.0893 \end{bmatrix}$.

The algorithm will be continued until it satisfies the stopping criterion. Consequently, the last jump point $(\hat{\mathbf{x}})$ of Algorithm 3 with 7 iterations is $[14.9534, 3.9689]^{\top}$ which it is binding on 3^{th} constraint.

Step 5: Relocate $\hat{\mathbf{x}}$ to the origin point of the transformed LP model and apply the new technique of solving the unrestricted variable problem for finding the solution.

Let $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$. Therefore, the transformed LP model is created as follows:

$$\begin{aligned}
\text{Maximize} \quad & \tilde{x}_1 + \tilde{x}_2 \\
\text{subject to} \quad & \tilde{x}_1 + 5\tilde{x}_2 \leq 1.2015, && (1) \\
& -2\tilde{x}_1 + 4\tilde{x}_2 \leq 17.0310, && (2) \\
& 2\tilde{x}_1 - 3\tilde{x}_2 \leq 0, && (3) \\
& -\tilde{x}_1 + 5\tilde{x}_2 \leq 5.1085, && (4) \\
& -9\tilde{x}_1 + 2\tilde{x}_2 \leq 111.6434, && (5) \\
& -\tilde{x}_2 \leq 2.9690. && (6)
\end{aligned} \tag{5.7}$$

After that, the slack variables are reinserted to the transformed LP model.

$$\begin{aligned}
\text{Maximize} \quad & \tilde{x}_1 + \tilde{x}_2 \\
\text{subject to} \quad & \tilde{x}_1 + 5\tilde{x}_2 + s_1 = 1.2015, && (1) \\
& -2\tilde{x}_1 + 4\tilde{x}_2 + s_2 = 17.0310, && (2) \\
& 2\tilde{x}_1 - 3\tilde{x}_2 + s_3 = 0, && (3) \\
& -\tilde{x}_1 + 5\tilde{x}_2 + s_4 = 5.1085, && (4) \\
& -9\tilde{x}_1 + 2\tilde{x}_2 + s_5 = 111.6434, && (5) \\
& -\tilde{x}_2 + s_6 = 2.9690, && (6) \\
& s_1, s_2, s_3, s_4, s_5, s_6 \geq 0.
\end{aligned} \tag{5.8}$$

Then, the new technique of solving the unrestricted variable by Visuthirattanamanee et al. [20] is applied. For the transformed LP model, x_1, x_2 are unrestricted variables and $x_3, s_1, s_2, s_3, s_4, s_5, s_6$ are restricted variables. So, the initial simplex tableau is constructed as follows:

	$\tilde{x}_1$	$\tilde{x}_2$	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	-1	-1	0	0	0	0	0	0	18.9223
s_1	1	5	1	0	0	0	0	0	1.2015
s_2	-2	4	0	1	0	0	0	0	17.0310
s_3	**2**	-3	0	0	1	0	0	0	0
s_4	-1	5	0	0	0	1	0	0	5.1085
s_5	-9	2	0	0	0	0	1	0	111.6434
s_6	0	-1	0	0	0	0	0	1	2.9690

The maximum value between the absolute value of the reduced cost of the unrestricted variables and the positive value of the reduced cost of the restricted variable is 1. Thus, the entering variable is $\tilde{x}_1$. Since $\tilde{x}_1$ is the unrestricted variable and the reduced cost is positive. Thus, the index of the leaving variables can be calculated as $\text{argmin}\left\{\frac{1.2015}{1}, \frac{0}{2}\right\} = 3$. Therefore, the leaving variable is s_3.

	$\tilde{x}_1$	$\tilde{x}_2$	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	0	-2.5000	0	0	0.5000	0	0	0	18.9223
s_1	0	**6.5000**	1	0	-0.5000	0	0	0	1.2015
s_2	0	1	0	1	1	0	0	0	17.0310
$\tilde{x}_1$	1	-1.5000	0	0	0.5000	0	0	0	0
s_4	0	3.5000	0	0	0.5000	1	0	0	5.1085
s_5	0	-11.5000	0	0	4.5000	0	1	0	111.6434
s_6	0	-1.0000	0	0	0	0	0	1	2.9690

The entering variable is $\tilde{x}_2$ which it is the unrestricted variable with the positive reduced cost. Thus, the leaving variable is s_1.

	$\tilde{x}_1$	$\tilde{x}_2$	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	0	0	0.3846	0	0.3077	0	0	0	19.3844
$\tilde{x}_2$	0	1	0.1538	0	−0.0769	0	0	0	0.1848
s_2	0	0	−0.1538	1	1.0769	0	0	0	16.8462
$\tilde{x}_1$	1	0	0.2308	0	0.3846	0	0	0	0.2773
s_4	0	0	−0.5385	0	0.7692	1	0	0	4.4615
s_5	0	0	1.7692	0	3.6154	0	1	0	113.7691
s_6	0	0	0.1538	0	−0.0769	0	0	1	3.1538

Since the current simplex tableau is the optimal tableau, so the method of solving the unrestricted variable problem by Visuthirattanamanee et al. terminates with the solution as $\tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2]^\top = [0.2773, 0.1848]^\top$.

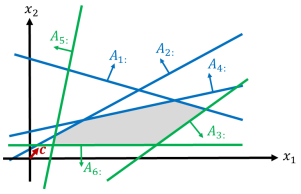
Step 6: Restored the original constraints instead of perturbed constraints.

Since $temp = \{i \in Index \mid b_i - \mathbf{A}_{i:} \cdot \tilde{\mathbf{x}} < 0\} = \emptyset$, all original constraints have been restored. Therefore, the solution of the transformed LP model is restored to the solution of the original LP model as follows:

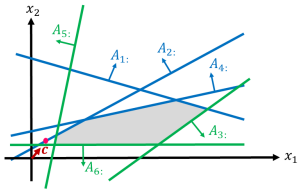
$$\begin{aligned}
[x_1, x_2]^\top &= [\tilde{x}_1, \tilde{x}_2]^\top + [\hat{x}_1, \hat{x}_2]^\top \\
&= [0.2773, 0.1848]^\top + [14.9534, 3.9689]^\top \\
&= [15.2308, 4.1538]^\top.
\end{aligned}$$

The objective value is 19.3846.

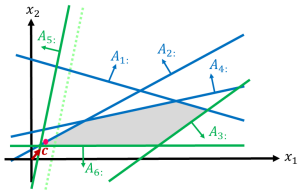
The geometric views of Example 5.1 are shown in Figure 5.4(a)-5.4(g) as follows:



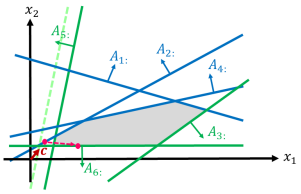
(a) Separate constraints in two groups



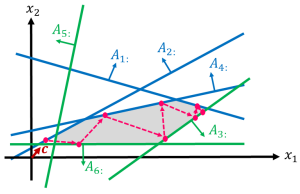
(b) Find an initial feasible point



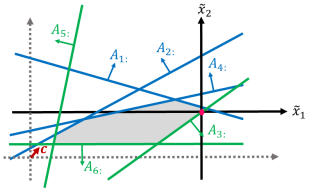
(c) Create a perturbation LP model



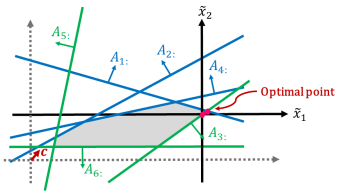
(d) Perform the iterative jump and restore original constraints



(e) Perform the iterative jump



(f) Construct a transformed LP model



(g) Perform the new technique of solving an unrestricted variable problem

Figure 5.4: Geometric views of AJSP performing on Example 5.1

5.2 Results and experiments

In this section, the LP problems used to test the effectiveness of AJSP are explained. The tested LP problems are divided into two collections similar to SAJS: the randomly generated problems and the Netlib problems. Both data sets of the randomly generated problems and data sets of the Netlib problems will be the same set of data used to test the effectiveness of SAJS. After that, the computational results of the wall-clock time of SAJS, AJSP, SNAR, and the two-phase simplex method (TP) are shown and summarized. The experiments were operated using an Intel(R) Core(TM) i7-3770 CPU@ 3.40 GHz processor with 8 GB RAM on Windows 10. All methods were written by NumPy.

Similarly to SAJS, the stopping criterion uses the stopping parameter defined as the least ratio improvement of two consecutive differences of the objective values. The average wall-clock times of AJSP with varying ε are tested and are shown in Table 5.1.

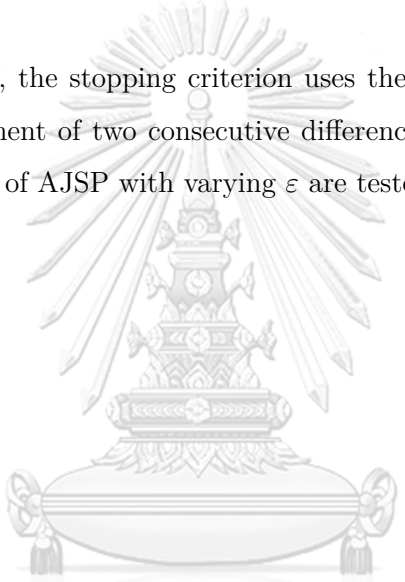


Table 5.1: AJSP performances varying ε.

Row	Col	The average wall-clock time of AJSP (sec.)				
		ε=0.20	ε=0.30	ε=0.40	ε=0.50	ε=0.60
100	10	0.0139 (±0.0065)	**0.0129** (±0.0059)	0.0199 (±0.0116)	0.0148 (±0.0182)	0.0224 (±0.0301)
	20	0.0351 (±0.0118)	0.0335 (±0.0170)	0.0364 (±0.0228)	**0.0270** (±0.0062)	0.0373 (±0.0192)
	30	**0.0433** (±0.0158)	0.0602 (±0.0255)	0.0588 (±0.0330)	0.0652 (±0.0420)	0.0475 (±0.0147)
	40	0.0738 (±0.0233)	**0.0631** (±0.0158)	0.0660 (±0.0170)	0.0703 (±0.0349)	0.0861 (±0.0511)
200	20	**0.0697** (±0.0349)	0.0918 (±0.0395)	0.0768 (±0.0386)	0.0779 (±0.0276)	0.0860 (±0.0404)
	40	**0.1299** (±0.0607)	0.1591 (±0.0273)	0.1358 (±0.0199)	0.1769 (±0.0553)	0.1914 (±0.0891)
	60	0.2553 (±0.0496)	**0.2542** (±0.0708)	0.2893 (±0.0853)	0.3089 (±0.0917)	0.2772 (±0.0777)
	80	0.3793 (±0.0454)	**0.3483** (±0.0774)	0.3792 (±0.0850)	0.3902 (±0.0777)	0.3687 (±0.1002)
300	30	0.2201 (±0.0719)	0.2033 (±0.0785)	0.2052 (±0.0818)	**0.1904** (±0.0472)	0.2117 (±0.1020)
	60	0.4843 (±0.0714)	**0.4048** (±0.0990)	0.4379 (±0.1051)	0.4503 (±0.0943)	0.4437 (±0.0880)
	90	0.6926 (±0.1278)	0.7004 (±0.0900)	0.7293 (±0.1245)	**0.6911** (±0.0982)	0.6921 (±0.0738)
	120	1.1041 (±0.1262)	**1.0772** (±0.1466)	1.1311 (±0.1004)	1.1706 (±0.1783)	1.0508 (±0.0812)

Continued on the next page

Table 5.1 – *Continued from the previous page*

Row	Col	The average wall-clock time of AJSP (sec.)				
		ε=0.20	ε=0.30	ε=0.40	ε=0.50	ε=0.60
400	40	**0.3792** (±0.0727)	0.4126 (±0.1146)	0.3943 (±0.0943)	0.4145 (±0.0974)	0.3949 (±0.0995)
	80	0.9043 (±0.1026)	0.9374 (±0.0987)	0.9295 (±0.1295)	**0.8922** (±0.1051)	0.9155 (±0.0984)
	120	1.4823 (±0.1143)	**1.4043** (±0.1237)	1.4049 (±0.1085)	1.4532 (±0.1749)	1.4503 (±0.1865)
	160	2.0271 (±0.2100)	**1.9981** (±0.1660)	2.1461 (±0.2775)	2.0426 (±0.2027)	1.9785 (±0.2056)
500	50	0.7675 (±0.1285)	0.8287 (±0.1028)	0.7328 (±0.1312)	0.7654 (±0.0723)	**0.7271** (±0.1403)
	100	1.7330 (±0.1824)	1.5983 (±0.1668)	**1.5761** (±0.1205)	1.6453 (±0.2195)	1.6606 (±0.2138)
	150	2.4855 (±0.3017)	**2.4383** (±0.1719)	2.4650 (±0.2290)	2.4891 (±0.1795)	2.4745 (±0.1295)
	200	3.7474 (±0.2270)	3.7158 (±0.2037)	3.6693 (±0.2594)	**3.6190** (±0.2321)	3.6499 (±0.2517)
Average		0.8514	**0.8371**	0.8442	0.8477	0.8383

Table 5.1 shows the average wall-clock time (in seconds) varying $\varepsilon = 0.20, 0.30, 0.40, 0.50, 0.60$ of AJSP. For Table 5.1, Row represents the number of constraints in the LP model, Col represents the number of variables in the LP model, the boldface numbers identify the smallest average wall-clock time and the number in parenthesis represents the standard deviations LP model size. AJSP is tested the stopping criterion with the vary size LP model as $m \in \{100, 200, 300, 400, 500\}$ and $n \in \left\{\frac{m}{10}, \frac{2m}{10}, \frac{3m}{10}, \frac{4m}{10}\right\}$. The results show that the average wall-clock times of AJSP for each problem size, they give the results that are not significantly different in each stopping parameter. However, AJSP with $\varepsilon = 0.30$ takes the least total average wall-clock time, but SAJS with $\varepsilon = 0.40$ is used previously. Therefore, in this section, SAJS and AJSP use both settings $\varepsilon = 0.30$ and 0.40 to compare with TP and SNAR with 280 randomly generated problems presented in Table 5.2.

Table 5.2: The average wall-clock time of SAJS with $\varepsilon = 0.30$, SAJS with $\varepsilon = 0.40$, AJSP with $\varepsilon = 0.30$, AJSP with $\varepsilon = 0.40$, TP, and SNAR on randomly generated linear programming problems.

Row	Col	SAJS		AJSP		TP	SNAR
		ε=0.30	ε=0.40	ε=0.30	ε=0.40		
100	10	0.0112 (±0.0085)	**0.0087** (±0.0078)	0.0129 (±0.0059)	0.0199 (±0.0116)	0.0545 (±0.0391)	0.0328 (±0.0295)
	20	**0.0293** (±0.0132)	0.0393 (±0.0203)	0.0335 (±0.0170)	0.0364 (±0.0228)	0.0754 (±0.0254)	0.0811 (±0.0350)
	30	0.0646 (±0.0162)	0.0621 (±0.0260)	0.0602 (±0.0255)	**0.0588** (±0.0330)	0.0859 (±0.0339)	0.1769 (±0.0778)
	40	0.0751 (±0.0227)	0.0814 (±0.0197)	**0.0631** (±0.0158)	0.0660 (±0.0170)	0.1232 (±0.0413)	0.3361 (±0.1292)
200	20	0.0716 (±0.0244)	**0.0653** (±0.0188)	0.0918 (±0.0395)	0.0768 (±0.0386)	0.2587 (±0.1502)	0.1060 (±0.0382)
	40	0.2257 (±0.0608)	0.2539 (±0.0627)	0.1591 (±0.0273)	**0.1358** (±0.0199)	0.4503 (±0.1196)	0.4667 (±0.1568)
	60	0.3907 (±0.1089)	0.3455 (±0.0720)	**0.2542** (±0.0708)	0.2893 (±0.0853)	0.4563 (±0.0778)	1.4911 (±0.4156)
	80	0.5634 (±0.1182)	0.5604 (±0.1117)	**0.3483** (±0.0774)	0.3792 (±0.0850)	0.5722 (±0.1070)	2.9059 (±0.5067)
300	30	0.2647 (±0.0879)	0.2937 (±0.0982)	**0.2033** (±0.0785)	0.2052 (±0.0818)	1.0987 (±0.2552)	0.4030 (±0.2304)

Continued on the next page

Table 5.2 – *Continued from the previous page*

Row	Col	SAJS		AJSP		TP	SNAR
		ε=0.30	ε=0.40	ε=0.30	ε=0.40		
300	60	0.8509 (±0.0983)	0.7412 (±0.1450)	**0.4048** (±0.0990)	0.4379 (±0.1051)	1.5624 (±0.2246)	1.5629 (±0.4752)
	90	1.0895 (±0.1556)	1.0973 (±0.1313)	**0.7004** (±0.0900)	0.7293 (±0.1245)	1.8698 (±0.4564)	6.4747 (±2.3881)
	120	1.6880 (±0.2283)	1.7038 (±0.2158)	**1.0772** (±0.1466)	1.1311 (±0.1004)	2.5036 (±0.3773)	16.8580 (±5.4162)
400	40	0.5355 (±0.0953)	0.5505 (±0.1447)	0.4126 (±0.1146)	**0.3943** (±0.0943)	1.8357 (±0.7608)	0.8795 (±0.2560)
	80	1.5954 (±0.2490)	1.5347 (±0.2348)	0.9374 (±0.0987)	**0.9295** (±0.1295)	3.9229 (±1.0108)	4.5473 (±1.0408)
	120	2.8127 (±0.3218)	2.8303 (±0.3240)	**1.4043** (±0.1237)	1.4049 (±0.1085)	4.2499 (±0.6664)	31.1020 (±16.5569)
	160	4.2352 (±0.5976)	4.3220 (±0.6476)	**1.9981** (±0.1660)	2.1461 (±0.2775)	6.3061 (±1.0764)	96.1301 (±14.6135)
500	50	1.1474 (±0.1681)	1.1290 (±0.1874)	0.8287 (±0.1028)	**0.7328** (±0.1312)	4.2233 (±1.6131)	1.7442 (±0.2062)
	100	3.3023 (±0.2934)	3.2889 (±0.2554)	1.5983 (±0.1668)	**1.5761** (±0.1205)	6.0040 (±1.4478)	10.1424 (±4.4312)
	150	5.2168 (±0.5792)	5.4445 (±0.3721)	**2.4383** (±0.1719)	2.4650 (±0.2290)	8.4096 (±1.2648)	88.5504 (±43.5178)

Continued on the next page

Table 5.2 – *Continued from the previous page*

Row	Col	SAJS		AJSP		TP	SNAR
		ε=0.30	ε=0.40	ε=0.30	ε=0.40		
500	200	8.1042 (±1.1673)	7.8459 (±1.1333)	3.7158 (±0.2037)	**3.6693** (±0.2594)	11.9502 (±1.3608)	248.8169 (±35.2017)
1000	100	9.4506 (±0.9110)	9.6681 (±1.0846)	**4.0071** (±0.4448)	4.0311 (±0.4510)	41.8906 (±16.9485)	13.5965 (±2.3317)
	200	29.9399 (±3.6104)	30.5525 (±4.3661)	**10.0950** (±0.8582)	10.2454 (±0.8534)	67.8041 (±19.6872)	345.9712 (±76.8162)
	300	58.6559 (±4.2964)	58.5633 (±4.5964)	15.7174 (±0.7225)	**15.6586** (±0.7762)	100.1301 (±18.2994)	1,325.4713 (±355.2482)
	400	89.0246 (±14.9300)	86.3726 (±14.0708)	23.3376 (±1.5929)	**23.1060** (±1.2241)	119.9710 (±12.2241)	4,005.7372 (±510.1724)
2000	200	98.4317 (±11.5698)	98.4510 (±10.9009)	**33.7290** (±2.5017)	33.8181 (±2.7245)	490.0733 (±129.8028)	255.0905 (±60.6867)
	400	358.2615 (±49.5030)	356.8204 (±40.8287)	**71.7862** (±5.6951)	72.3548 (±5.1443)	628.8453 (±171.6103)	4,675.7620 (±1419.6281)
	600	735.4034 (±67.1526)	732.6358 (±83.2131)	117.6444 (±8.5141)	**116.8141** (±7.1897)	1,020.2260 (±161.6943)	24,843.2105 (±12659.5754)
	800	978.5941 (±66.2494)	983.3027 (±72.5009)	**165.4271** (±6.3931)	167.0471 (±10.0859)	1,322.7826 (±140.8523)	54,348.7565 (±9224.8811)
Average		85.3584	85.3059	**16.3745**	16.4271	137.4191	3225.9430

The comparison of SAJS with $\varepsilon = 0.30$ and $\varepsilon = 0.40$, AJSP with $\varepsilon = 0.30$ and $\varepsilon = 0.40$, TP and SNAR performances on the randomly generated problems are shown in Table 5.2. In Table 5.2, Row represents the number of constraints in the LP model, Col represents the number of variables in the LP model, the boldface numbers identify the smallest average wall-clock time and the number in parenthesis represents the standard deviations of each size of the LP model.

In Table 5.2, the average wall-clock time for each size of the LP problems of SAJS with $\varepsilon = 0.30$ and SAJS with $\varepsilon = 0.40$ does not give different results. Similarly, the average wall-clock time of both AJSP with $\varepsilon = 0.30$ and AJSP with $\varepsilon = 0.40$ are similar. However, the wall-clock time of AJSP both with $\varepsilon = 0.30$ and $\varepsilon = 0.40$ outperforms SAJS with $\varepsilon = 0.30$ and SAJS with $\varepsilon = 0.40$ respectively for the LP model which the number of constraints are greater than 200. Moreover, both SAJS with $\varepsilon = 0.30$ and AJSP with $\varepsilon = 0.40$ outperform TP and SNAR for all randomly generated LP models. Furthermore, Figure 5.5 to Figure 5.11 present the comparison of the average wall-clock time of SAJS with $\varepsilon = 0.30$, SAJS with $\varepsilon = 0.40$, AJSP with $\varepsilon = 0.30$, AJSP with $\varepsilon = 0.40$, TP and SNAR for each size of the number of constraints $m \in \{100, 200, 300, 400, 500, 1000, 2000\}$.

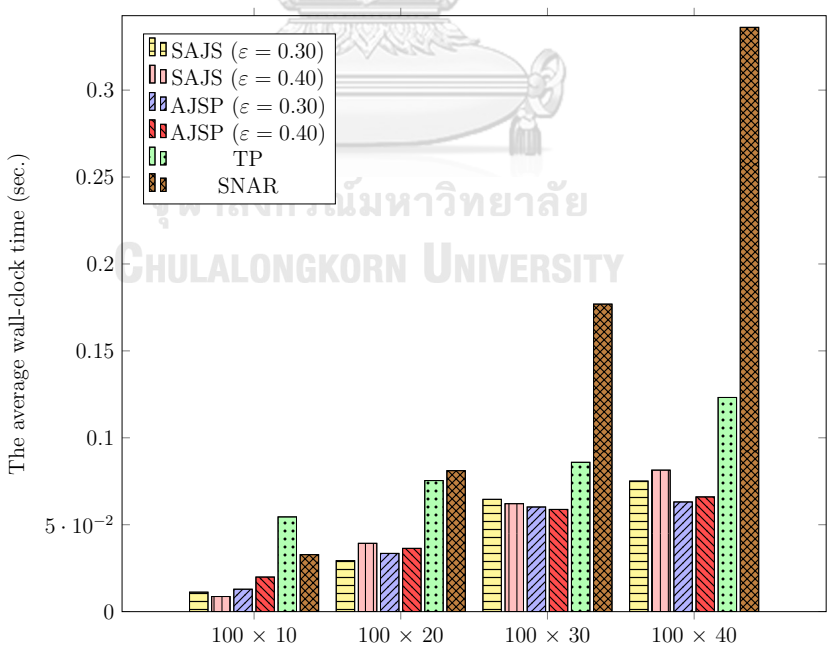


Figure 5.5: The comparison of SAJS, AJSP, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 100 constraints.

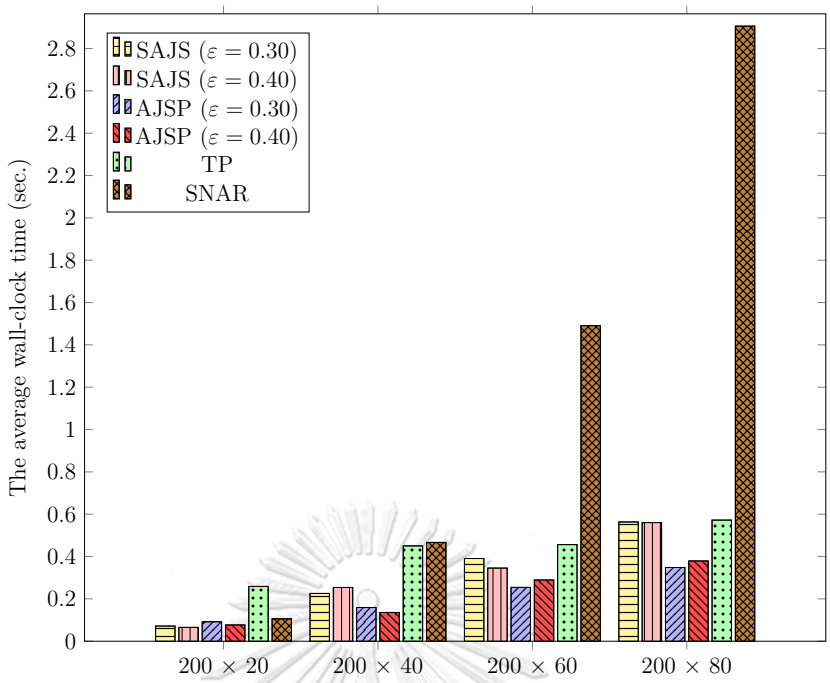


Figure 5.6: The comparison of SAJS, AJSP, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 200 constraints.

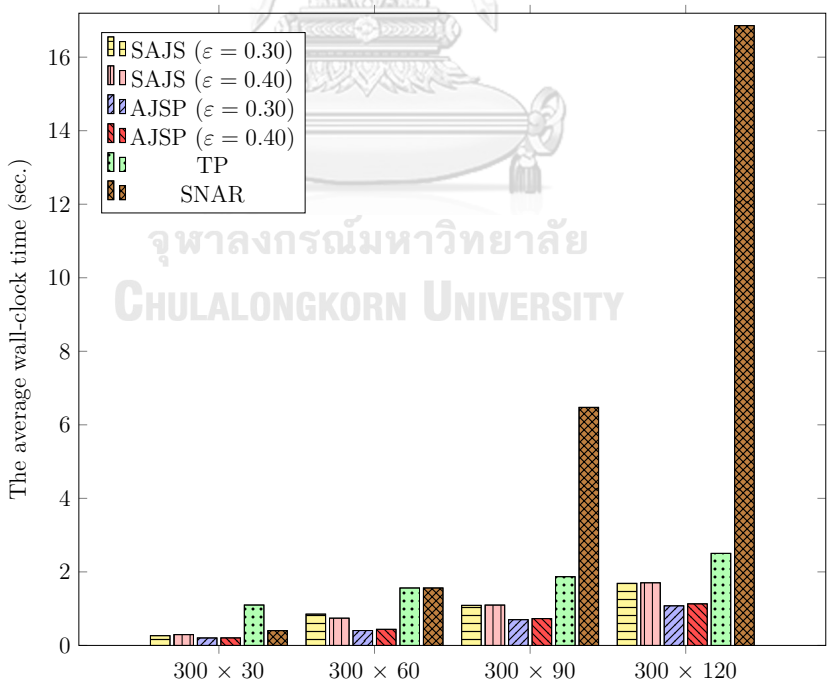


Figure 5.7: The comparison of SAJS, AJSP, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 300 constraints.

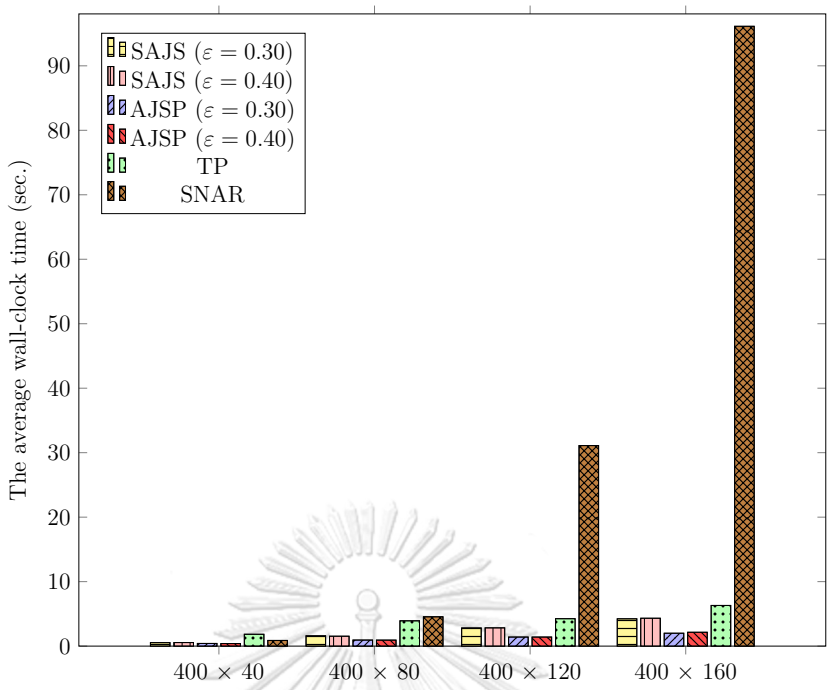


Figure 5.8: The comparison of SAJS, AJSP, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 400 constraints.

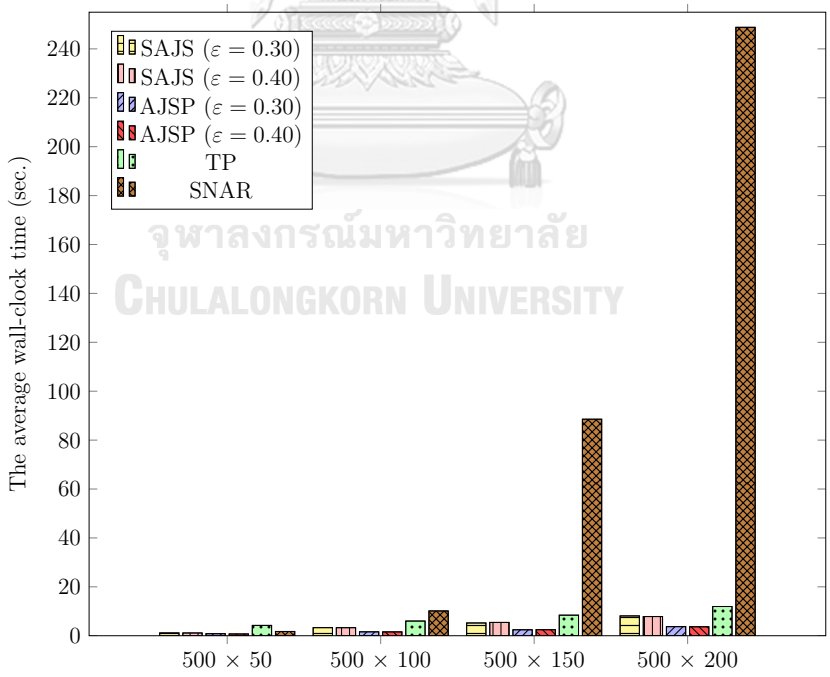


Figure 5.9: The comparison of SAJS, AJSP, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 500 constraints.

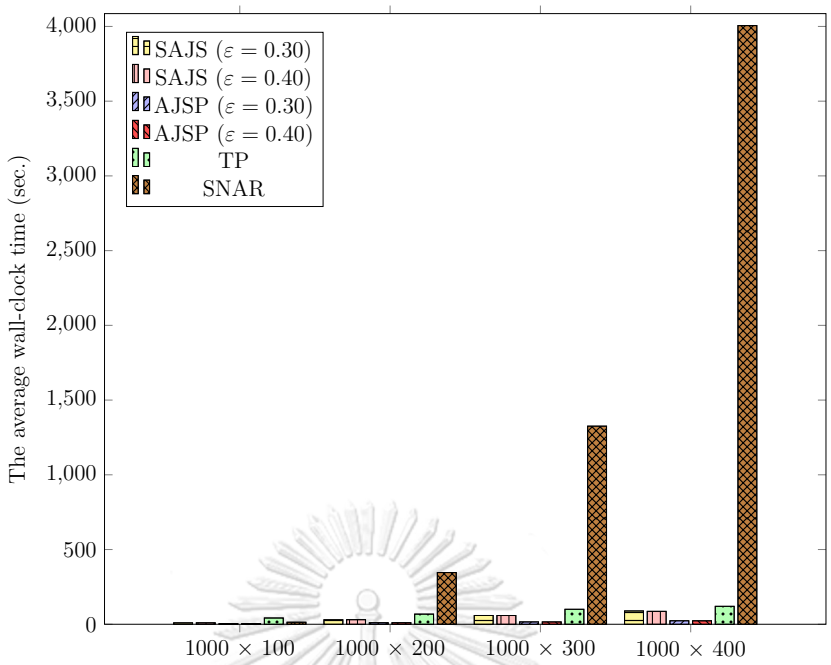


Figure 5.10: The comparison of SAJS, AJSP, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 1000 constraints.

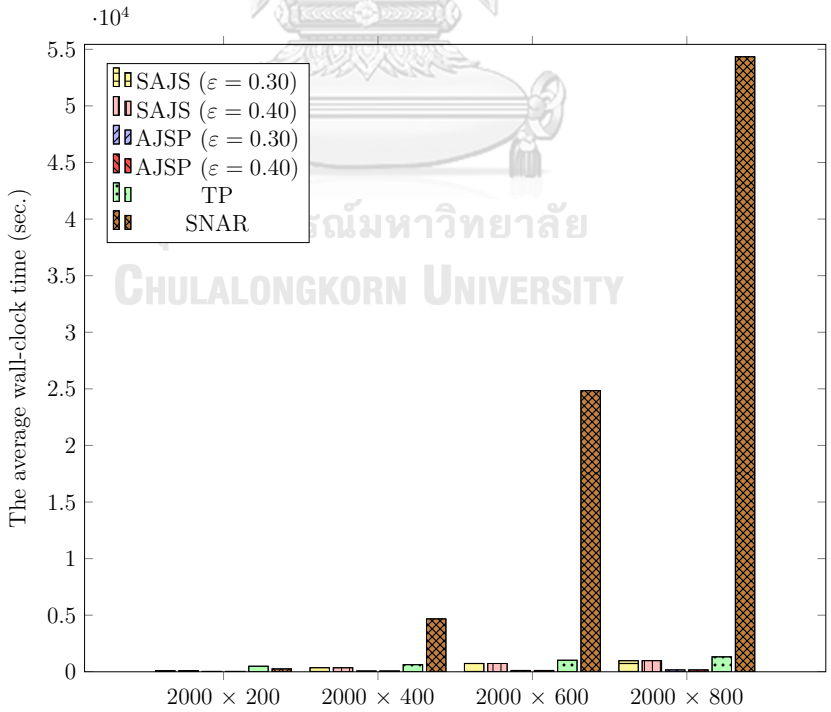


Figure 5.11: The comparison of SAJS, AJSP, TP, and SNAR using the average wall-clock time-randomly generated linear programming problems with 2000 constraints.

From Figure 5.5 - Figure 5.11, they show that the efficiency of SAJS and AJSP are similar. While TP gives more differences to both methods when the size of the LP model is larger. However, the performance of TP still better than SNAR. When the LP model has a larger size, SNAR tends to spend significantly more time.

Table 5.3: The average wall-clock time of SAJS with $\varepsilon = 0.30$, SAJS with $\varepsilon = 0.40$, AJSP with $\varepsilon = 0.30$, AJSP with $\varepsilon = 0.40$, TP, and SNAR on Netlib problems.

	#Acute	#NonAcute	Row	Col	SAJS		AJSP		TP	SNAR
					ε=0.30	ε=0.40	ε=0.30	ε=0.40		
AFIRO	13	38	51	24	0.0083	**0.0070**	0.0094	0.0094	0.0090	0.0160
SC50B	10	68	78	28	0.0259	0.0090	0.0092	**0.0085**	0.0137	0.0269
SC50A	5	73	78	28	0.0041	0.0156	**0.0000**	0.0156	0.0156	0.0312
BLEND	67	71	138	82	0.1127	0.1007	0.1785	0.2394	0.2234	**0.0625**
STOCFOR1	42	72	114	40	**0.0378**	0.0625	0.0892	0.1001	0.1189	0.0937
SC105	63	99	162	66	0.2798	0.3226	**0.1471**	0.2005	0.2323	0.2290
SCAGR7	10	153	163	58	0.0479	**0.0439**	0.0588	0.0668	0.0954	0.3854
SHARE2B	75	90	165	48	0.0868	**0.0828**	0.1456	0.1426	0.1526	0.3788
ADLITTLE	68	117	185	56	**0.0924**	0.3255	0.2765	0.1895	0.1983	0.4397
SHARE1B	145	171	316	142	1.4741	1.7278	**0.9854**	1.2207	2.5567	3.1531
SC205	84	169	253	136	1.0087	0.9219	0.6713	**0.6612**	1.8421	1.5883

Continued on the next page

Table 5.3 – *Continued from the previous page*

	#Acute	#NonAcute	Row	Col	SAJS		AJSP		TP	SNAR
					ε=0.30	ε=0.40	ε=0.30	ε=0.40		
BEACONFD	19	298	317	112	0.4045	0.4648	**0.2783**	0.3271	0.6144	2.6241
BRANDY	135	160	295	122	**0.3927**	0.4721	0.6552	0.7417	0.4358	2.1742
ISRAEL	124	242	366	213	1.8441	**1.6921**	1.9262	2.0311	2.5204	8.0066
SCORPION	7	323	330	110	1.7543	1.6356	1.5376	**1.5372**	2.4394	15.1751
LOTFI	454	306	760	683	13.5085	13.0909	**10.9710**	11.5286	57.3044	49.6100
AGG	240	232	472	249	**5.4080**	5.6064	9.2000	6.7768	8.7660	165.9386
BANDM	361	254	615	127	1.1416	**0.9824**	1.9094	1.7938	1.5254	105.5491
E226	276	220	496	108	0.6050	**0.4608**	0.9948	0.8068	1.1479	137.5253
SCAGR25	219	253	472	167	2.2920	**2.1669**	5.3310	5.1445	6.9685	290.3034
SCFXM1	463	197	660	360	**5.9818**	6.1334	6.3405	6.2642	8.3432	320.4382
AGG2	210	391	601	271	**3.9559**	4.1853	6.4992	6.4314	8.6766	444.9718
AGG3	513	245	758	242	**2.6894**	2.7424	4.4055	4.2887	5.5820	437.4691

Continued on the next page

Table 5.3 – *Continued from the previous page*

	#Acute	#NonAcute	Row	Col	SAJS		AJSP		TP	SNAR
					ε=0.30	ε=0.40	ε=0.30	ε=0.40		
SCTAP1	445	313	758	242	**3.1925**	3.3793	4.8217	5.1526	6.7133	9.5161
DEGEN2	350	409	759	315	14.9319	15.3564	10.2070	**9.9414**	31.5566	333.8480
SCSD1	285	386	671	200	4.0371	**3.9966**	4.4773	4.5660	10.0661	916.5727
SCFXM2	808	542	1350	1203	77.4370	76.3067	60.2844	**59.2801**	308.8540	4199.4493
SCRS8	764	511	1275	785	35.6970	36.3433	**35.0475**	35.3820	109.6057	1979.8150
SCFXM3	753	795	1548	1146	**52.0785**	52.6607	54.3054	53.8516	157.6138	NA
SCSD6	400	802	1202	542	30.0847	30.2915	31.7399	**29.6791**	95.9760	NA
SHIP04S	1111	1097	2208	1806	**159.2345**	159.7317	163.6130	165.0386	562.2637	16490.8526
25FV47	1722	1028	2750	2353	**506.0775**	506.4883	515.4360	509.2073	2391.857715	NA
STOCFOR2	724	1154	1878	1056	568.9057	568.5233	441.0444	**437.7803**	2931.932598	8929.0809
DEGEN3	630	1173	1803	813	**71.1405**	71.2765	89.3766	91.9261	267.8125567	NA
SCTAP2	1206	1327	2533	1755	167.1202	**166.4448**	174.1611	174.5755	706.7055604	303.7931

Continued on the next page

Table 5.3 – *Continued from the previous page*

	#Acute	#NonAcute	Row	Col	SAJS		AJSP		TP	SNAR
					ε=0.30	ε=0.40	ε=0.30	ε=0.40		
SHIP04L	1773	727	2500	1410	430.6910	443.0405	**155.8303**	156.2967	767.0421391	58380.0571
SHIP08S	1180	1426	2606	1103	680.2626	681.2449	**312.3661**	312.7364	2253.149468	NA
SHIP12S	1295	1683	2978	1827	**240.1443**	240.8176	246.3935	244.1919	877.398793	NA
SCTAP3	2298	2131	4429	3651	1254.9843	1257.8763	1131.0695	**1130.7818**	6013.012741	1245.7888
SCSD8	2359	981	3340	1860	824.9749	791.3759	373.8371	**372.5368**	2179.094793	NA
SHIP08L	1773	1272	3045	888	**154.2271**	155.5084	203.6143	199.3007	977.6429229	NA
Average					129.7165	129.3638	99.0304	**98.6037**	506.2946	

Table 5.3 shows the comparison of SAJS with $\varepsilon = 0.30$, SAJS with $\varepsilon = 0.40$, AJSP with $\varepsilon = 0.30$, AJSP with $\varepsilon = 0.40$, TP and SNAR on 41 standard problems from Netlib. #Acute represents the number of acute constraints, #NonAcute represents the number of non-acute constraints, Row represents the number of constraints in the LP model, Col represents the number of variables in the LP model, the boldface numbers identify the smallest wall-clock time.

The minimum average wall-clock time of all Netlib problems matches with AJSP with $\varepsilon = 0.40$. Moreover, for each Netlib problem, SAJS with $\varepsilon = 0.30$ and SAJS with $\varepsilon = 0.40$ take quite similar computational time, likewise AJSP with $\varepsilon = 0.30$ and AJSP with $\varepsilon = 0.40$ which take a slightly different time. Furthermore, both the average wall-clock time of SAJS with $\varepsilon = 0.40$ and the average wall-clock time of AJSP with $\varepsilon = 0.40$ outperform SAJS with $\varepsilon = 0.30$ and AJSP with $\varepsilon = 0.30$, respectively. Accordingly, SAJS with $\varepsilon = 0.40$ and AJSP with $\varepsilon = 0.40$ are used to compare the computational time with TP and SNAR. shown in Table 5.4.

Table 5.4: The difference of the average wall-clock time for SAJS with $\varepsilon = 0.40$, AJSP with $\varepsilon = 0.40$, TP and SNAR on Netlib problems.

	SAJS - AJSP	SAJS-TP	AJSP-TP	SAJS-SNAR	AJSP-SNAR
AFIRO	−0.0024	−0.0020	0.0004	−0.0090	−0.0066
SC50B	0.0005	−0.0047	−0.0052	−0.0179	−0.0184
SC50A	0.0000	0.0000	0.0000	−0.0156	−0.0156
BLEND	−0.1386	−0.1227	0.0160	0.0382	0.1769
STOCFOR1	−0.0376	−0.0565	−0.0188	−0.0312	0.0064
SC105	0.1221	0.0903	−0.0318	0.0936	−0.0285
SCAGR7	−0.0229	−0.0515	−0.0286	−0.3415	−0.3186
SHARE2B	−0.0598	−0.0698	−0.0100	−0.2960	−0.2362
ADLITTLE	0.1360	0.1272	−0.0088	−0.1142	−0.2502
SHARE1B	0.5071	−0.8289	−1.3359	−1.4253	−1.9324

Continued on the next page

Table 5.4 – *Continued from the previous page*

	SAJS - AJSP	SAJS-TP	AJSP-TP	SAJS-SNAR	AJSP-SNAR
SC205	0.2606	−0.9202	−1.1808	−0.6664	−0.9271
BEACONFD	0.1376	−0.1496	−0.2872	−2.1593	−2.2970
BRANDY	−0.2696	0.0363	0.3059	−1.7021	−1.4325
ISRAEL	−0.3390	−0.8282	−0.4893	−6.3145	−5.9755
SCORPION	0.0983	−0.8038	−0.9022	−13.5395	−13.6379
LOTFI	1.5623	−44.2135	−45.7758	−36.5191	−38.0814
AGG	−1.1704	−3.1595	−1.9892	−160.3322	−159.1618
BANDM	−0.8114	−0.5431	0.2684	−104.5667	−103.7553
E226	−0.3460	−0.6871	−0.3411	−137.0645	−136.7185
SCAGR25	−2.9776	−4.8016	−1.8240	−288.1365	−285.1589
SCFXM1	−0.1308	−2.2099	−2.0791	−314.3048	−314.1740
AGG2	−2.2460	−4.4913	−2.2453	−440.7865	−438.5404
AGG3	−1.5462	−2.8396	−1.2933	−434.7267	−433.1804
SCTAP1	−1.7733	−3.3340	−1.5607	−6.1368	−4.3635
DEGEN2	5.4150	−16.2002	−21.6152	−318.4916	−323.9066
SCSD1	−0.5694	−6.0696	−5.5002	−912.5761	−912.0067
SCFXM2	17.0266	−232.5474	−249.5740	−4123.1426	−4140.1692
SCRS8	0.9612	−73.2624	−74.2237	−1943.4717	−1944.4330
SCFXM3	−1.1908	−104.9531	−103.7623	None	None
SCSD6	0.6124	−65.6845	−66.2969	None	None
SHIP04S	−5.3070	−402.5320	−397.2250	−16331.1209	−16325.8140
25FV47	−2.7190	−1885.3694	−1882.6504	None	None

Continued on the next page

Table 5.4 – *Continued from the previous page*

	SAJS - AJSP	SAJS-TP	AJSP-TP	SAJS-SNAR	AJSP-SNAR
STOCFOR2	130.7429	−2363.4093	−2494.1523	−8360.5576	−8491.3006
DEGEN3	−20.6496	−196.5361	−175.8864	None	None
SCTAP2	−8.1307	−540.2607	−532.1300	−137.3483	−129.2176
SHIP04L	286.7438	−324.0017	−610.7455	−57937.0166	−58223.7604
SHIP08S	368.5085	−1571.9045	−1940.4130	None	None
SHIP12S	−3.3743	−636.5812	−633.2069	None	None
SCTAP3	127.0945	−4755.1364	−4882.2310	12.0875	−115.0070
SCSD8	418.8391	−1387.7189	−1806.5580	None	None
SHIP08L	−43.7923	−822.1345	−778.3422	None	None

Table 5.4 demonstrates subtraction of the wall-clock time of SAJS with $\varepsilon = 0.40$ against AJSP with $\varepsilon = 0.40$, TP and SNAR. SAJS-AJSP represents the wall-clock time of SAJS with $\varepsilon = 0.40$ minus the wall-clock time of AJSP with $\varepsilon = 0.40$ for each Netlib problem. SAJS-TP represents the wall-clock time of SAJS with $\varepsilon = 0.40$ minus the wall-clock time of TP for each Netlib problem. AJSP-TP represents the wall-clock time of AJSP with $\varepsilon = 0.40$ minus the wall-clock time of TP for each Netlib problem. SAJS-SNAR represents the wall-clock time of SAJS with $\varepsilon = 0.40$ minus the wall-clock time of SNAR for each Netlib problem. AJSP-SNAR represents the wall-clock time of AJSP with $\varepsilon = 0.40$ minus the wall-clock time of SNAR for each Netlib problem. None represents the Netlib problems that the difference is not applicable.

From Table 5.4, it shows that the wall-clock time of SAJS with $\varepsilon = 0.40$ and AJSP with $\varepsilon = 0.40$ are slightly different except for STOFOR2, SHIP04L, SHIP08S, SCTAP3 and SCSD8 where SAJS with $\varepsilon = 0.40$ takes time more than AJSP with $\varepsilon = 0.40$ exceeding 100 seconds. Moreover, the performance of SAJS with $\varepsilon = 0.40$ and AJSP with $\varepsilon = 0.40$ are superior to that of TP since the wall-clock time performed by both methods are less than TP for most Netlib problems except AFIRO, BLEND, SC105, ADLITTLE, BRANDY and BANDM. Nevertheless, the difference of the wall-clock time

of TP and SAJS with $\varepsilon = 0.40$ for these problems are less than 0.13 seconds. Similarly, the difference of the wall-clock time of TP and AJSP with $\varepsilon = 0.40$ for these problems are less than 0.31 seconds. Likewise, the wall-clock time of SNAR is greater than both the wall-clock time of SAJS with $\varepsilon = 0.40$ and the wall-clock time of AJSP with $\varepsilon = 0.40$ except for BLEND, STOCFOR1, SC105, and SCTAP3. For BLEND, STOCFOR1 and SC105, the difference of the wall-clock time between SNAR and SAJS with $\varepsilon = 0.40$, and the difference of the wall-clock time between SNAR and AJSP with $\varepsilon = 0.40$ are less than 0.18. While the difference of the wall-clock time of SNAR and SAJS with $\varepsilon = 0.40$ for SCTAP3 is less than 12.10. Although SAJS with $\varepsilon = 0.40$ takes more time than SNAR to 12.10 seconds for computation on SCTAP3, the percentage of the difference of the wall-clock time and the wall-clock time of SAJS with $\varepsilon = 0.40$ is only 0.9703%.

5.3 Conclusion

The artificial-free linear programming using a jump and the simplex method by starting with perturbed constraints, namely AJSP is presented in this chapter. It is a method for solving the LP model by applying the iterative jump method without artificial variables. AJSP starts by finding the initial feasible point from the iterative jump method based on all acute constraints which can guarantee the existence of the feasible point. Next, all constraints are checked for consistency with the initial feasible point. If there is a constraint that makes an initial feasible point infeasible, then that constraint is disturbed and replaced. The new LP model with perturbed constraints is called the perturbation LP model. After that, the iterative jump method is performed on the perturbation LP model by starting with the initial feasible point. For each iteration of the iterative jump method, the original constraints must be checked for consistency with each new jump point. If there is a constraint which the new jump point satisfies, then that constraint is restored. The iterative jump method will continue until it reaches the stopping criterion. After that, the last jump point obtained from the iterative jump method is defined as the origin point of the new LP model, called the transformed LP model. Since the transformed LP model is the unrestricted variable LP model. Therefore, the new technique for solving the unrestricted variables LP model proposed by Visuthirattanamanee et al. in chapter 3 is applied to find the solution of the transformed LP model. After the solution of the transformed LP model is found, the original constraints are restored to the LP model. If the current solution satisfies all constraints, then the current solution is the solution

of the original LP model. Otherwise, the current tableau is expanded to the standard simplex tableau and the dual simplex method is performed to find the solution.

AJSP uses the same iterative jump method as SAJS in order to avoid visiting unnecessary extreme points. SAJS starts by creating the LP relaxation while AJSP starts by creating the perturbation LP model. The technique of disturbing the constraints is applied to expand the feasible region in order to easily find the initial point for the iterative jump method. Furthermore, AJSP applies the new technique of solving the unrestricted variables LP model to find the solution of the transformed LP model. As a result, it does not increase the size of the LP model. Moreover, artificial variables are not used in AJSP. So it can reduce the computational time from the standard simplex method.

The two-phase simplex method and SNAR are used to test the effectiveness of AJSP from randomly generated problems and Netlib problems similar to SAJS. For all randomly generated problems, AJSP outperforms SAJS, the two-phase simplex method, and SNAR. Since the artificial variables are added to the LP model for the two-phase simplex method while other methods do not use them. For SAJS, the iterative jump method is performed on the LP relaxation. Although the size of the relaxation LP is smaller than AJSP, the last jump point obtained from the iterative jump method may be far from the optimal point. As a result, it takes a long time to find the solution of the original LP model. While SNAR needs to reinsert a single non-acute constraint one by one to the LP relaxation when the solution of the LP relaxation is unbounded.

To verify the effectiveness of AJSP, Netlib problems are used. The results show that the average wall-clock time of AJSP is less than SAJS, the two-phase simplex method, and SNAR, significantly. Moreover, the nonparametric Wilcoxon test verified the effectiveness of AJSP. For the Wilcoxon signed-rank, the p-value of the difference between AJSP with $\varepsilon = 0.30$ and SAJS with $\varepsilon = 0.30$ is equal to 0.4484, the p-value of the difference between AJSP with $\varepsilon = 0.40$ and SAJS with $\varepsilon = 0.40$ is equal to 0.6573, the p-value of the difference between AJSP with $\varepsilon = 0.30$ and TP is equal to 6.7115×10^{-7}, and the p-value of the difference between AJSP with $\varepsilon = 0.30$, and SNAR is equal to 9.3561×10^{-7}. Therefore, AJSP statistically significantly outperforms other methods.

Although AJSP is effective for solving the LP model, the steps to implement the

iterative jump method for each iteration are more complicated than SAJS. AJSP has to check the consistency of the new jump point obtained in each iteration with the constraints of the perturbation LP model. Moreover, if the solution acquired from the new technique of solving the unrestricted variables LP model does not satisfy with all original constraints, then the current tableau needs to restore the standard simplex tableau before performing the dual simplex method.

CHAPTER VI

CONCLUSION

In this dissertation, both SAJS and AJSP are proposed which are applied to solve a linear programming model without artificial variables using the iterative jump method. The iterative jump method is applied to locate the feasible point for the better objective value while it can avoid visiting unnecessary extreme points of the LP model. The directions of the iterative jump method can improve the objective value of the new jump point and the step size of the iterative jump method will still maintain the feasibility of a new jump point which they are proven in this dissertation. However, finding an initial feasible point is quite complicated. Therefore, SAJS and AJSP are presented.

First, SAJS is introduced. It starts by creating the LP relaxation with acute constraints before it finds the better feasible solution by the iterative jump method. The last jump point obtained from the iterative jump method is set as the origin point of the new LP model called the transformed LP model. Then, all non-acute constraints are reinserted to the transformed LP model to find the solution of the original LP model.

Consider the LP model as follows:

$$\begin{aligned} \text{Maximize} \quad & x_2 \\ \text{subject to} \quad & -x_1 + x_2 \leq 1, \\ & -x_1 + 2x_2 \leq 4, \\ & -x_1 + 3x_2 \leq 8, \\ & x_1 - x_2 \leq 4, \\ & 2x_1 - x_2 \leq 10, \\ & 2x_1 \leq 11. \end{aligned} \tag{6.1}$$

The initial graphical view for LP (6.1) is created in Figure 6.1 which the blue constraints represent the acute constraints and the green constraints represent the non-

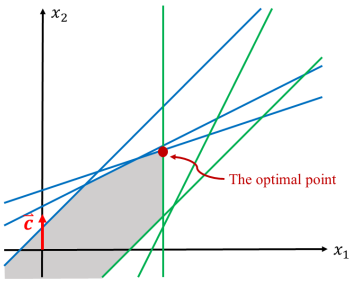


Figure 6.1: Separation of the groups of constraints.

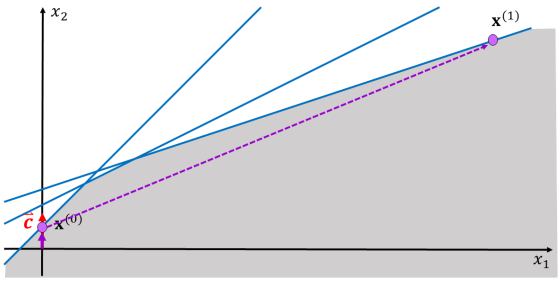


Figure 6.2: Creation of the LP relaxation and performing the iterative jump method

acute constraints.

SAJS begins by establishing the LP relaxation and performing the iterative jump method on the LP relaxation terminating at the last jump point ($\mathbf{x}^{(1)}$) presented in Figure 6.2.

Although the LP relaxation of SAJS can guarantee the available feasible point and can reduce the number of constraints in the computation for each iteration, it may not be the good initial jump point. As a result, $\mathbf{x}^{(1)}$ obtained from the iterative jump method may be infeasible for the original LP model. Therefore, the perturbation technique by Pan [2] is used to guarantee the feasibility of the dual solution before applying the dual simplex method which affects the computational time. For this example, after the non-acute constraints are reinserted to the LP relaxation. The dual simplex method uses 2 iterations to find the optimal solution shown in Figure 6.3.

The improved SAJS called AJSP is introduced. It starts with the same LP relaxation of SAJS. Note the initial feasible point obtained from the first step is not necessarily

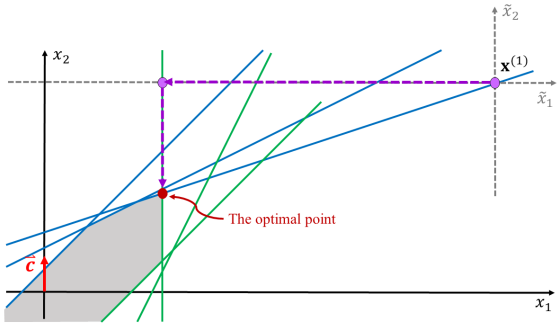


Figure 6.3: Reinsertation of the non-acute constraints and performing the dual simplex method.

a feasible point for the original LP model. It validates all constraints of the LP model with the initial feasible point to construct the perturbation LP model. Next, the iterative jump method is performed on the perturbation LP model while it checks the consistency of constraints with respect to the jump point. After the iterative jump method terminates, the last jump point obtained from the iterative jump method is set as the origin point of the new LP model called the transformed LP model. Since the transformed LP model is an unrestricted variable model. Thus, the new technique for solving the unrestricted variables model is applied. After that, the original constraints are substituted back to the constraints of the transformed LP model in order to find the solution of the original LP model.

The two-phase simplex method (TP) and SNAR are used to test the effectiveness of SAJS and AJSP with the randomly generated LP problems and Netlib problems. For the randomly generated LP problems, the experimental results are shown that both SAJS and AJSP upwardly more effective than TP and SNAR, especially for large LP problems. For solving the LP model by TP, the artificial variables are added when the origin point is infeasible. So, the size of the LP model is expanded which affects the computational time.

In addition, the results of the randomly generated LP problems show that AJSP outperforms SAJS. Although SAJS performs the iterative jump method on the LP relaxation which the number of constraints is less than the number of constraints of the original LP model, it may find the last jump point far from the optimal point. As a result, it takes a long time to move back to the optimal solution of the original LP model.

Moreover, AJSP applies the new technique for solving the unrestricted variables model to find the solution of the transformed LP model avoid the expansion of the LP model.

In order to verify the effectiveness of SAJS and AJSP, Netlib problems are tested with the TP and SNAR. The wall-clock time of both methods is shown that they are rarely different except some of Netlib problems. However, the efficiency of both SAJS and AJSP is significantly better than TP and SNAR. Moreover, the nonparametric Wilcoxon test verified the effectiveness of both SAJS and AJSP. It is shown that SAJP and AJSP statistically significantly outperform TP and SNAR.

Both SAJS and AJSP are proposed to improve solving the linear programming model. The iterative jump is used as an important part of both methods, since it can avoid visiting unnecessary extreme points. SAJS attempts to reduce the number of constraints in the calculation of the iterative jump method. However, the last jump point obtained from the iterative jump method may be an infeasible point for the original LP model which is quite complicated to move the last jump point back to the feasible point of the original LP model. While AJSP tires to disturb the constraints in order to make the suitable point which is defined by considering only the acute constraints as the feasible point. Although the last jump point obtained from the iterative jump method does not locate far from the optimal point of the original LP model, the number of constraints used to implement the iterative jump method is equal to the number of the original constraints. Moreover, the advantage of AJSP has applied the new technique of solving the unrestricted variables LP model which does not necessary increase variables. However, the LP model with a small number of constraints may reduce the efficiency of both methods, since it may cause the LP model having small number of extreme points. As a result, the standard simplex method uses a small number of iterations for solving that LP model while the iterative jump method may cause zigzag (see in Figure 5.4(e)).

In future work, the direction of the iterative jump method can be improved. Moreover, other pivot rules for moving the last jump point to the solution point of the LP model will be applied to improve the performance of SAJS and AJSP.

REFERENCES

[1] G. Dantzig, *Linear programming and extensions.* Princeton, NJ: Princeton Univ. Press, 1963.

[2] P.-Q. Pan, "Primal perturbation simplex algorithms for linear programming," *Journal of Computational Mathematics*, vol. 18, pp. 587–596, 2000.

[3] H. W. Corley, J. Rosenberger, W. chang Yeh, and T. K. Sung, "The cosine simplex algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 27, no. 9, pp. 3397–3401, 2006.

[4] A. Boonperm and K. Sinapiromsaran, "Artificial-free simplex algorithm based on the non-acute constraint relaxation," *Applied Mathematics and Computation*, vol. 234, pp. 385–401, 2014.

[5] H. Nabli and S. Chahdoura, "Algebraic simplex initialization combined with the nonfeasible basis method," *European Journal of Operational Research*, vol. 245, 2015.

[6] H. V. Junior and M. P. E. Lins, "An improved initial basis for the simplex algorithm," *Computers & Operations Research*, vol. 32, no. 8, pp. 1983 – 1993, 2005.

[7] J.-F. Hu, "A note on "an improved initial basis for the simplex algorithm"," *Computers & Operations Research*, vol. 34, no. 11, pp. 3397 – 3401, 2007.

[8] H. Nabli, "An overview on the simplex algorithm," *Applied Mathematics and Computation*, vol. 210, pp. 479–489, 2009.

[9] N. Stojkovic, P. Stanimirovic, and M. Petkovic, "Modification and implementation of two-phase simplex method," *International Journal of Computer Mathematics*, vol. IJCM, pp. 1231–1242, 2009.

[10] N. Stojkovic, P. Stanimirovic, M. Petkovic, and D. Milojković, "On the simplex algorithm initializing," *Abstract and Applied Analysis*, vol. 2012, 2012.

[11] N. Yawila, B. Intiyot, and K. Sinapiromsaran, "Simplex method with objective jump," *Proceedings of International Conference on Applied Statistics (ICAS)*, 2016.

[12] N. Kafakthong and K. Sinapiromsaran, "Preceding-jump simplex method," *Proceedings of International conference on Optimaization and Learning (OLA'2019)*, 2019.

[13] H. Luh and R. Tsaih, "An efficient search direction for linear programming problems," *Computers & Operations Research*, vol. 29, no. 2, pp. 195–203, 2002.

[14] C. Al-Najjar and B. Malakooti, "Hybrid-lp: Finding advanced starting points for simplex, and pivoting lp methods," *Computers & OR*, vol. 38, pp. 427–434, 2011.

[15] W.-C. Yeh and H. Corley, "A simple direct cosine simplex algorithm," *Applied Mathematics and Computation*, vol. 214, pp. 178–186, 2009.

[16] N. Karmarkar, "A new polynomial-time algorithm for linear programming-ii," *Combinatorica*, vol. 4, pp. 373–395, 1984.

[17] K. G. Murty, "The gravitational method for linear programming," *Opsearch*, vol. 23, pp. 206–214, 1986.

[18] P.-Q. Pan, "An affine-scaling pivot algorithm for linear programming," *Optimization*, vol. 62, no. 4, pp. 431–445, 2013.

[19] E. R. Barnes, "A variation on karmarkar's algorithm for solving linear programming problems," *Math. Program.*, vol. 36, pp. 174–182, June 1986.

[20] R. Visuthirattanamanee, K. Sinapiromsaran, and A. Booperm, "the simplex algorithm improvement for unrestricted variable problem," *proceedings of the 5th KMITL-TKU international joint symposium on mathematics and applied mathematics*, pp. 51–60, 2016.

APPENDIX

The stopping criterion introduced in this section is the stopping criterion involved the number of jumps having the stopping parameter, τ. The average wall-clock time of SAJS varying $\tau = 2, 3, 4, 5, 6$ are shown in Table 1.

Table 1 demonstrates the average wall-clock time (in seconds) of SAJS with the different number of jumps. Row represents the number of constraints in the LP model, Col represents the number of variables in the LP model, the boldface numbers identify the smallest average wall-clock time and the number in parenthesis represents the standard deviations of each size of the LP model. For each size of the LP model, the results of the average wall-clock times in SAJS are not significantly different in each τ. Moreover, the average wall-clock times of SAJS with the stopping criterion which involved the improvement of the objective values and the stopping criterion which involved the number of jumps are not different.

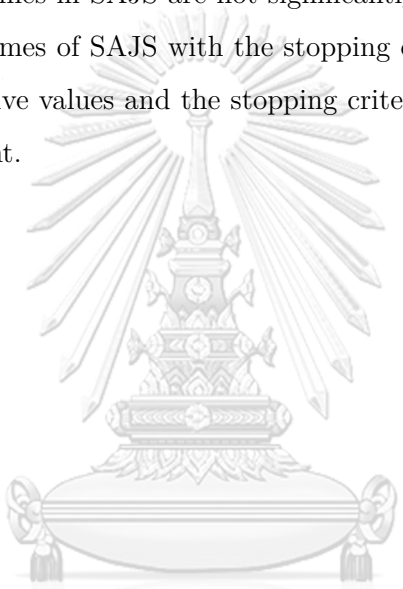


Table 1: SAJS performances varying τ.

Row	Col	The average wall-clock time of SAJS (sec.)				
		τ=2	τ=3	τ=4	τ=5	τ=6
100	10	0.0170 (±0.0059)	0.0158 (±0.0100)	0.0158 (±0.0102)	0.0196 (±0.0102)	**0.0095** (±0.0079)
	20	0.0400 (±0.0137)	0.0363 (±0.0134)	**0.0362** (±0.0120)	0.0425 (±0.0168)	0.0401 (±0.0173)
	30	0.0707 (±0.0147)	0.0708 (±0.0182)	**0.0651** (±0.0214)	0.0657 (±0.0199)	0.0658 (±0.0364)
	40	0.0843 (±0.0253)	0.0923 (±0.0339)	0.0893 (±0.0230)	**0.0776** (±0.0229)	0.0857 (±0.0313)
200	20	0.1043 (±0.0327)	0.0844 (±0.0241)	0.0859 (±0.0251)	**0.0797** (±0.0261)	0.0831 (±0.0278)
	40	0.2117 (±0.0247)	0.2336 (±0.0632)	0.2494 (±0.0612)	**0.2018** (±0.0384)	0.2196 (±0.0363)
	60	0.3821 (±0.0806)	0.3875 (±0.0952)	0.4047 (±0.0583)	0.3959 (±0.0330)	**0.3768** (±0.0703)
	80	0.5571 (±0.1011)	0.5540 (±0.1136)	0.5328 (±0.0695)	**0.4868** (±0.0844)	0.5093 (±0.1086)
300	30	**0.2665** (±0.0704)	0.3003 (±0.0725)	0.2880 (±0.0579)	0.3050 (±0.0602)	0.2935 (±0.0593)
	60	0.7232 (±0.1982)	0.7036 (±0.1175)	0.6855 (±0.0955)	0.6695 (±0.0860)	**0.6204** (±0.1101)
	90	1.1933 (±0.1918)	1.1013 (±0.1345)	1.0992 (±0.1836)	1.0469 (±0.1325)	**0.9922** (±0.1235)
	120	1.8667 (±0.2520)	**1.6408** (±0.1948)	1.6648 (±0.2347)	1.6854 (±0.2009)	1.7263 (±0.2927)

Continued on the next page

Table 1 – *Continued from the previous page*

Row	Col	The average wall-clock time of SAJS (sec.)				
		τ=2	τ=3	τ=4	τ=5	τ=6
400	40	**0.5423** (±0.1120)	0.5899 (±0.1382)	0.5907 (±0.0763)	0.5967 (±0.1175)	0.5781 (±0.1483)
	80	1.6659 (±0.3136)	1.5498 (±0.2145)	1.5333 (±0.1287)	1.5527 (±0.2084)	**1.4368** (±0.1533)
	120	2.9087 (±0.3820)	2.7183 (±0.2693)	2.7509 (±0.3367)	**2.6748** (±0.2508)	2.7418 (±0.2515)
	160	3.9900 (±0.3484)	3.9190 (±0.4911)	3.7316 (±0.4399)	**3.7197** (±0.3511)	3.8249 (±0.5119)
500	50	1.1520 (±0.2002)	1.1597 (±0.1407)	1.0289 (±0.1632)	1.0300 (±0.1391)	**0.9856** (±0.1716)
	100	**3.1884** (±0.2512)	3.2167 (±0.2524)	3.3571 (±0.4467)	3.3245 (±0.4692)	3.3864 (±0.3086)
	150	5.6986 (±0.8133)	5.4631 (±0.6083)	**5.2164** (±0.3883)	5.2896 (±0.4799)	5.6356 (±0.8237)
	200	7.4101 (±0.8956)	**7.3001** (±0.8457)	7.4175 (±0.7734)	7.6858 (±1.1085)	7.5085 (±1.3096)
Average		1.6036	1.5569	**1.5422**	1.5475	1.5560

Likewise, the average wall-clock times of AJSP with the stopping parameter which is defined as the number of jumps of the iterative jump method are tested and are shown in Table 2.

Table 2 demonstrates the average wall-clock time (in seconds) of AJSP varying $\tau = 2, 3, 4, 5, 6$. Row represents the number of constraints in the LP model, Col represents the number of variables in the LP model, the boldface numbers identify the smallest average wall-clock time and the number in parenthesis represents the standard deviations of each size of the LP model. The result in Table 2 shows that the average wall-clock time for each τ is rarely different. Moreover, it gives similar results to the stopping criterion which the stopping parameter is defined as least ratio improvement of two consecutive differences of the objective values.

Table 2: AJSP performances varying τ.

Row	Col	The average wall-clock time of AJSP (sec.)				
		τ=0.20	τ=0.30	τ=0.40	τ=0.50	τ=0.60
100	10	**0.0137** (±0.0049)	0.0169 (±0.0032)	0.0199 (±0.0099)	0.0239 (±0.0103)	0.0172 (±0.0064)
	20	0.0397 (±0.0129)	0.0350 (±0.0145)	0.0339 (±0.0155)	**0.0265** (±0.0103)	0.0275 (±0.0091)
	30	**0.0446** (±0.0136)	0.0539 (±0.0212)	0.0501 (±0.0147)	0.0587 (±0.0240)	0.0589 (±0.0190)
	40	**0.0685** (±0.0269)	0.0767 (±0.0155)	0.0818 (±0.0228)	0.0929 (±0.0301)	0.0794 (±0.0185)
200	20	0.0832 (±0.0261)	0.0762 (±0.0207)	0.0720 (±0.0233)	0.0786 (±0.0275)	**0.0697** (±0.0204)
	40	**0.1454** (±0.0430)	0.1816 (±0.0501)	0.1577 (±0.0499)	0.1762 (±0.0258)	0.1541 (±0.0376)
	60	0.2755 (±0.0276)	0.2539 (±0.0344)	0.2556 (±0.0427)	**0.2479** (±0.0319)	0.2594 (±0.0360)
	80	0.3531 (±0.0467)	**0.3315** (±0.0158)	0.3414 (±0.0331)	0.3469 (±0.0341)	0.3644 (±0.0557)
300	30	0.1939 (±0.0334)	**0.1655** (±0.0360)	0.2014 (±0.0452)	0.1867 (±0.0439)	0.1975 (±0.0412)
	60	0.3972 (±0.0491)	0.4055 (±0.0487)	**0.3931** (±0.0397)	0.4025 (±0.0460)	0.4097 (±0.0581)
	90	0.6896 (±0.1022)	0.6626 (±0.0691)	**0.6244** (±0.0500)	0.6875 (±0.1077)	0.6620 (±0.0772)
	120	1.0294 (±0.0719)	**1.0137** (±0.0879)	1.0155 (±0.0780)	1.0337 (±0.0819)	1.0336 (±0.0875)

Continued on the next page

Table 2 – *Continued from the previous page*

Row	Col	The average wall-clock time of AJSP (sec.)				
		τ=0.20	τ=0.30	τ=0.40	τ=0.50	τ=0.60
400	40	0.4003 (±0.0632)	0.4149 (±0.0615)	0.4202 (±0.0902)	**0.4001** (±0.0556)	0.4319 (±0.0754)
	80	0.8490 (±0.1309)	0.8678 (±0.0897)	0.8856 (±0.1074)	0.8435 (±0.0749)	**0.8396** (±0.0507)
	120	1.3967 (±0.0937)	1.3322 (±0.0801)	1.3500 (±0.0781)	1.3592 (±0.0983)	**1.3269** (±0.1123)
	160	1.8927 (±0.0990)	1.9588 (±0.1746)	1.9247 (±0.1509)	1.9163 (±0.0898)	**1.8812** (±0.1323)
500	50	0.7160 (±0.1111)	0.6788 (±0.0979)	**0.6405** (±0.1118)	0.6559 (±0.0652)	0.6768 (±0.0707)
	100	1.6133 (±0.2037)	1.5141 (±0.1354)	1.6098 (±0.2178)	**1.4939** (±0.1747)	1.5145 (±0.1967)
	150	2.3901 (±0.2002)	2.3922 (±0.1793)	2.4186 (±0.1998)	**2.2801** (±0.2121)	2.2915 (±0.2663)
	200	**3.4089** (±0.2878)	3.4330 (±0.1157)	3.4590 (±0.2176)	3.5054 (±0.2357)	3.4589 (±0.2173)
Average		0.8000	0.7933	0.7978	0.7908	**0.7877**

BIOGRAPHY

Name Miss Rujira Visuthirattanamanee

Date of Birth April 8, 1990

Place of Birth Ratchaburi, Thailand

Educations B.S. (Mathematics), Kasetsart University, 2011

M.Sc. (Applied Mathematics and Computational Science), Chulalongkorn University, 2013

Scholarships Science Achievement Scholarship of Thailand (SAST)

Publications

- R. Visuthirattanamanee, K. Sinapiromsaran and A. Boonperm, The simplex algorithm improvement for unrestricted variable problem, *Proceeding of the 5th KMITL-TKU International Joint Symposium On Mathematics and Applied Mathematics Conference* (2016), 51–60.
- R. Visuthirattanamanee, K. Sinapiromsaran and A. Boonperm, Self-regulating artificial-free linear programming solver using a jump and simplex method, Mathematics, vol. 8, pp. 356, 2020.