Linear Programming

Xuyang Li - xli0302@outlook.com

December 2020

Abstract

Briefly summarizes some important linear programming topics.

Acknowledgements

MAGA

Contents

List of Tables

Listings

1 Introduction

A linear programming problem is defined as the problem of optimizing a linear function subject to linear constraints.

1.1 Terminology

- **Objective Function**: The function to be optimized.
- **Constraint**: Nonnegative constraint and main constraint.
- **Feasible Vector**: A vector is feasible if it satisfies the corresponding constraints.
- **Feasible Problem**: A linear programming problem is feasible if the constraint set is not empty; otherwise, it is infeasible.
- **Bounded Problem**: A feasible maximum/minimum problem is unbounded if the objective function can assume arbitrarily large positive/negative values at feasible vectors; otherwise, it is bounded.
- **Vector**: $\boldsymbol{x} = \begin{pmatrix} x_1 & x_2 & \dots & x_N \end{pmatrix}^T \in \mathbb{R}^N$, $\boldsymbol{b} = \begin{pmatrix} b_1 & b_2 & \dots & b_M \end{pmatrix}^T \in \mathbb{R}^M$, $\boldsymbol{y} = \begin{pmatrix} y_1 & y_2 & \dots & y_M \end{pmatrix}^T \in \mathbb{R}^M$, $\boldsymbol{c} = \begin{pmatrix} c_1 & c_2 & \dots & c_N \end{pmatrix}^T \in \mathbb{R}^N$.
- **Optimal Vector**: $\boldsymbol{x}^\star = \begin{pmatrix} x_1^\star & x_2^\star & \dots & x_N^\star \end{pmatrix}^T$, $\boldsymbol{y}^\star = \begin{pmatrix} y_1^\star & y_2^\star & \dots & y_M^\star \end{pmatrix}^T$.
- **Matrix**: $\boldsymbol{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,N} \\ a_{2,1} & a_{2,2} & \dots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \dots & a_{M,N} \end{pmatrix}$.

1.2 The Standard Problem

Find the optimal vector $\boldsymbol{x}^\star$ to maximize the objective function 1

$$\boldsymbol{c}^T\boldsymbol{x} \tag{1}$$

subject to the main constraint 2

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{2}$$

and the nonnegative constraint 3.

$$\boldsymbol{x} \geq \boldsymbol{0} \tag{3}$$

All linear programming problems can be converted into the standard form like above. Two typical techniques are following.

1. *Some constraints may be equalities.*
   Each equality constraint $\sum_j^N a_{i,j}x_j = b_i$ can be replaced by two inequality constraints $\sum_j^N a_{i,j}x_j \leq b_i$ and $\sum_j^N (-a_{i,j})x_j \leq -b_i$. This removes one equality constraint and adds two inequality constraints to the problem.

2. *Some variables may not have nonnegative constraints.*
   An unrestricted variable x_j can be replaced by $x_j = x'_j - x''_j$, where $x'_j, x''_j \geq 0$. This adds one variable and two nonnegative constraints to the problem.

1.3 Four General Linear Programming Problems

1.3.1 The Diet Problem

There are M different types of food, that supply varying quantities of the N nutrients, that are essential to good health. Let c_j be the minimum daily requirement of nutrient N_j. Let b_i be the price per unit of food F_i. How to supply the required nutrients at minimum cost?

1.3.2 The Transportation Problem

There are M ports, that supply a certain commodity, and there are N markets, to which this commodity must be shipped. Port P_i processes an amount s_i of the commodity, and market M_j must receive the amount r_j of the commodity. Let $b_{i,j}$ be the cost of transporting one unit of the commodity from port P_i to market M_j. How to meet the market requirements at minimum transportation cost?

1.3.3 The Activity Analysis Problem

There are M activities, that a company may employ, using the available supply of N resources. Let b_i be the available supply of resource R_i. Let $a_{i,j}$ be the amount of resource R_i used in operating activity A_j at unit intensity. Let c_j be the net value to the company of operating activity A_j at unit intensity. How to choose the intensities which the various activities are to be operated to maximize the value of the output to the company subject to the given resources?

1.3.4 The Optimal Assignment Problem

There are M persons available for N jobs. Let $a_{i,j}$ be the value of person P_i working one day at job J_j. How to choose an assignment of persons to jobs to maximize the total value?

2 Duality

To every linear program there is a dual linear program with which it is intimately connected.

2.1 Definition

The dual of the standard problem 4

$$\begin{aligned} &find \quad \boldsymbol{x}^\star \quad to \quad maximize \quad \boldsymbol{c}^T\boldsymbol{x} \\ &s.t. \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \quad and \quad \boldsymbol{x} \geq \boldsymbol{0} \end{aligned} \tag{4}$$

is defined as 5.

$$\begin{aligned} &find \quad \boldsymbol{y}^\star \quad to \quad minimize \quad \boldsymbol{y}^T\boldsymbol{b} \\ &s.t. \quad \boldsymbol{y}^T\boldsymbol{A} \geq \boldsymbol{c}^T \quad and \quad \boldsymbol{y} \geq \boldsymbol{0} \end{aligned} \tag{5}$$

Table 1 below exhibits the relationship between the primal 4 and its dual 5.

Table 1: The Relationship between Primal and Dual

	x_1	x_2	$\dots$	x_N	
y_1	$a_{1,1}$	$a_{1,2}$	$\dots$	$a_{1,N}$	$\leq b_1$
y_2	$a_{2,1}$	$a_{2,2}$	$\dots$	$a_{2,N}$	$\leq b_2$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
y_M	$a_{M,1}$	$a_{M,2}$	$\dots$	$a_{M,N}$	$\leq b_M$
	$\geq c_1$	$\geq c_2$	$\dots$	$\geq c_N$	

Besides, there are three possibilities for a linear programming problem and its dual. Table 2 shows different types of the primal problem and the associated dual problem.

Table 2: Linear Programming Problem Types

Primal → Dual
Bounded Feasible → Bounded Feasible
Unbounded Feasible → Infeasible
Infeasible → Unbounded Feasible or Infeasible

2.2 The Duality Theorem

2.2.1 Weak Duality Theorem

If $\boldsymbol{x}$ is feasible for the standard problem 4 and if $\boldsymbol{y}$ is feasible for its dual 5, then inequality 6 holds.

$$\boldsymbol{c}^T\boldsymbol{x} \leq \boldsymbol{y}^T\boldsymbol{b} \tag{6}$$

2.2.2 Strong Duality Theorem

If the standard problem 4 is bounded feasible, then so is its dual 5. Their optimal values are equal, and equality 7 holds.

$$\boldsymbol{c}^T\boldsymbol{x}^\star = \boldsymbol{y}^{\star T}\boldsymbol{b} \tag{7}$$

2.3 The Complementary Slackness Conditions

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be feasible vectors for the primal 4 and its dual 5 respectively. Then, $\boldsymbol{x}$ and $\boldsymbol{y}$ are optimal vectors, if and only if the complementary slackness conditions 8 hold.

$$\begin{cases} \boldsymbol{y}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = \boldsymbol{0} \\ (\boldsymbol{y}^T\boldsymbol{A} - \boldsymbol{c})\boldsymbol{x} = \boldsymbol{0} \end{cases} \tag{8}$$

For example, there is a standard problem, where $\boldsymbol{A} = \begin{pmatrix} 1 & -1 & 1 \\ 2 & 1 & 3 \\ -1 & 0 & 2 \\ 1 & 1 & 1 \end{pmatrix}$, $\boldsymbol{b} = \begin{pmatrix} 4 & 6 & 3 & 8 \end{pmatrix}^T$, and $\boldsymbol{c} = \begin{pmatrix} 3 & 2 & 1 \end{pmatrix}^T$. Someone suspects the vector $\boldsymbol{x}^\star = \begin{pmatrix} 0 & 6 & 0 \end{pmatrix}^T$ is optimal. Is this conjecture correct?

Assume $\boldsymbol{x}^\star = \begin{pmatrix} 0 & 6 & 0 \end{pmatrix}^T$ is an optimal vector for the primal, and $\boldsymbol{y}^\star = \begin{pmatrix} y_1^\star & y_2^\star & y_3^\star & y_4^\star \end{pmatrix}^T$ is an optimal vector for its dual. Then, the complementary slackness conditions 8 implies the following equation 9.

$$\begin{cases} \boldsymbol{y}^{\star T}(\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}) = \begin{pmatrix} y_1^\star & y_2^\star & y_3^\star & y_4^\star \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ -3 \\ -2 \end{pmatrix} = \boldsymbol{0} \\ (\boldsymbol{y}^{\star T}\boldsymbol{A} - \boldsymbol{c})\boldsymbol{x}^\star = \begin{pmatrix} y_1^\star + 2y_2^\star - y_3^\star + y_4^\star - 3 \\ -y_1^\star + y_2^\star + y_4^\star - 2 \\ y_1^\star + 3y_2^\star + 2y_3^\star + y_4^\star - 1 \end{pmatrix}^T \begin{pmatrix} 0 \\ 6 \\ 0 \end{pmatrix} = \boldsymbol{0} \end{cases} \tag{9}$$

Solving equation 9 gives $\boldsymbol{y}^\star = \begin{pmatrix} 0 & 2 & 0 & 0 \end{pmatrix}^T$, and it is feasible on the constraints. Thus, $\boldsymbol{x}^\star$ and $\boldsymbol{y}^\star$ are optimal vectors, and the problem has an optimal value of $\boldsymbol{c}^T\boldsymbol{x}^\star = \boldsymbol{y}^{\star T}\boldsymbol{b} = 12$.

3 The Simplex Method

3.1 The Pivot Operation

Given a matrix $\boldsymbol{A}$ and a nonzero pivot $p = a_{h,k}$, the pivot operation transforms $\boldsymbol{A}$ to $\hat{\boldsymbol{A}}$ as equation 10 shows.

$$\hat{\boldsymbol{A}} = \begin{pmatrix} \hat{a}_{1,1} & \hat{a}_{1,2} & \dots & \hat{a}_{1,N} \\ \hat{a}_{2,1} & \hat{a}_{2,2} & \dots & \hat{a}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{a}_{M,1} & \hat{a}_{M,2} & \dots & \hat{a}_{M,N} \end{pmatrix} \quad \hat{a}_{i,j} = \begin{cases} \frac{1}{p} & i = h, j = k \\ \frac{a_{h,j}}{p} & i = h, j \neq k \\ -\frac{a_{i,k}}{p} & i \neq h, j = k \\ a_{i,j} - \frac{a_{h,j}a_{i,k}}{p} & i \neq h, j \neq k \end{cases} \tag{10}$$

3.2 The Simplex Tableau

The simplex tableau of a standard problem can be written as Table 3 below.

Table 3: The Simplex Tableau

	x_1	x_2	$\dots$	x_N	
y_1	$a_{1,1}$	$a_{1,2}$	$\dots$	$a_{1,N}$	b_1
y_2	$a_{2,1}$	$a_{2,2}$	$\dots$	$a_{2,N}$	b_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
y_M	$a_{M,1}$	$a_{M,2}$	$\dots$	$a_{M,N}$	b_M
	$-c_1$	$-c_2$	$\dots$	$-c_N$	0

Pivot around $a_{i,j}$ until the last row and the last column do not have negative items(exclusive of the bottom right corner). The minimum problem is obtained by letting the y_i on the left be zero and the y_i on the top equal to the corresponding items in the last row. While the maximum problem is obtained by letting the x_i on the top be zero and the x_i on the left equal to the corresponding items in the last column. Besides, the bottom right corner denotes the optimal value.

3.3 The Standard Simplex Method

At first, initialize the simplex tableau as Table 3, where the optimal value v must be set to 0. Assume after pivoting a while, someone obtains a simplex tableau as below. Then, there may occur three cases. Each case is discussed following.

	$\boldsymbol{r}^T$	
$\boldsymbol{t}$	$\boldsymbol{A}$	$\boldsymbol{b}$
	$-\boldsymbol{c}^T$	v

3.3.1 Case 1(Termination)

Condition: $\boldsymbol{b} \geq \mathbf{0}$ and $-\boldsymbol{c} \geq \mathbf{0}$.
Operation:

- For the standard problem, obtain $\boldsymbol{x}^\star$ from the equation $\begin{cases} \boldsymbol{r} = \mathbf{0} \\ \boldsymbol{t} = \boldsymbol{b} \end{cases}$.
- For the dual problem, obtain $\boldsymbol{y}^\star$ from the equation $\begin{cases} \boldsymbol{r} = -\boldsymbol{c} \\ \boldsymbol{t} = \mathbf{0} \end{cases}$.
- The optimal value v is at the bottom right corner.

3.3.2 Case 2(Negative Last Row Items)

Condition: $\boldsymbol{b} \geq \mathbf{0}$ but some $-c_j$ are negative.
Operation:

- Let k denote the pivot's column. Choose k that satisfies $-c_k < 0$.
- Let h denote the pivot's row. Choose $h = \arg\min_i \frac{b_i}{a_{i,k}}$, where $a_{i,k} > 0$.
- Pivot the tableau around $a_{h,k}$ by 10.

Property: After pivoting, the simplex tableau has the following properties.

- $\hat{\boldsymbol{b}} \geq \mathbf{0}$.
- $\hat{v} \geq v$.

Wrong Case: After picking the pivot's column k, if $a_{i,k} \leq 0$ holds for all i, then, the standard problem is unbounded feasible.

3.3.3 Case 3(Negative Last Column Items)

Condition: Some b_i are negative.
Operation:

- Let k denote the pivot's column, and f denote the first negative row of $\boldsymbol{b}$, say $b_f < 0$ and $b_1, b_2, \ldots, b_{f-1} \geq 0$. Then, choose k that satisfies $a_{f,k} < 0$
- Let h denote the pivot's row. Choose $h = \begin{cases} \arg\min_i \frac{b_i}{a_{i,k}} & \min \frac{b_i}{a_{i,k}} < \frac{b_f}{a_{f,k}} \\ f & else \end{cases}$, where $b_i \geq 0$ and $a_{i,k} > 0$.
- Pivot the tableau around $a_{h,k}$ by 10.

Property: After pivoting, the simplex tableau has the following properties.

- If $b_i \geq 0$, then $\hat{b}_i \geq 0$.

- $\hat{b}_f \geq b_f$.

Wrong Case: After picking the first negative row f, if $a_{f,j} \geq 0$ holds for all j, then, the standard problem is infeasible.

3.4 The Dual Simplex Method

Similar to the standard method, but it is stated with regard to the dual problem.

3.5 Cycling

Cycling may occur in the simplex method, say several pivot operations lead the current tableau back to the original one. Then the program will be stuck in an infinite loop. Cycling is rare in practical problems, and the smallest-subscript rule can avoid it by a simple modification to the simplex method.

3.6 Python Implementation

A python implementation is shown in Appendix A.

4 Four Problems with Non-Linear Objective Functions

4.1 Maximizing the Minimum of Values

The problem, so-called maximizing the minimum of values, is described as 11 below.

$$\begin{aligned} &find \quad \boldsymbol{x}^\star \quad to \quad maximize \quad \min_{i\in\{1,2,\ldots,p\}} \boldsymbol{c}_i^T\boldsymbol{x} \\ &s.t. \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \quad and \quad \boldsymbol{x} \ge \boldsymbol{0} \end{aligned} \tag{11}$$

Add a variable $\lambda \in \mathbb{R}$ that satisfies $\lambda \le \boldsymbol{c}_i^T\boldsymbol{x}$ for $i \in \{1, 2, \ldots, p\}$, which can be treated as the lower bound of the set $\{\boldsymbol{c}_i^T\boldsymbol{x}\}_{i\in\{1,2,\ldots,p\}}$. Then, by maximizing the lower bound λ, problem 11 can be transformed into a standard problem 12, where $\boldsymbol{C} = \begin{pmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \ldots & \boldsymbol{c}_p \end{pmatrix}$.

$$\begin{aligned} &find \quad \begin{pmatrix} \boldsymbol{x}^\star \\ \lambda^\star \end{pmatrix} \quad to \quad maximize \quad \begin{pmatrix} \boldsymbol{0}_{1\times N} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \lambda \end{pmatrix} \\ &s.t. \quad \begin{pmatrix} \boldsymbol{A} & \boldsymbol{0}_{M\times 1} \\ -\boldsymbol{C}^T & \boldsymbol{1}_{p\times 1} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \lambda \end{pmatrix} \le \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0}_{p\times 1} \end{pmatrix} \quad and \quad \boldsymbol{x} \ge \boldsymbol{0}_{N\times 1} \end{aligned} \tag{12}$$

Problem 12 has a linear objective function and linear constraints, but actually it is not standard, since λ is unrestricted. Fortunately, it can be replaced by $\lambda = \lambda' - \lambda''$, with $\lambda', \lambda'' \ge 0$.

4.2 Minimizing the Sum of Absolute Values

The problem, so-called minimizing the sum of absolute values, is described as 13 below, where $\theta_j \in \mathbb{R}$ for $j \in \{1, 2, \ldots, p\}$.

$$\begin{aligned} &find \quad \boldsymbol{y}^\star \quad to \quad minimize \quad \sum_{j=1}^{p} |\boldsymbol{y}^T\boldsymbol{b}_j - \theta_j| \\ &s.t. \quad \boldsymbol{y}^T\boldsymbol{A} \ge \boldsymbol{c}^T \quad and \quad \boldsymbol{y} \ge \boldsymbol{0} \end{aligned} \tag{13}$$

Add a vector, $\boldsymbol{y}' = \begin{pmatrix} y'_1 & y'_2 & \ldots & y'_p \end{pmatrix}^T \in \mathbb{R}^p$, where y'_j is the upper bound of $|\boldsymbol{y}^T\boldsymbol{b}_j - \theta_j|$. Then, for $j \in \{1, 2, \ldots, p\}$, there come p nonlinear constraints, $y'_j \ge |\boldsymbol{y}^T\boldsymbol{b}_j - \theta_j|$, which are equivalent to $2p$ linear constraints, $\boldsymbol{y}^T\boldsymbol{b}_j + y'_j \ge \theta_j$ and $-\boldsymbol{y}^T\boldsymbol{b}_j + y'_j \ge -\theta_j$. Also, the inequality $y'_j \ge |\cdot|$ implies that all y'_j are nonnegative. As a result, problem 13 can be transformed into 14 by minimizing $\sum_{j=1}^{p} y'_j$ which is the sum of these upper bounds, where $\boldsymbol{\theta} = \begin{pmatrix} \theta_1 & \theta_2 & \ldots & \theta_p \end{pmatrix}^T$, $\boldsymbol{B} = \begin{pmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \ldots & \boldsymbol{b}_p \end{pmatrix}$.

$$\begin{aligned} &find \quad \begin{pmatrix} \boldsymbol{y}^\star \\ \boldsymbol{y}'^\star \end{pmatrix} \quad to \quad minimize \quad \begin{pmatrix} \boldsymbol{y}^T & \boldsymbol{y}'^T \end{pmatrix} \begin{pmatrix} \boldsymbol{0}_{M\times 1} \\ \boldsymbol{1}_{p\times 1} \end{pmatrix} \\ &s.t. \quad \begin{pmatrix} \boldsymbol{y}^T & \boldsymbol{y}'^T \end{pmatrix} \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} & -\boldsymbol{B} \\ \boldsymbol{0}_{p\times N} & \boldsymbol{I}_{p\times p} & \boldsymbol{I}_{p\times p} \end{pmatrix} \ge \begin{pmatrix} \boldsymbol{c}^T & \boldsymbol{\theta}^T & -\boldsymbol{\theta}^T \end{pmatrix} \quad and \quad \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{y}' \end{pmatrix} \ge \boldsymbol{0}_{(M+p)\times 1} \end{aligned} \tag{14}$$

Problem 14 has linear a objective function and linear constraints, and it is the dual to a standard problem.

4.3 Minimizing the Maximum of Absolute Values

The problem, so-called minimizing the maximum of absolute values, is described as 15 below, where $\theta_j \in \mathbb{R}$ for $j \in \{1, 2, \ldots, p\}$.

$$\begin{aligned} &find \quad \boldsymbol{y}^\star \quad to \quad minimize \quad \max_{j\in\{1,2,\ldots,p\}} |\boldsymbol{y}^T\boldsymbol{b}_j - \theta_j| \\ &s.t. \quad \boldsymbol{y}^T\boldsymbol{A} \geq \boldsymbol{c}^T \quad and \quad \boldsymbol{y} \geq \boldsymbol{0} \end{aligned} \tag{15}$$

Since this objective function is similar to the combination of those two previous problems, add a variable $\mu \in \mathbb{R}$ which is the upper bound of the set $\{|\boldsymbol{y}^T\boldsymbol{b}_j - \theta_j|\}_{j\in\{1,2,\ldots,p\}}$. Then, for $j \in \{1, 2, \ldots, p\}$, there come p nonlinear constraints, $\mu \geq |\boldsymbol{y}^T\boldsymbol{b}_j - \theta_j|$, which are equivalent to $2p$ linear constraints, $\boldsymbol{y}^T\boldsymbol{b}_j + \mu \geq \theta_j$ and $-\boldsymbol{y}^T\boldsymbol{b}_j + \mu \geq -\theta_j$. Also, the inequality $\mu \geq |\cdot|$ implies that μ is nonnegative. Finally, by minimizing the upper bound μ, problem 15 can be transformed into a standard problem 16, where $\boldsymbol{\theta} = \begin{pmatrix}\theta_1 & \theta_2 & \ldots & \theta_p\end{pmatrix}^T$, $\boldsymbol{B} = \begin{pmatrix}\boldsymbol{b}_1 & \boldsymbol{b}_2 & \ldots & \boldsymbol{b}_p\end{pmatrix}$.

$$\begin{aligned} &find \quad \begin{pmatrix}\boldsymbol{y}^\star \\ \mu_\star\end{pmatrix} \quad to \quad minimize \quad \begin{pmatrix}\boldsymbol{y}^T & \mu\end{pmatrix}\begin{pmatrix}\boldsymbol{0}_{M\times 1} \\ 1\end{pmatrix} \\ &s.t. \quad \begin{pmatrix}\boldsymbol{y}^T & \mu\end{pmatrix}\begin{pmatrix}\boldsymbol{A} & \boldsymbol{B} & -\boldsymbol{B} \\ \boldsymbol{0}_{1\times N} & \boldsymbol{1}_{1\times p} & \boldsymbol{1}_{1\times p}\end{pmatrix} \geq \begin{pmatrix}\boldsymbol{c}^T & \boldsymbol{\theta}^T & -\boldsymbol{\theta}^T\end{pmatrix} \quad and \quad \begin{pmatrix}\boldsymbol{y} \\ \mu\end{pmatrix} \geq \boldsymbol{0}_{(M+1)\times 1} \end{aligned} \tag{16}$$

Problem 16 has linear a objective function and linear constraints, and it is the dual to a standard problem.

4.4 Linear Fractional Programming

The linear fractional programming problem 17 is described below, where $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^N$ and $\alpha, \beta \in \mathbb{R}$. To avoid technical difficulties, make two assumptions as the problem is bounded feasible, as well as the denominator $\boldsymbol{d}^T\boldsymbol{x} + \beta$ is strictly positive throughout the constraint set.

$$\begin{aligned} &find \quad \boldsymbol{x}^\star \quad to \quad maximize \quad \frac{\boldsymbol{c}^T\boldsymbol{x} + \alpha}{\boldsymbol{d}^T\boldsymbol{x} + \beta} \\ &s.t. \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \quad and \quad \boldsymbol{x} \geq \boldsymbol{0} \end{aligned} \tag{17}$$

Due to constraints listed above, it is possible to find a positive multiplication factor t to hold the denominator fixed, say $(\boldsymbol{d}^T\boldsymbol{x} + \beta)t = 1$ and $t > 0$. Then, with the change of variable $\boldsymbol{z} = t\boldsymbol{x}$, the objective function can be rewritten as $\frac{\boldsymbol{c}^T\boldsymbol{x}+\alpha}{\boldsymbol{d}^T\boldsymbol{x}+\beta} = \frac{(\boldsymbol{c}^T\boldsymbol{x}+\alpha)t}{(\boldsymbol{d}^T\boldsymbol{x}+\beta)t} = \boldsymbol{c}^T\boldsymbol{z} + \alpha t$, and the main constraint can be rewritten as $\boldsymbol{A}\boldsymbol{z} - t\boldsymbol{b} \leq \boldsymbol{0}$. To standardize the problem, the

equality constraint $(\boldsymbol{d}^T\boldsymbol{x}+\beta)t = \boldsymbol{d}^T\boldsymbol{z}+\beta t = 1$, can be replaced by two inequality constraints $\boldsymbol{d}^T\boldsymbol{z}+\beta t \leq 1$ and $-\boldsymbol{d}^T\boldsymbol{z}-\beta t \leq -1$. In conclusion, the linear fractional programming problem 17 is transformed as 18 below.

$$\begin{aligned} &find \quad \begin{pmatrix} \boldsymbol{z}^\star \\ t^\star \end{pmatrix} \quad to \quad maximize \quad \begin{pmatrix} \boldsymbol{c}^T & \alpha \end{pmatrix} \begin{pmatrix} \boldsymbol{z} \\ t \end{pmatrix} \\ &s.t. \quad \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{b} \\ \boldsymbol{d}^T & \beta \\ -\boldsymbol{d}^T & -\beta \end{pmatrix} \begin{pmatrix} \boldsymbol{z} \\ t \end{pmatrix} \leq \begin{pmatrix} \mathbf{0}_{M\times 1} \\ 1 \\ -1 \end{pmatrix} \quad and \quad \begin{pmatrix} \boldsymbol{z} \\ t \end{pmatrix} \geq \mathbf{0}_{(N+1)\times 1} \end{aligned} \tag{18}$$

Problem 18 has linear a objective function and linear constraints, and it is a standard problem.

A The Simplex Method Implementation

Listing 1 shows the implementation of the simplex method by python. It solves a linear programming problem as 4 describes.

```
import numpy as np


def simplex(A, b, c, r, t, v):
    """
    The Simplex Method with Regard to a Standard Problem
    - Initialize the Optimal Value v as 0
    - Determine tehe Current Case
    - Pivot until It Reaches the Base Case
    - Conclude the Final Results
    """

    M = np.shape(A)[0]
    N = np.shape(A)[1]

    """Determine Cases"""
    case = case_determination(b, c)

    if case == 1:  # Base Case
        solution(M, N, b, c, r, t, v)
        return
    else:  # Recursive Case
        h, k = pivot_index(A, b, c, M, N, case)
        if h < 0 or k < 0:
            # Protection
            # The problem can be infeasible or unbounded.
            return
        else:
            A, b, c, r, t, v = pivot(A, b, c, r, t, M, N, v, h, k)
            return simplex(A, b, c, r, t, v)


def case_determination(b, c):
    """Determine the Current Case in Terms of b and c."""

    # Case 1
    if b.min() >= 0 and c.min() >= 0:
        return 1
    # Case 2
    if b.min() >= 0 and c.min() < 0:
        return 2
    # Case 3
    if b.min() < 0:
        return 3


def pivot_index(A, b, c, M, N, case):
    """Find the pivot A[h, k] in cases 2 or 3."""

```

```
    """Case 2"""
    if case == 2:
        """Find the Column Index k"""
        for j in range(N):
            if c[j] < 0:
                k = j
                break

        """Find the Row Index h"""
        # Compute the values of b[i] / A[i, k]
        h_value = []
        h_index = []
        for i in range(M):
            if A[i, k] > 0:
                h_value.append(b[i] / A[i, k])
                h_index.append(i)

        if h_value:
            h = h_index[h_value.index(min(h_value))]
        else:
            # Protection
            print("The standard problem is unbounded feasible.")
            return -1, -1

        return h, k

    """Case 3"""
    if case == 3:
        """Find the First Negative Row in b"""
        for i in range(M):
            if b[i] < 0:
                f = i
                break

        """Find the Column Index k"""
        k = -1
        for j in range(N):
            if A[f, j] < 0:
                k = j
                break
        # Protection
        if k == -1:
            print("The standard problem is infeasible.")
            return -1, -1

        """Find the Row Index h"""
        # Compute the values of b[i] / A[i, k]
        f_value = b[f] / A[f, k]
        h_value = []
        h_index = []
        for i in range(N):
            if b[i] >= 0 and A[i, k] > 0:
                h_value.append(b[i] / A[i, k])
                h_index.append(i)
```

```
104
105         if not h_value:
106             h = f
107         else:
108             if f_value < min(h_value):
109                 h = f
110             else:
111                 h = h_index[h_value.index(min(h_value))]
112
113         return h, k
114
115
116 def pivot(A, b, c, r, t, M, N, v, h, k):
117     """Pivot the Simplex Tableau"""
118
119     """Combine Matrices"""
120     S = np.empty((M + 1, N + 1))
121     S[:M, :N] = A
122     S[:M, N] = b
123     S[M, :N] = c
124     S[M, N] = v
125
126     """Pivot Operation"""
127     S_hat = np.empty((M + 1, N + 1))
128     p = S[h, k]
129     for i in range(M + 1):
130         for j in range(N + 1):
131             if i == h and j == k:
132                 S_hat[i, j] = 1 / p
133             if i == h and j != k:
134                 S_hat[i, j] = S[h, j] / p
135             if i != h and j == k:
136                 S_hat[i, j] = -S[i, k] / p
137             if i != h and j != k:
138                 S_hat[i, j] = S[i, j] - S[h, j] * S[i, k] / p
139
140     """Decompose Matrix"""
141     A = S_hat[:M, :N]
142     b = S_hat[:M, N]
143     c = S_hat[M, :N]
144     v = S_hat[M, N]
145
146     """Pivot r and t"""
147     swap = np.empty((1, 2))
148     swap[0, :] = r[k]
149     r[k] = t[h]
150     t[h] = swap
151
152     return A, b, c, r, t, v
153
154
155 def solution(M, N, b, c, r, t, v):
156     """Process r, t, b, and c to find the solution(optimal vector/value)."""
157
```

```
    """The Standard Problem"""
    print("Optimal Vector x for the Primal:")
    for i in range(M):
        if t[i, 0] == 0:
            print("x_" + str(int(t[i, 1])) + " = " + str(b[i]))
    for j in range(N):
        if r[j, 0] == 0:
            print("x_" + str(int(r[j, 1])) + " = 0")

    """The Dual Problem"""
    print("Optimal Vector y for the Dual:")
    for i in range(M):
        if t[i, 0] == 1:
            print("y_" + str(int(t[i, 1])) + " = 0")
    for j in range(N):
        if r[j, 0] == 1:
            print("y_" + str(int(r[j, 1])) + " = " + str(c[j]))

    """Value"""
    print("The Optimal Value: " + str(v))

    return


def initialization(A):
    """Initialize r, t, and v"""

    M = np.shape(A)[0]
    N = np.shape(A)[1]

    """
    r & t
    - The First Column: 0 denotes x; 1 denotes y.
    - The Second Column: Denotes the subscript of x or y.
    """
    r = np.empty((N, 2))
    r[:, 0] = 0
    r[:, 1] = range(1, N + 1)

    t = np.empty((M, 2))
    t[:, 0] = 1
    t[:, 1] = range(1, M + 1)

    v = 0

    return r, t, v


"""
    Test Samples
    Remind: enter A, b as usually, but enter -c into the variable c.
    For example, try to minimize
    x_{1} + x_{2}
    s. t. ([1, 2], [3, 4])x <= (5, 6) and x >= 0
```

```
    then
    A = np.array([
        [1, 2],
        [3, 4],
    ])
    b = np.array([5, 6])
    c = np.array([-1, -1])
"""
if __name__ == '__main__':
    # Sample 1
    A = np.array([
        [2, 1, -7],
        [-1, 0, 4],
        [1, 2, -6],
    ])
    b = np.array([3, -1, 2])
    c = np.array([1, -2, -1])

    # Sample 2
    A = np.array([
        [1, -1, -2, -1],
        [2, 0, 1, -4],
        [-2, 1, 0, 1],
    ])
    b = np.array([4, 2, 1])
    c = np.array([-1, 2, 3, 1])

    # Sample 3
    A = np.array([
        [2, 3, 4, 0],
        [-1, -2, -3, 1],
        [-2, -1, -3, 1],
        [-1, -3, -2, 1],
    ])
    b = np.array([12, 0, 0, 0])
    c = np.array([0, 0, 0, -1])

    # Implementation
    r, t, v = initialization(A)
    simplex(A, b, c, r, t, v)
```

Listing 1: The Simplex Method