

**ĐẠI HỌC ĐÀ NẴNG
ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN**



ĐỒ ÁN

GIẢI THUẬT & LẬP TRÌNH

**Đề tài: Tìm hiểu các phương pháp giải bài toán
quy hoạch tuyến tính & xây dựng ứng dụng**



**GVHD : TS. Nguyễn Văn Hiệu
SVTH : Lê Trọng Hiếu
LỚP : 15TCLC2
NHÓM : 15.NH15**

Đà Nẵng 12-2017

LỜI MỞ ĐẦU

Lịch sử của Toán học gắn liền với sự phát triển của loài người, những khái niệm được hình thành hầu hết xuất phát từ đời sống thực tiễn, từ nhu cầu tìm tòi và khám phá của con người. Thời xưa khi con người chưa có sự hỗ trợ của máy móc nên bản thân các bài toán phát sinh chỉ là các bài đơn giản, số lượng tính toán là cỡ nhỏ, vì vậy các công cụ toán để sử dụng cũng là những công thức vô cùng đơn giản và sơ khai như phép cộng, phép chia, hay khai căn một cách gần đúng... Ngày nay cùng với sự tiến bộ về kinh tế - xã hội của loài người, toán học từng bước phát triển nhảy vọt. Những yêu cầu cấp bách về sự phát triển nền kinh tế và quốc phòng làm nảy sinh những ý tưởng về sự tính toán các hành động đem lại lợi ích cao nhất trong những điều kiện nhất định. Chính vì sự cấp bách này, bài toán tìm phương án tối ưu đã ra đời.

Để giải quyết một cách có hiệu quả bài toán tối ưu chúng ta phải xây dựng một mô hình toán học cho nó, trên đó thể hiện được bản chất của mỗi đối tượng đã được khảo sát và các mối liên hệ ràng buộc giữa chúng; ngoài ra phải chỉ rõ mục tiêu mong muốn đạt được. Bài toán tìm phương án tối ưu với mô hình toán học đã được xây dựng gọi là bài toán quy hoạch toán học hay bài toán tối ưu.

Năm 1947, George Bemanrd Dantzig khai sinh ra quy hoạch tuyến tính cùng với phương pháp đơn hình là một lĩnh vực con chính trong bài toán tối ưu hóa. Phương pháp của ông đã làm thay đổi bộ mặt của nền kinh tế sản xuất, vận tải và được ứng dụng rộng rãi trong cuộc sống.

Đồ án I – Giải thuật và lập trình của em với đề tài: “Tìm hiểu các phương pháp giải bài toán quy hoạch tuyến tính và xây dựng ứng dụng” gồm 5 chương:

Chương I : Giới thiệu đề tài

Chương II : Cơ sở lý thuyết

Chương III : Tổ chức cấu trúc dữ liệu và thuật toán

Chương IV : Chương trình và kết quả

Chương V : Kết luận và hướng phát triển

Trong quá trình thực hiện đề tài em luôn cố gắng tận dụng những kiến thức đã học ở trường cùng với sự tìm tòi, nghiên cứu. Em xin chân thành cảm ơn các thầy, cô giáo bộ môn và đặc biệt là TS. Nguyễn Văn Hiệu đã tận tình hướng dẫn em hoàn thành đề tài này.

Do thời gian làm đồ án có hạn và trình độ còn nhiều hạn chế nên không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến của các thầy, cô giáo cũng như các bạn sinh viên để bài đồ án này hoàn thiện hơn nữa.

Một lần nữa em xin chân thành cảm ơn

. Đà Nẵng, tháng 12 năm 2017

Sinh viên thực hiện

Lê Trọng Hiếu

Nhận xét của giáo viên hướng dẫn

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Đà Nẵng, ngày ... tháng ... năm 2017

Giáo viên hướng dẫn

TS. Nguyễn Văn Hiệu

Nhận xét của giáo viên phản biện

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Đà Nẵng, ngày ... tháng ... năm 2017

Giáo viên phản biện

MỤC LỤC

LỜI MỞ ĐẦU	2
MỤC LỤC.....	5
TIÊU ĐỀ	7
I. GIỚI THIỆU ĐỀ TÀI.....	7
I.1. Tên đề tài.....	7
I.2. Lý do chọn đề tài.....	7
I.3. Mục đích của đề tài	7
II. CƠ SỞ LÝ THUYẾT	8
II.1. Ý tưởng	8
II.2. Cơ sở lý thuyết.....	8
III. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN	16
III.1. Phát biểu bài toán	16
III.2. Cấu trúc dữ liệu	18
III.3. Thuật toán.....	18
III.3.1. Các hàm sử dụng trong thuật toán.....	18
III.3.2. Các biến sử dụng trong thuật toán.....	18
III.3.3. Sơ đồ khối thuật toán.....	20
III.3.4. Độ phức tạp thuật toán	23
IV. CHƯƠNG TRÌNH VÀ KẾT QUẢ.....	24
IV.1. Tổ chức chương trình.....	24
IV.2. Ngôn ngữ cài đặt	24
IV.3. Kết quả	24
IV.3.1. Giao diện chính của chương trình	24
IV.3.2. Kết quả thực thi của chương trình.....	26
IV.3.3. Nhận xét	27
V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	28
V.1. Kết luận.....	28
V.2. Hướng phát triển.....	28
TÀI LIỆU THAM KHẢO.....	29
PHỤ LỤC.....	30

DANH MỤC HÌNH VẼ

Hình 1. File text input	16
Hình 2. File text output	17
Hình 3. Sơ đồ khối thuật toán	20
Hình 4. Giao diện chính của chương trình	24
Hình 5. Thông tin bài toán được tự động điền vào DataGridView	25
Hình 6. Tạo lập DataGridView để nhập dữ liệu	25
Hình 7. Chạy chương trình thực tế	26

DANH MỤC BẢNG

Bảng 1. Danh sách các từ viết tắt	6
Bảng 2. Quy tắc thành lập bài toán đối ngẫu	9
Bảng 3. Danh sách các biến sử dụng	19
Bảng 4. File text input bài toán mẫu	26
Bảng 5. File text output bài toán mẫu	27

DANH SÁCH CÁC TỪ VIẾT TẮT

Từ viết tắt	Nghĩa tương minh
QHTT	Quy hoạch tuyến tính
PATƯ	Phương án tối ưu
PACB	Phương án cực biên

Bảng 1. Danh sách các từ viết tắt

TIÊU ĐỀ: SỬ DỤNG PHƯƠNG PHÁP ĐƠN HÌNH GIẢI QUYẾT BÀI TOÁN QUY HOẠCH TUYẾN TÍNH & XÂY DỰNG ỨNG DỤNG

I. GIỚI THIỆU ĐỀ TÀI

I.1. Tên đề tài

Quy hoạch tuyến tính (QHTT- Linear Programming) là một kỹ thuật toán học nhằm xác định các biến x_1, x_2, \dots, x_n (được gọi là các biến quyết định) sao cho:

- Cực đại hóa hoặc cực tiểu hóa giá trị của hàm mục tiêu (Objective function): $Z = f(x_1, x_2, \dots, x_n)$
- Thỏa mãn các ràng buộc (Constraints) : $R_j = r_j(x_1, x_2, \dots, x_n)$

Ta nghiên cứu các bài toán QHTT dạng tổng quát với **n ẩn, m ràng buộc** có dạng:

$$\begin{cases} f(x_1, x_2, \dots, x_n) = c_1x_1 + c_2x_2 + \dots + c_nx_n \rightarrow \max \text{ (min)} & (1) \\ a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \begin{cases} \geq \\ = \\ \leq \end{cases} b_i, i = \overline{1, m} & (2) \\ x_j \geq 0, j = \overline{1, n} & (3) \end{cases}$$

Trong đó :

- (1) là hàm mục tiêu (Objective Function)
- (2) là hệ ràng buộc chính (Constraints)
- (3) là ràng buộc dấu
- (2) và (3) là hệ ràng buộc của bài toán

I.2. Lý do chọn đề tài

Kiến thức sau khi học QHTT rất cần thiết, đây là những kiến thức rất quan trọng để xây dựng một mô hình toán học cho bất kỳ bài toán phức tạp nào trong thực tế, chỉ cần xây dựng các thuật toán đã mô hình hóa ngôn ngữ nhờ việc lập trình trên máy tính ta có thể giải quy hoạch tuyến tính một cách dễ dàng nhanh chóng và chính xác. Như vậy, QHTT rất quan trọng, nó đem lại những hiệu quả kinh tế rất lớn nếu biết lập các mô hình và tính toán đúng quy cách. Chính vì những lợi ích to lớn này là lý do mà em chọn đề tài QHTT để nghiên cứu.

I.3. Mục đích của đề tài

Hiểu rõ một trong những phương pháp giải bài toán QHTT, cụ thể ở đây là **phương pháp đơn hình**. Xây dựng được thuật toán và ứng dụng để giải bài toán QHTT. Áp dụng để giải các bài toán trong thực tế.

Đồng thời tìm hiểu sơ lược qua các phương pháp, kỹ thuật khác có thể được sử dụng để giải quyết bài toán QHTT.

II. CƠ SỞ LÝ THUYẾT

II.1. Ý tưởng

Sử dụng **thuật toán đơn hình hai pha** để giải bài toán QHTT cực đại hóa hàm mục tiêu. Nếu là bài toán QHTT cực tiểu hóa hàm mục tiêu thì **chuyển về bài toán đối ngẫu** rồi giải quyết như bài toán cực đại hóa.

II.2. Cơ sở lý thuyết

II.1.1. Tìm bài toán đối ngẫu

Vì sử dụng phương pháp đơn hình trên bài toán QHTT cực đại hóa hàm mục tiêu (Maximization) nên ta chỉ xét vấn đề tìm bài toán đối ngẫu của bài toán QHTT cực tiểu hóa (Minimization).

Cho (P) là bài toán QHTT có dạng chính tắc như sau :

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= c_1x_1 + c_2x_2 + \dots + c_nx_n \rightarrow \min \\ a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &\geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &\geq b_2 \\ &\dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &\geq b_m \\ x_j &\geq 0, j = \overline{1, n} \end{aligned}$$

Từ bài toán (P) ta lập được bài toán (D) như sau và gọi bài toán (D) là bài toán đối ngẫu của bài toán (P)

$$\begin{aligned} f(y_1, y_2, \dots, y_m) &= b_1y_1 + b_2y_2 + \dots + b_my_m \rightarrow \max \\ a_{11}y_1 + a_{21}y_2 + \dots + a_{m1}y_m &\leq c_1 \\ a_{12}y_1 + a_{22}y_2 + \dots + a_{m2}y_m &\leq c_2 \\ &\dots\dots\dots \\ a_{1n}y_1 + a_{2n}y_2 + \dots + a_{mn}y_m &\leq c_n \\ y_i &\geq 0, i = \overline{1, m} \end{aligned}$$

Chú ý. Do tính đối xứng của cặp bài toán đối ngẫu nên khái niệm bài toán gốc và bài toán đối ngẫu chỉ mang tính tương đối, nghĩa là nếu bài toán này là bài toán gốc thì bài toán kia là bài toán đối ngẫu và ngược lại.

Bài toán (D) được lập từ bài toán (P) theo nguyên tắc sau:

- Số ẩn của bài toán (D) bằng số ràng buộc chính của bài toán (P) và số ràng buộc chính của bài toán (D) bằng số ẩn của bài toán (P).
- Hệ số của ẩn y_i trong hàm mục tiêu của bài toán (D) là số hạng tự do b_i trong hệ ràng buộc chính của bài toán (P).
- Các hệ số của các ẩn và hệ số tự do trong ràng buộc chính thứ j của bài toán (D) là các hệ số tương ứng của ẩn x_j trong hệ ràng buộc chính và hàm mục tiêu của bài toán (P).
- Nếu (P) là bài toán max thì (D) là bài toán min và hệ ràng buộc chính của bài toán (D) là hệ bất phương trình với dấu \geq . Nếu (P) là bài toán min thì (D) là bài toán max và hệ ràng buộc chính của bài toán (D) là hệ bất phương trình với dấu \leq .
- Các ẩn của bài toán (D) đều có dấu tùy ý.

Lược đồ tổng quát các quy tắc thành lập bài toán đối ngẫu của bài toán QHTT tối thiểu hóa hàm mục tiêu:

Bài toán gốc $f(X) = \sum_{i=1}^n c_j x_j \rightarrow \min$	Bài toán đối ngẫu: $g(Y) = \sum_{i=1}^m b_i y_i \rightarrow \max$
<ul style="list-style-type: none"> • Nếu $x_j \geq 0$ • Nếu $x_j \leq 0$ • Nếu x_j không ràng buộc về dấu • Nếu $\sum_{j=1}^n a_{ij} x_j = b_i$ • Nếu $\sum_{j=1}^n a_{ij} x_j \leq b_i$ • Nếu $\sum_{j=1}^n a_{ij} x_j \geq b_i$ 	<ul style="list-style-type: none"> • thì $\sum_{j=1}^m a_{ij} y_i \leq c_j$ • thì $\sum_{j=1}^m a_{ij} y_i \geq c_j$ • thì $\sum_{i=1}^m a_{ij} y_i = c_j$ • thì y_i không ràng buộc về dấu • thì $y_i \leq 0$ • thì $y_i \geq 0$

Bảng 2. Quy tắc thành lập bài toán đối ngẫu

II.2.2. Từ bài toán QHTT dạng tổng quát về dạng chính tắc

Khi tìm được phương án tối ưu (PATU) của bài toán dạng chính tắc ta chỉ cần tính giá trị của các ẩn ban đầu và bỏ đi các ẩn phụ thì ta sẽ được PATU của bài toán dạng tổng quát đã cho.

Bài toán quy hoạch tuyến tính dạng chính tắc với n ẩn, m ràng buộc có dạng:

$$\begin{cases} f(X) = c_1x_1 + c_2x_2 + \dots + c_nx_n \rightarrow \max \text{ (min)} & (4) \end{cases}$$

$$\begin{cases} a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i, \quad i = \overline{1, m} & (5) \end{cases}$$

$$\begin{cases} x_j \geq 0, \quad j = \overline{1, n} & (6) \end{cases}$$

Nhận xét. Bài toán QHTT dạng chính tắc là bài toán QHTT dạng tổng quát trong đó:

- Các ràng buộc chính đều là phương trình.
- Các ẩn đều không âm.

Ta có thể biến đổi bài toán dạng tổng quát về dạng chính tắc bằng các bước sau:

Bước 1. Kiểm tra hệ ràng buộc chính

- Nếu có ràng buộc chính dạng:

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \leq b_i$$

thì ta cộng vào vế trái ràng buộc đó ẩn phụ x_{n+k} , nghĩa là thay ràng buộc trong bài toán bằng ràng buộc:

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n + x_{n+k} = b_i$$

- Nếu có ràng buộc chính dạng:

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \geq b_i$$

thì ta trừ vào vế trái ràng buộc đó ẩn phụ x_{n+k} , nghĩa là thay ràng buộc trong bài toán bằng ràng buộc:

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n - x_{n+k} = b_i$$

Chú ý. Các ẩn phụ là không âm và hệ số của các ẩn phụ trong hàm mục tiêu bằng 0.

Bước 2. Kiểm tra điều kiện dấu của ẩn số

- Nếu có ẩn $x_j \leq 0$ thì ta thực hiện phép đổi ẩn số $x_j = -x_j'$ với $x_j' \geq 0$.
- Nếu có ẩn x_j nào đó không có ràng buộc về dấu ($x_j \in \mathbb{R}$) thì ta thay x_j bởi hai biến phụ không âm $x_j^+ \geq 0$ và $x_j^- \geq 0$ sao cho:

$$x_j^+ + x_j^- = 0$$

II.2.3 Phương pháp đơn hình mở rộng giải bài toán QHTT dạng chính tắc

a. Các tính chất cơ bản của bài toán QHTT dạng chính tắc

- Giả sử vector x thỏa mãn mọi ràng buộc (hệ (5), (6)) của bài toán thì được gọi là phương án, thỏa mãn chặt là thỏa mãn với dấu “=” còn thỏa mãn lỏng là thỏa mãn với dấu bất đẳng thức.
- Phương Án Cực Biên: là phương án thỏa mãn chặt n ràng buộc độc lập tuyến tính. PACB thỏa mãn chặt đúng n (số nghiệm của bài toán) ràng buộc được gọi là PACB không suy biến, còn thỏa mãn chặt hơn n ràng buộc được gọi là PACB suy biến.
- Phương Án Tối Ưu: là phương án mà tại đó hàm mục tiêu $f(x)$ đạt cực tiểu hay cực đại (hay là phương án tốt nhất)
- Bài toán giải được và không giải được:
 - Bài toán giải được là bài toán có PATU, tức là có 1 vector x thỏa mãn (4),(5),(6).
 - Bài toán không giải được là bài toán không có phương án hoặc có phương án nhưng hàm mục tiêu không bị chặn (tăng hoặc giảm vô cùng).
- Số PACB của 1 bài toán luôn là hữu hạn.

Về sự tồn tại PATU của bài toán QHTT: Nếu bài toán QHTT có tập phương án khác rỗng và hàm mục tiêu bị chặn trên đối với bài toán $f(X) \rightarrow \max$ (bị chặn dưới đối với bài toán $f(X) \rightarrow \min$) trong tập các phương án thì bài toán có phương án tối ưu.

b. Bài toán QHTT chính tắc dưới dạng ma trận

Để sử dụng phương pháp đơn hình, ta cần chuyển bài toán QHTT sang dạng ma trận, từ đó thực hiện các phép xoay bảng để tìm PATU.

Bài toán dạng chính tắc:

$$\begin{cases} Z = c_1x_1 + c_2x_2 + \dots + c_nx_n \rightarrow \max \\ a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i, \quad i = \overline{1, m} \\ x_j \geq 0, \quad j = \overline{1, n} \end{cases}$$

Được viết lại dưới dạng:

$$\left\{ \begin{array}{l} Z = C X \rightarrow \max \\ \sum_{j=1}^n X_j A_j = B \\ X \geq 0 \end{array} \right. \quad \text{hay gọn hơn} \quad \left\{ \begin{array}{l} Z = C X \rightarrow \max \\ A X = B \\ X \geq 0 \end{array} \right.$$

Với:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \quad A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix} \quad C = \begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{bmatrix} \quad A_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \dots \\ a_{mj} \end{bmatrix}$$

c. Bảng đơn hình

c.1. Các thành phần của bảng đơn hình

- A là ma trận hệ số của các ràng buộc.
- B là ma trận các giá trị ở vế phải của các ràng *buộc* (còn được gọi là các giá trị *lambda* λ).
- Giá trị của hàm mục tiêu Z.
- X' là ma trận các hệ số của hàm mục tiêu được xây dựng sau khi di chuyển tất cả các biến ở hàm mục tiêu sang bên trái (nghĩa là ta đưa hàm mục tiêu về dạng $Z - C X = 0$), khi đó:

$$X' = \begin{bmatrix} -x_1 \\ -x_2 \\ \dots \\ -x_n \end{bmatrix}$$

Ta sẽ có cấu trúc của bảng đơn hình:

a_{11}	a_{12}	\dots	a_{1n}	b_1
a_{21}	a_{22}	\dots	a_{2n}	b_2
\dots				\dots
a_{m1}	a_{m2}	\dots	a_{mn}	b_m
$-c_1$	$-c_2$	\dots	$-c_n$	Z

Để thuận tiện trong việc trình bày, ta quy ước rằng:

cột $\{b_1, b_2, \dots, b_m\}$ là cột lambda λ

hàng $\{-c_1, -c_2, \dots, -c_n\}$ là hàng delta Δ

c.2. Dấu hiệu tối ưu

Theo tính chất của bài toán QHTT cực đại hóa hàm mục tiêu, ta có định lý về dấu hiệu tối ưu:

Nếu $X_o = \{x_{1o}, x_{2o}, \dots, x_{mo}, 0, \dots, 0\}$ là một PACB sao cho:

$$\Delta_j \geq 0, \forall j = \overline{1, n}$$

thì X_o là PATU của bài toán.

Như vậy, ta có thể nhận ra bảng đơn hình cuối cùng của bài toán QHTT là bảng mà mọi giá trị Δ đều không âm.

c.3. Dấu hiệu bài toán không giải được

Khái niệm tỉ số dòng trong bảng đơn hình: Tỉ số của các dòng là kết quả của phép chia giá trị ở cột Lambda λ với hệ số ràng buộc trong ma trận A ở hàng tương ứng và cột có giá trị Δ âm nhất (hoặc nhỏ nhất) trong bảng đơn hình.

Ta quy ước rằng tập hợp:

$$r = \{r_1, r_2, \dots, r_m\}$$

là tập hợp chứa các giá trị tỉ lệ dòng.

Theo tính chất của bài toán QHTT cực đại hóa hàm mục tiêu, ta có định lý về dấu hiệu bài toán không giải được:

Nếu tồn tại tập hợp $r = \{r_1, r_2, \dots, r_m\}$ với phương án cực biên $X_o = \{x_{1o}, x_{2o}, \dots, x_{mo}, 0, \dots, 0\}$ mà:

$$r_j < 0, \forall j = \overline{1, m}$$

thì bài toán không có phương án tối ưu.

c.4. Vị trí cột mốc (phần tử trục- pivot)

Nếu bảng đơn hình không thỏa mãn các dấu hiệu tối ưu và dấu hiệu bài toán không giải được thì chúng ta sẽ dựa vào giá trị cột mốc để thực hiện phép xoay thay đổi bảng đơn hình.

Giá trị cột mốc p_{ij} được chọn trong ma trận hệ số các ràng buộc A sao cho

$$i \in [1, n] \text{ với } r_i > 0, r_i = \min(r)$$

$$\text{và } j \in [1, m] \text{ với } \Delta_j < 0, \Delta_j = \min(\Delta).$$

Hay nói cách khác, p_{ij} có tọa độ hàng tương ứng với giá trị tỉ lệ hàng r_i dương nhỏ nhất, và tọa độ cột tương ứng với giá trị delta Δ_j âm nhất.

Hàng và cột ứng với giá trị cột mốc lần lượt được gọi là **hàng xoay** và **cột xoay**.

d. Quy tắc xoay bảng

Từ cơ sở lý luận của phương pháp đơn hình đã được trình bày ta thấy rằng xuất phát từ phương án cực biên ban đầu X_o được coi như đã biết, sau khi kiểm tra nếu thấy chưa thỏa mãn dấu hiệu tối ưu và dấu hiệu bài toán không giải được thì ta phải tiến hành xây dựng một phương án cực biên mới X_1 tốt hơn X_o và kiểm tra nó. Quá trình đó sau một số hữu hạn bước sẽ kết thúc, lúc đó ta tìm được phương án tối ưu hoặc kết luận bài toán không giải được. Ta nói rằng đã thực hiện được một *bước lặp* nếu như mỗi phương án xuất hiện trong quá trình nói trên, sau khi đã kiểm tra xong ta có được một quyết định tiếp theo.

Trong mỗi bước lặp ta cần xác định các giá trị thay đổi ở bảng đơn hình, giá trị cột mốc nếu có, Z (giá trị hàm mục tiêu tại PACB đang xét), trong đó việc xác định các giá trị hệ số các ràng buộc ở ma trận A là khó khăn nhất.

Chi tiết các công việc để thực hiện việc xoay bảng (*thay đổi các giá trị trong bảng đơn hình*) có thể được trình bày tuần tự như sau:

- Định vị giá trị âm nhất Δ_j ở hàng Δ . Giá trị này xác định cột xoay.
- Định vị giá trị tỉ lệ hàng dương nhỏ nhất r_i . Giá trị này xác định hàng xoay.
- Sau khi xác định được giá trị cột mốc p ta thực hiện phép thay đổi bảng đơn hình.

Giả sử rằng ta tìm được hàng xoay h_p , cột xoay $c_{p'}$. Ta biến đổi các hàng xoay:

$$h_i, i \in [1, m] \setminus \{p\}$$

sao cho các giá trị ở cột xoay $c_{p'}$ khác với giá trị cột mốc p , nghĩa là các giá trị:

$$a_{ip'} = 0, i \in [1, m] \setminus \{p\}$$

Ta có thể thay đổi các hàng h_i này theo công thức:

$$h_i \rightarrow -\frac{p}{a_{ip'}} * h_p + h_i$$

Ta cũng thực hiện công thức tương tự với các giá trị ở cột λ và hàng Δ :

$$\lambda_i \rightarrow -\frac{p}{a_{ip'}} * \lambda_p + \lambda_i$$

$$\Delta \rightarrow -\frac{p}{\Delta_{p'}} * h_p + \Delta$$

Sau khi thực hiện các công việc trên, ta đã thay đổi được bảng đơn hình. Quá trình này lặp đi lặp lại cho đến khi ta tìm được bảng đơn hình cuối cùng.

III. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

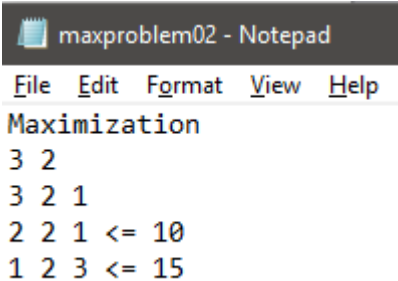
III.1. Phát biểu bài toán

III.1.1. Cấu trúc file input

Để xác định đầy đủ các yêu cầu bài toán QHTT, ta cần xác định các yếu tố:

- Yêu cầu của hàm mục tiêu: Cực đại hóa (Maximization) hay cực tiểu hóa (Minimization).
- Số ẩn chính và số ràng buộc chính của bài toán.
- Các hệ số của hàm mục tiêu.
- Các hệ số của các ràng buộc chính, dấu và giá trị vế phải (λ) của từng ràng buộc.

Từ các yếu tố trên, file text input sẽ có dạng như sau:



```
maxproblem02 - Notepad
File Edit Format View Help
Maximization
3 2
3 2 1
2 2 1 <= 10
1 2 3 <= 15
```

Hình 1. File text input

Cấu trúc của file text input được giải thích theo dòng như sau:

- Hàng đầu tiên nêu yêu cầu của hàm mục tiêu.
- Hàng thứ 2 khai báo **n ẩn chính, m ràng buộc chính**.
- Hàng thứ 3 là **n hệ số** của hàm mục tiêu.
- **m hàng còn lại**: Mỗi hàng khai báo các **n hệ số** của ràng buộc, **dấu** và **giá trị vế phải λ** .

III.1.2. Cấu trúc file output

File text output thể hiện các thông số trong bảng xoay sau mỗi lần thay đổi và kết quả cuối cùng (PATU) cùng với giá trị tối ưu hóa của hàm mục tiêu.

```

LPOutput - Notepad
File Edit Format View Help
Objective function
      3      2      1
Coefficient constraints matrix
      2      2      1      10
      1      2      3      15

Tableau #1
Pivot a[0,0] = 2
      x1      x2      x3      x4      x5      λ
-----
      [2]      2      1      1      0      10
      1      2      3      0      1      15
-----
      -3      -2      -1      0      0      0
Tableau #2
Final Tableau
      x1      x2      x3      x4      x5      λ
-----
      2      2      1      1      0      10
      0      1      2.5      -0.5      1      10
-----
      0      1      0.5      1.5      0      15
*** Final Solution ***
      x1 = 5      x2 = 0      x3 = 0      x4 = 0      x5 = 10      f(max) = 15

```

Hình 2. File text output

Cấu trúc của file text output được giải thích như sau:

- Thể hiện các hệ số của hàm mục tiêu, ma trận các hệ số ràng buộc chính của bài toán đầu vào.
- Các thông số của từng bảng xoay: Tọa độ và giá trị của vị trí cột mốc, các giá trị thay đổi trong bảng xoay (hệ số các ràng buộc, giá trị về phải λ , giá trị của hàm mục tiêu Z).
- Sau khi đã tìm được bảng đơn hình cuối cùng, in ra kết quả cuối cùng là **giá trị các biến** để tối ưu hóa hàm mục tiêu và **giá trị tối ưu hóa của hàm mục tiêu**.

III.2. Cấu trúc dữ liệu

Cấu trúc dữ liệu chủ yếu sử dụng trong thuật toán: Mảng 1 chiều, mảng 2 chiều.

Lý do sử dụng mảng:

- Mảng dễ dàng duyệt và thay đổi các hệ số ở ma trận ràng buộc, hơn nữa còn dễ dàng truy xuất tọa độ của vị trí cột mốc.
- Sử dụng bộ nhớ hiệu quả, tính cục bộ cao.
- Thuật toán QHTT không yêu cầu thay đổi kích thước ma trận hay xóa đi vài phần tử nên sử dụng cấu trúc mảng là hợp lý, đơn giản.

III.3. Thuật toán

III.3.1. Các hàm sử dụng trong thuật toán

Chúng ta sơ lược qua về các hàm chính có mặt trong chương trình để thuận tiện trong việc thông hiểu. Chức năng của từng hàm sẽ được trình bày trực tiếp với sơ đồ khối ở phần sau.

```
+ void ReadFile(string address)
+ void GenerateInformation()
+ void GenerateMaxProblem()
+ void GenerateMinProblem()
+ void InitiateTableau()
+ void ShowTableau()
+ void FindPivot()
+ void ChangeTableau()
+ bool isFinalTableau()
+ bool isSolutionExist()
+ void FindVars()
+ void FinalOutput()
```

III.3.2. Các biến sử dụng trong thuật toán

```
static List<string> lines = new List<string>();
static bool isMax;
static int numberOfVars, numberOfConstraints;
static int numberOfSlackVars = 0;
static double[] coEfficient;
static double[,] matrix;
static string[] sign;

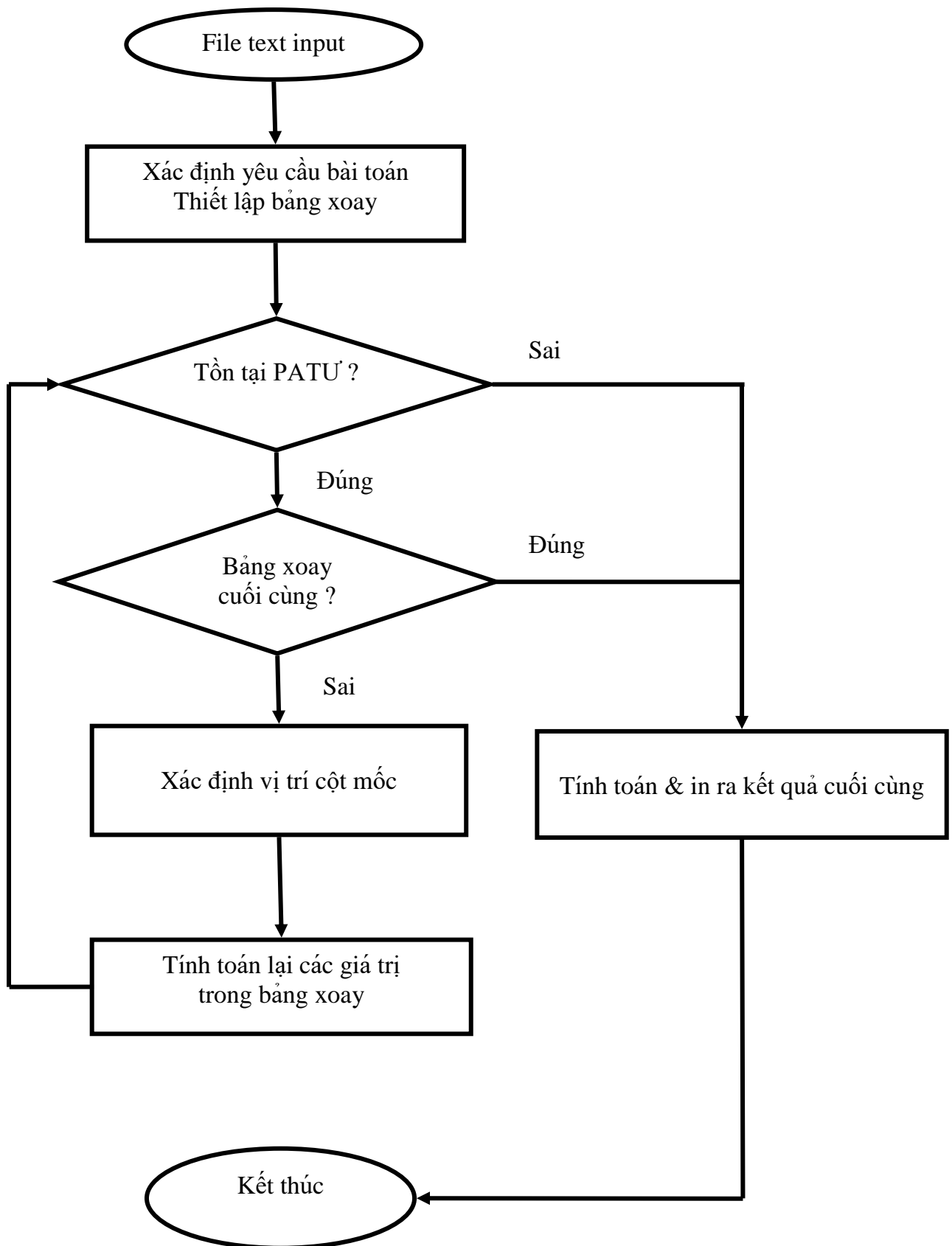
static double[,] extendedMatrix;
static double[] lambda;
static double[] indicators;
static double pivot;
static int pivotColumn = 0;
static int pivotRow = 0;
static double pivotOfIndicators, pivotOfLambda;
static double targetValue = 0;

static double[] vars;
```

Bảng 3. Danh sách các biến sử dụng

- *List<string> lines*: Lưu các hàng trong file text input.
- *bool isMax*: Kiểm tra yêu cầu của bài toán, giá trị true là Maximization còn giá trị false là Minimization.
- *int numberOfVars, int numberOfConstraints*: Số biến trong hàm mục tiêu, số ràng buộc chính trong chương trình.
- *int numberOfSlackVars*: Số biến phụ sẽ được tạo ra, khởi tạo bằng 0.
- Mảng *double[] coEfficient*: Lưu trữ các hệ số đầu vào ở hàm mục tiêu.
- Mảng 2 chiều *double[,] matrix*: Lưu trữ ma trận hệ số đầu vào ở các ràng buộc chính.
- Mảng *double[] sign*: Lưu trữ dấu của các ràng buộc.
- Mảng 2 chiều *double[,] extendedMatrix*: Lưu trữ ma trận hệ số ở các ràng buộc sau khi đã thêm các biến phụ (sau khi chuyển bài toán dạng tổng quát sang dạng chính tắc).
- Mảng *double[] lambda*: Lưu trữ các giá trị ở vế phải của các ràng buộc.
- Mảng *double[] indicators*: Lưu trữ các đối số của các hệ số hàm mục tiêu.
- *double pivot*: Giá trị của phần tử trục. Giá trị này thay đổi sau mỗi lần xoay bảng.
- *int pivotColumn, pivotRow*: Vị trí của hàng cột xoay và hàng xoay trên ma trận, giá trị này thay đổi sau mỗi lần xoay bảng.
- *double targetValue*: Giá trị của hàm mục tiêu z được khởi tạo bằng 0. Giá trị này được cộng thêm sau mỗi lần xoay bảng.
- Mảng *double[] vars*: Lưu giá trị của các biến ở phương án tối ưu để đạt được giá trị tối ưu.

III.3.3. Sơ đồ khối thuật toán



Hình 3. Sơ đồ khối thuật toán

Thực hiện phân tích thuật toán từng bước theo sơ đồ khối

1. Đọc file text input

Đảm nhận công việc đọc file input là hàm *void ReadFile(string address)*. Hàm này sử dụng *StreamReader* để đọc từng dòng trong file text input sau đó lưu các dòng này vào *List <string> lines*.

2. Xác định yêu cầu bài toán. Thiết lập bảng xoay

- Đầu tiên phân loại bài toán bằng hàm *void GenerateInfomation()*. Xác định yêu cầu bài toán là “Maximize” hoặc “Minimize” bằng cách kiểm tra hàng đầu tiên trong file text input có chứa chuỗi “max” hay không? Nếu có thực hiện hàm *void GenerateMaxProblem()*, nếu không thực hiện hàm *void GenerateMinProblem()*. Đồng thời trong quá trình này thực hiện đọc hàng thứ hai trong file text input và lưu vào số biên, số ràng buộc.
 - *void GenerateMaxProblem()*: Truyền các giá trị hệ số của hàm mục tiêu vào mảng *double[] coEfficient*, các giá trị về phải ràng buộc Lambda vào mảng *double[] lambda* đồng thời lưu và kiểm tra dấu trong mảng *double[] sign*. Nếu là dấu “ \leq ” hoặc “ \geq ” thì giá trị số biên phụ *int numberOfSlackVars* sẽ được cộng thêm 1. Nếu là dấu “ \leq ” hoặc “ $=$ ” lưu giá trị trong mảng *double[] sign* là 1, nếu là dấu “ \geq ” thì lưu giá trị là -1. Việc này nhằm đảm bảo mọi ràng buộc chính ở trong bài toán luôn quy về dấu “ \leq ” và “ $=$ ”, nếu là dấu “ \geq ” ta nhân bất phương trình với -1 để đổi dấu. Sau đó ta sẽ lưu ma trận hệ số các ràng buộc vào mảng 2 chiều *double[,] matrix*.
 - *void GenerateMinProblem()*: Hàm này đại khái thực hiện các công việc tương tự như hàm *void GenerateMaxProblem()* đồng thời chuyển bài toán “Minimization” thành bài toán đối ngẫu “Maximization” tương ứng. Điều này có nghĩa là, các giá trị về phải ràng buộc chính sẽ được lưu vào mảng *double[] coEfficient*, các giá trị hệ số của hàm mục tiêu sẽ được lưu vào mảng *double[] lambda* và dữ liệu của mảng 2 chiều *double[,] matrix* là ma trận chuyển vị của ma trận hệ số các ràng buộc chính trong bài toán “Minimization”.
- Sau khi thực hiện phân loại bài toán và xử lý dữ liệu đầu vào từ file text input. Ta thực hiện công việc thiết lập bảng đơn hình (bảng xoay) bằng hàm *void InitiateTableau()*. Mảng *double[] lambda* được giữ nguyên, ta sử dụng mảng *double[] indicators* để lưu hàng Δ của bảng xoay, tức là mảng này lưu giá trị các số đối của mảng *double[] coEfficient*. Ta thiết lập thêm ma trận hệ số các ràng buộc với các ẩn phụ, tức là sử dụng mảng 2 chiều *double[,] extendedMatrix* với số hàng bằng số các ràng buộc chính, số cột bằng số các ẩn chính + số các ẩn phụ *int numberOfSlackVars*. Ta sao chép các giá trị ở mảng *double[,] matrix* vào mảng *double[,] extendedMatrix* và thiết lập các giá trị ẩn phụ bằng 0 hoặc bằng 1 phù hợp với các ràng buộc chính.

Như vậy ở bước này, ta đã thực hiện các công việc:

- Chuyển dữ liệu từ file text input sang dữ liệu tin học của thuật toán.
- Thiết lập các dữ liệu chính cho bài toán “Maximization” để chuẩn bị áp dụng phương pháp đơn hình. Nếu là bài toán “Minimization” thì thực hiện phương pháp đối ngẫu chuyển về bài toán “Maximization”.
- Chuyển bài toán từ dạng tổng quát sang dạng chính tắc. Thiết lập bảng đơn hình (bảng xoay).

3. Kiểm tra sự tồn tại của PATU

Để thực hiện sự tồn tại của PATU, đầu tiên cần xác định giá trị cột mốc (sẽ được trình bày ở phần 5. của phần này). Sau đó thực hiện kiểm tra bài toán có tồn tại PATU hay không bằng hàm *bool isSolutionExist()*. Hàm này kiểm tra các giá trị tỷ lệ dòng trong bảng xoay, nếu tồn tại 1 giá trị tỷ lệ dòng bất kỳ dương thì trả về *true*, nếu không tồn tại giá trị dương nào thì trả về *false*.

4. Kiểm tra dấu hiệu tối ưu

Sử dụng phương pháp tương tự như việc kiểm tra sự tồn tại của PATU, ta sử dụng hàm *bool isFinalTableau()* để kiểm tra bảng xoay hiện tại có phải là bảng đơn hình cuối cùng hay không. Hàm này trả về giá trị *true* nếu không tồn tại giá trị Δ nào âm, ngược lại trả về giá trị *false* nếu tồn tại giá trị Δ bất kỳ âm.

Cơ sở lý thuyết của dấu hiệu bài toán không giải được và dấu hiệu tối ưu đã được trình bày ở phần II.2.3.c.2 và phần II.2.3.c.3.

5. Xác định vị trí cột mốc

Sử dụng hàm *void FindPivot()* quét và tìm giá trị Δ âm lớn nhất, tương tự quét và tìm giá trị ở cột λ có giá trị tỷ lệ dòng dương nhỏ nhất. Trong khi quét lưu lại chỉ số tọa độ hàng và cột của giá trị cột mốc vào các biến *int pivotColumn*, *pivotRow* giá trị cột mốc được lưu vào biến *double pivot*.

Cơ sở lý thuyết của giá trị cột mốc đã được trình bày ở phần II.2.3.c.4.

6. Tính toán các giá trị trong bảng xoay

Sử dụng hàm *void ChangeTableau()* để tính toán các giá trị thay đổi trong bảng xoay sau khi tìm được giá trị cột mốc.

- Thay đổi từng hàng trong bảng xoay theo công thức có ở quy tắc xoay bảng đã trình bày ở phần (trừ hàng có chứa giá trị cột mốc).
- Giá trị của hàm mục tiêu *double targetValue* sẽ được cộng thêm với giá trị cột λ ở tọa độ hàng của phần tử trục.

7. Tính toán kết quả cuối cùng

Khi đã tìm được bảng đơn hình cuối cùng, ta thực hiện giải hệ phương trình. Nếu ở bảng xoay cuối cùng, các biến có giá trị Δ khác 0 sẽ có giá trị bằng 0. Các biến có giá trị Δ bằng 0 sẽ được giải theo từng phương trình trong bảng đơn hình.

Giá trị tối ưu hóa của hàm mục tiêu chính là giá trị *double targetValue* ở bảng đơn hình cuối cùng.

III.3.4. Độ phức tạp thuật toán

Có thể hiểu thuật toán đơn hình là thuật toán xuất phát từ một nghiệm cơ sở chấp nhận được x_B của bài toán gốc (P) , ta thực hiện một loạt các phép biến đổi hệ cơ sở sao cho nghiệm cơ sở tương ứng của x_B trở thành nghiệm cơ sở chấp nhận được (và cũng là nghiệm tối ưu) của bài toán đối ngẫu (D) .

Về phương diện toán học, phương pháp đơn hình được chứng minh trong trường hợp xấu nhất có độ phức tạp lũy thừa (*exponential complexity*). Có thể hiểu rằng do phương pháp đơn hình chạy trên biên của đa diện lồi nên mất nhiều thời gian để đi đến nghiệm tối ưu (nếu nghiệm này nằm ở mặt bên kia của đa diện). Tuy có độ phức tạp lũy thừa, phương pháp đơn hình vẫn được cài đặt thành công và có ứng dụng rất rộng rãi trên nhiều bài toán QHTT lớn (hàng nghìn biến và ràng buộc). Lý do là trong thực hành, người ta nhận thấy thuật toán đơn hình thường đi đến nghiệm tối ưu chỉ trong $m \rightarrow 3m$ lần lặp.

Ở thuật toán đơn hình sử dụng trong đề tài này, về cơ bản có 2 thao tác chính lồng vào nhau:

- + Thao tác kiểm tra có phải là bảng đơn hình cuối cùng hay không, nếu không thì thực hiện các phép thay đổi bảng đơn hình. Ở trường hợp xấu nhất, tất cả các biến có giá trị Δ âm và sau mỗi lần xoay bảng chỉ có một biến có giá trị Δ chuyển thành giá trị dương. Khi đó, thao tác này thực hiện tối đa n lần, với n là số ẩn chính có trong hàm mục tiêu.

- + Thao tác tính toán các giá trị được thay đổi trong bảng xoay. Thao tác này luôn thực hiện với số lần lặp $(n+p)m$ lần với p là số ẩn phụ, m là số ràng buộc chính. Dễ dàng thấy rằng số ẩn phụ p tối đa bằng số ràng buộc chính m .

Như vậy thuật toán đơn hình có số bước tối đa là:

$$f(n, p, m) = nm(n + m)$$

Hay độ phức tạp của thuật toán là:

$$\theta(n^3)$$

Nhìn chung thuật toán đơn hình sử dụng trong đề tài này trung bình chạy trong thời gian đa thức.

IV. CHƯƠNG TRÌNH VÀ KẾT QUẢ

IV.1. Tổ chức chương trình

Ứng dụng được viết gồm 2 module:

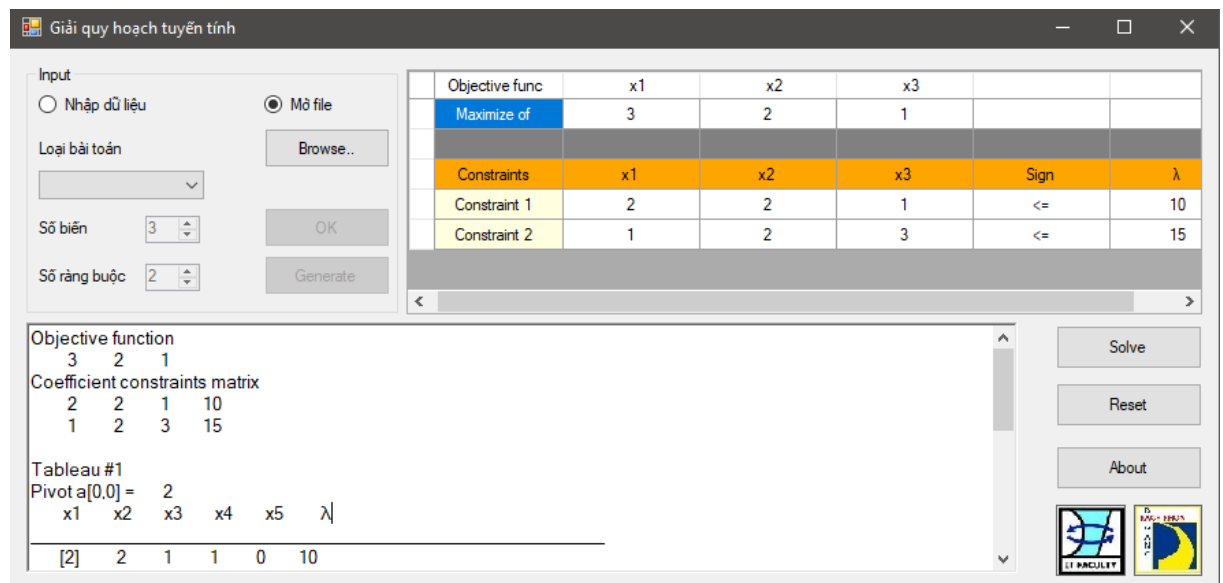
- Module xử lý giải thuật đơn hình để giải bài toán QHTT.
- Module giao diện thể hiện dữ liệu bài toán, kết quả trả về.

IV.2. Ngôn ngữ cài đặt

Ứng dụng giải bài toán QHTT được cài đặt bằng ngôn ngữ C#, lập trình Winform với .NET Framework 4.5.2. Ứng dụng hoàn toàn không sử dụng những thư viện toán học có sẵn liên quan đến việc tính toán QHTT.

IV.3. Kết quả

IV.3.1. Giao diện chính của chương trình

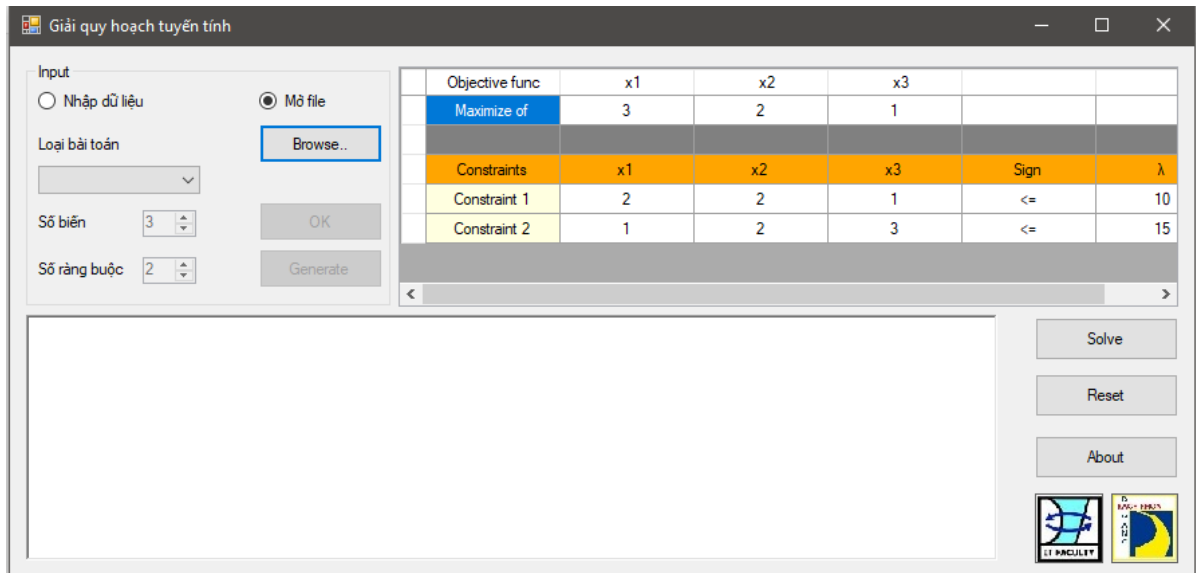


Hình 4. Giao diện chính của chương trình

Chương trình cung cấp giao diện cho phép tìm file text input có sẵn hoặc nhập dữ liệu tùy ý.

a. Sử dụng chức năng tìm kiếm file text input có sẵn

- Chọn RadioButton Mở file sau đó click vào Button Browse.
- Hộp thoại OpenFileDialog mở ra, tìm kiếm file text input và nhấn Open.
- Thông tin dữ liệu của bài toán sẽ được đổ ra DataGridView.

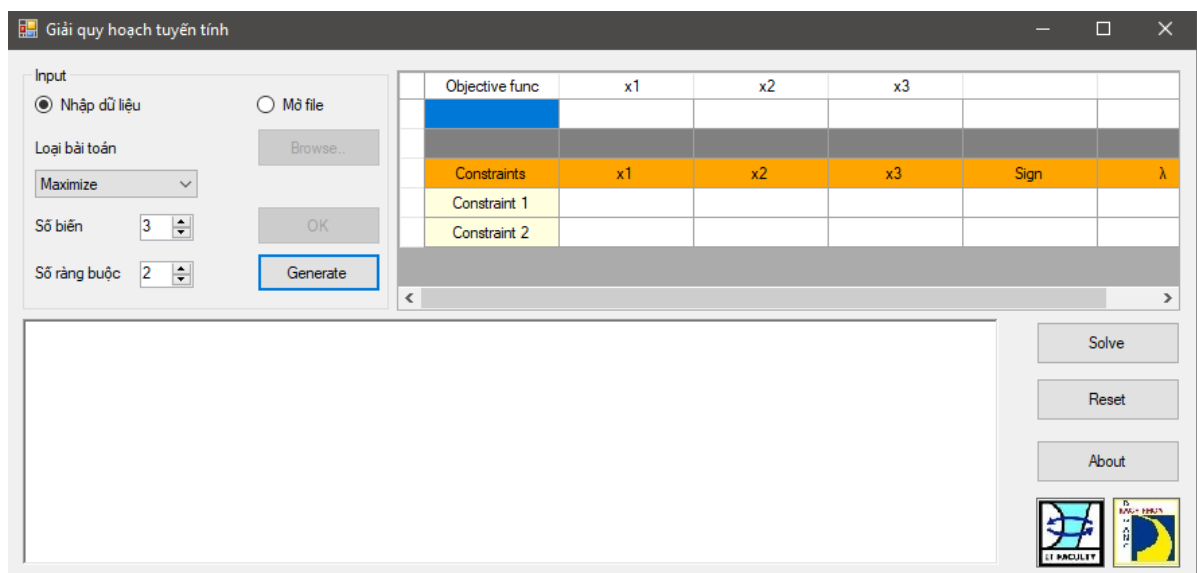


Hình 5. Thông tin bài toán được tự động điền vào DataGridView

- Click Button Solve để thực hiện giải bài toán, lời giải sẽ được in ra ở RichTextBox đồng thời file text output sẽ được lưu ở thư mục *C:\temp*.
- Nếu muốn sử dụng lại chương trình để giải bài toán QHTT khác, nhấn vào nút Reset.

b. Sử dụng chức năng tự nhập dữ liệu

- Chọn RadioButton Nhập dữ liệu.
- Chọn loại bài toán: Maximize hoặc Minimize. Nhập vào số biến, số ràng buộc.
- Click vào Button OK. DataGridView sẽ được tạo lập.



Hình 6. Tạo lập DataGridView để nhập dữ liệu

- Nhập vào dữ liệu bài toán: Các hệ số hàm mục tiêu, dữ liệu của các ràng buộc chính.
- Click vào Button Generate, lúc này bài toán đã sẵn sàng để giải, file text input sẽ được tạo ra và lưu ở thư mục $C:\temp$.
- Click Button Solve để giải bài toán và nhận lời giải.

IV.3.2. Kết quả thực thi của chương trình

Ta thực thi chương trình với bài toán mẫu sau:

$$5. \text{ Maximize: } z = 4x_1 - 3x_2 + 2x_3$$

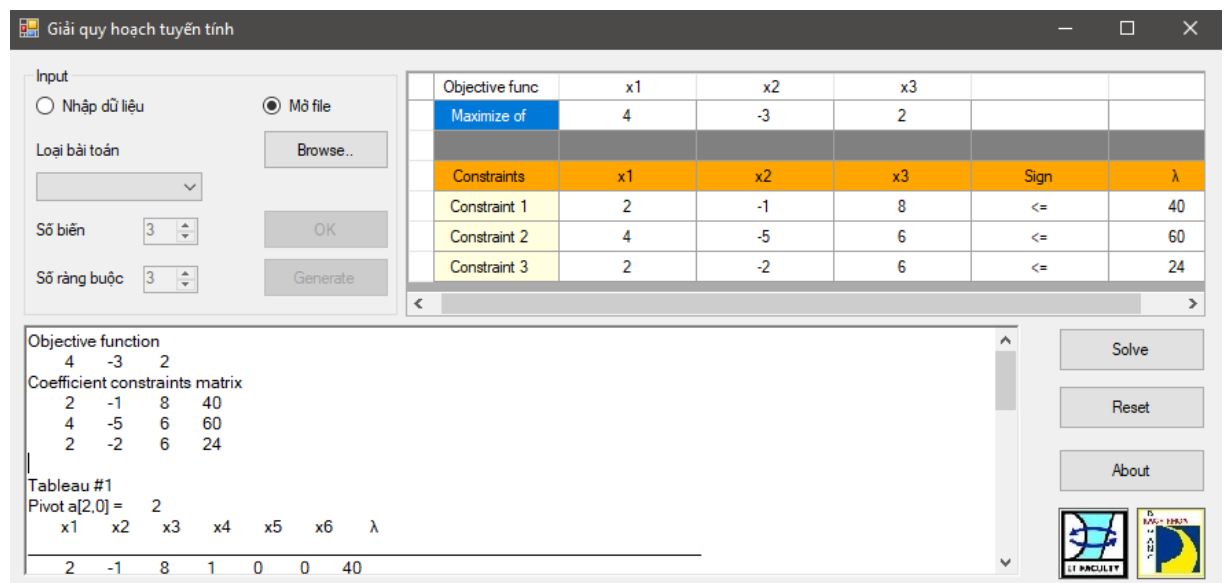
$$\text{Subject to: } \begin{cases} 2x_1 - x_2 + 8x_3 \leq 40 \\ 4x_1 - 5x_2 + 6x_3 \leq 60 \\ 2x_1 - 2x_2 + 6x_3 \leq 24 \end{cases}$$

- Nội dung file text input

```
Maximize
3 3
4 -3 2
2 -1 8 <= 40
4 -5 6 <= 60
2 -2 6 <= 24
```

Bảng 4. File text input bài toán mẫu

- Ảnh màn hình khi chạy chương trình



Hình 7. Chạy chương trình thực tế

- Kết quả file text output thu được

Objective function

4	-3	2				
---	----	---	--	--	--	--

Coefficient constraints matrix

2	-1	8	40
4	-5	6	60
2	-2	6	24

Tableau #1

Pivot a[2,0] = 2

x1	x2	x3	x4	x5	x6	λ
2	-1	8	1	0	0	40
4	-5	6	0	1	0	60
[2]	-2	6	0	0	1	24
-4	3	-2	0	0	0	0

Tableau #2

Pivot a[0,1] = 1

x1	x2	x3	x4	x5	x6	λ
0	[1]	2	1	0	-1	16
0	-1	-6	0	1	-2	12
2	-2	6	0	0	1	24
0	-1	10	0	0	2	48

Tableau #3

Final Tableau

x1	x2	x3	x4	x5	x6	λ
0	1	2	1	0	-1	16
0	0	-4	1	1	-3	28
2	0	10	2	0	-1	56
0	0	12	1	0	1	64

*** Final Solution ***

x1 = 28	x2 = 16	x3 = 0	x4 = 0	x5 = 28	x6 = 0	f(max) = 64
---------	---------	--------	--------	---------	--------	-------------

Bảng 5. File text output bài toán mẫu

IV.3.3. Nhận xét

Có thể thấy rằng ứng dụng load dữ liệu từ file text input vào DataGridView chính xác. Các bước giải được trình bày rõ ràng, tường minh trong file text output cũng như hiển thị trên RichTextbox của ứng dụng. Kết quả đã được đối chiếu với nhiều tài liệu học thuật nước ngoài về QHTT là chính xác tuyệt đối (các tài liệu sẽ được nêu ở phần tham khảo).

V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

V.1. Kết luận:

Quy hoạch tuyến tính là bài toán tối ưu hóa trải qua hàng chục năm đã phát triển với nhiều cách giải tối ưu, tuy nhiên phương pháp đơn hình được kiểm nghiệm và áp dụng được cho là rất hiệu quả, với thời gian tính toán khá ngắn. Đồ án lần này đã đạt được một số mục tiêu:

- Tìm hiểu và nắm được mô hình toán học của bài toán QHTT.
- Chỉ ra phương pháp đơn hình để giải bài toán QHTT, thông hiểu nội dung của thuật toán và chuyển đổi từ mô hình toán học sang mô hình tin học. Sử dụng ngôn ngữ lập trình C# để tạo lập ứng dụng giải bài toán QHTT. Ví dụ khi thực thi chương trình và các kết quả số phù hợp với các kết quả trên lý thuyết.
- Hiểu được vai trò to lớn của QHTT trong các lĩnh vực: vận tải, sản xuất, quốc phòng...

V.2. Hướng phát triển:

Thuật toán đơn hình được cài đặt trong chương trình nhìn chung đã giải quyết được bài toán QHTT ổn định trong thời gian đa thức. Tuy nhiên, vẫn có thể giải bài toán QHTT bằng nhiều phương pháp khác như:

- Áp dụng chung với thuật toán đơn hình đối ngẫu (dual simplex method). Nhìn chung, trong hai thuật toán đơn hình, luôn luôn có một thuật toán có xuất phát điểm dễ dàng hơn. Vì vậy, trong thực hành người ta thường sử dụng luân phiên hai thuật toán này.
- Giải bài toán QHTT bằng phương pháp điểm trong với thuật toán giãn affine (*affine scaling algorithm*). Ý tưởng chính của phương pháp này là xuất phát từ một nghiệm bên trong đa diện lồi, ta lần lượt tìm các nghiệm khác có giá trị hàm mục tiêu tốt hơn và đi dần đến nghiệm tối ưu. Các phương pháp điểm trong được chứng minh là có độ phức tạp thuật toán đa thức (*polynomial complexity*) trong mọi trường hợp.
- Phương pháp Ellipsoid và phương pháp Ellipsoid cải tiến tìm một điểm thỏa mãn hệ bất phương trình tuyến tính và ứng dụng phương pháp này vào bài toán QHTT. Ý tưởng của phương pháp này là xây dựng một dãy Ellipsoid có thể tích giảm dần chứa tập nghiệm P sao cho dãy các điểm tâm của Ellipsoid hội tụ về một điểm nào đó của P. Phương pháp này được trình bày bởi ThS. Phạm Quý Mười của Đại học Đà Nẵng trong “Tập chí khoa học và công nghệ số 9(94). 2015, Đại học Đà Nẵng”.

Nhìn chung, đến nay đã có rất nhiều phương pháp khác nhau được trình bày để giải bài toán QHTT. Tuy nhiên, với nhiều hạn chế về trình độ toán học hiện nay của em, việc thông hiểu và cài đặt các thuật toán này khá là phức tạp.

TÀI LIỆU THAM KHẢO

- *Các tài liệu đọc hiểu về QHTT*

1. Linear Programming – UCLA Department of Mathematics
2. Linear Programming: Foundations and Extensions – Robert J. Vanderbei
3. Extended Mathematics for Cambridge IGCSE – Audrey Simpson
4. Linear Programming: The Simplex Method – Del Mar College, Texas, U.S.A
5. Linear Programming: The Simplex Method Maximization & Minimization – CengageAsia
6. Simplex Method: Solving Standard Maximization Problems - <https://www.zweigmedia.com>
7. Chuyên đề quy hoạch tuyến tính: Phương pháp đơn hình, phương pháp đơn hình đối ngẫu, phương pháp điểm trong - <https://csstudyfun.wordpress.com/category/quy-ho%E1%BA%A1ch-tuy%E1%BA%BFn-tinh/>
8. Giáo trình toán kinh tế – ThS. Nguyễn Thị Hà, Đại học Vinh.
9. Tối ưu hóa – PGS.TS Nguyễn Hải Thanh, NXB Bách khoa- Hà Nội
10. Phương pháp Ellipsoid cải tiến và ứng dụng giải bài toán quy hoạch tuyến tính, Tạp chí khoa học và công nghệ ĐHQĐN số 9(94).2015- ThS. Phạm Quý Mười
11. Các trang web thảo luận <https://math.stackexchange.com/>, <https://stackoverflow.com/>, <https://www.researchgate.net/>
12. Simplex algorithm - https://en.wikipedia.org/wiki/Simplex_algorithm

- *Các bài toán được sử dụng để đối chiếu kết quả với ứng dụng được lấy từ:*

1. Linear Programming – UCLA Department of Mathematics
2. Linear Programming: The Simplex Method – Del Mar College, Texas, U.S.A
3. Linear Programming: The Simplex Method Maximization & Minimization – CengageAsia

PHỤ LỤC

Mã nguồn chương trình *LPApplication*
Class Form1.cs

```
/*
-----
Namespace:      <LPSolve>
Class:          <LPApplication>
Description:     <Solving Linear Programming using Simplex Method>
Author:         <Le Trong Hieu>           Date: <2017-11-01>
School:         <Da Nang University of Technology>
-----
*/

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace LPApplication
{
    public partial class Form1 : Form
    {
        static string pathInput = @"c:\temp\LPInput.txt";
        static string pathOutput = @"c:\temp\LPOutput.txt";
        static List<string> lines = new List<string>();
        static int countTable = 1;

        static bool isMax;
        static int numberOfVars, numberOfConstraints;
        static int numberOfSlackVars = 0;
        static double[] coEfficient;
        static double[,] matrix;
        static string[] sign;

        static double[,] extendedMatrix;
        static double[] lambda;
        static double[] indicators;
        static double pivot;
        static int pivotColumn = 0;
        static int pivotRow = 0;
        static double pivotOfIndicators, pivotOfLambda;
        static double targetValue = 0;

        static double[] vars;

        public static void ReadFile(string address)
        {
            try
            {
                StreamReader sr = new StreamReader(address);
                {

```

```

        string line;

        while ((line = sr.ReadLine()) != null)
        {
            lines.Add(line);
        }
    }

    catch (Exception e)
    {
        Console.WriteLine("Khong the doc du lieu tu file da cho: ");
        Console.WriteLine(e.Message);
    }
}

public static void ShowProblem()
{
    if (isMax == false) Console.WriteLine("The given minimization problem
corresponding to the dual maximization problem");
    Console.WriteLine("Objective function");
    for (int i = 0; i < coEfficient.Length; i++)
    {
        Console.Write(String.Format("{0,10:0.###}", coEfficient[i]));
    }

    Console.WriteLine("\nCoefficient constraints matrix");

    for (int i = 0; i < matrix.GetLength(0); i++)
    {
        for (int j = 0; j < matrix.GetLength(1); j++)
        {
            Console.Write(String.Format("{0,10:0.###}", matrix[i, j]));
        }
        Console.Write(String.Format("{0,10:0.###}", lambda[i]));

        Console.WriteLine();
    }
}

public static void GenerateInfomation()
{
    isMax = (lines[0].ToLower().Contains("max")) ? true : false;

    string[] temp = lines[1].Split(' ');
    numberOfVars = int.Parse(temp[0]);
    numberOfConstraints = int.Parse(temp[1]);

    if (isMax) GenerateMaxProblem();
    else GenerateMinProblem();
}

public static void GenerateMaxProblem()
{
    coEfficient = new double[numberOfVars];
    matrix = new double[numberOfConstraints, numberOfVars];
    lambda = new double[numberOfConstraints];
    sign = new string[numberOfConstraints];

    //Put values into an coefficient of objective function Array
    string[] tempLineObjectiveFunc = lines[2].Split(' ');
    for (int i = 0; i < numberOfVars; i++)

```

```

        {
            coEfficient[i] = double.Parse(tempLineObjectiveFunc[i]);
        }

        for (int i = 3; i < lines.Count; i++)
        {
            string[] tempLine = lines[i].Split(' ');

            //Change sign to <= or =; Count how many slack variables will be
generated
            int checkSign = (tempLine[numberOfVars] == ">=") ? -1 : 1;
            sign[i - 3] = (tempLine[numberOfVars] == "=") ? "=" : "<=";
            numberOfSlackVars = (tempLine[numberOfVars] == "=") ?
numberOfSlackVars : numberOfSlackVars + 1;

            //Put coefficient from Constraints into matrix
            for (int j = 0; j < numberOfVars; j++)
            {
                matrix[i - 3, j] = checkSign * double.Parse(tempLine[j]);
            }

            //Put values into lambda column
            lambda[i - 3] = checkSign * double.Parse(tempLine[numberOfVars +
1]);

        }
        InitiateTableau();
    }

    public static void GenerateMinProblem()
    {
        double[] minCoEfficient = new double[numberOfVars];
        double[,] minMatrix = new double[numberOfConstraints, numberOfVars];
        double[] minLambda = new double[numberOfConstraints];

        string[] tempLineObjectiveFunc = lines[2].Split(' ');
        for (int i = 0; i < numberOfVars; i++)
        {
            minCoEfficient[i] = double.Parse(tempLineObjectiveFunc[i]);
        }
        for (int i = 3; i < lines.Count; i++)
        {
            string[] tempLine = lines[i].Split(' ');

            //Count how many slack variables will be generated
            int checkSign = (tempLine[numberOfVars] == "<=") ? -1 : 1;
            numberOfSlackVars = numberOfVars;

            //Put coefficient from Constraints into minMatrix
            for (int j = 0; j < numberOfVars; j++)
            {
                minMatrix[i - 3, j] = checkSign * double.Parse(tempLine[j]);
            }

            //Put values into minLambda column
            minLambda[i - 3] = checkSign * double.Parse(tempLine[numberOfVars +
1]);

        }

        //Change to max problem
        coEfficient = new double[numberOfConstraints];
        matrix = new double[numberOfVars, numberOfConstraints];
    }

```



```

        lambda = new double[numberOfVars];
        sign = new string[numberOfVars];
        for (int i = 0; i < sign.Length; i++)
        {
            sign[i] = "<=";
        }

        minLambda.CopyTo(coEfficient, 0);
        minCoEfficient.CopyTo(lambda, 0);

        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                matrix[i, j] = minMatrix[j, i];
            }
        }

        int temp = numberOfVars;
        numberOfVars = numberOfConstraints;
        numberOfConstraints = temp;

        InitiateTableau();
    }

    public static void InitiateTableau()
    {
        extendedMatrix = new double[numberOfConstraints, numberOfVars +
numberOfSlackVars];
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                extendedMatrix[i, j] = matrix[i, j];
            }
        }

        int indexOfSlackVar = 0;
        for (int i = 0; i < extendedMatrix.GetLength(0); i++)
        {
            if (sign[i] == "<=")
            {
                extendedMatrix[i, matrix.GetLength(1) + indexOfSlackVar] = 1;
                indexOfSlackVar++;
            }
        }

        indicators = new double[numberOfVars + numberOfSlackVars];
        for (int i = 0; i < coEfficient.Length; i++)
        {
            indicators[i] = -coEfficient[i];
        }
    }

    public static void ShowTableau()
    {
        FindPivot();
        if (!isFinalTableau())
            Console.WriteLine("Pivot a[" + pivotRow + ", " + pivotColumn + "] = "
+ String.Format("{0,8:0.###}", pivot));
        else Console.WriteLine("Final Tableau");
    }

```

```

    int countName = 1;

    for (int i = 0; i < numberOfVars + numberOfSlackVars; i++)
    {
        string varName = ((isMax == true) ? "x" : "y") + countName;
        Console.WriteLine(String.Format("{0,10:[0.###]}", varName));
        countName++;
    }
    Console.WriteLine(String.Format("{0,10:[0.###]}", "λ"));
    Console.WriteLine();
    for (int i = 0; i < numberOfSlackVars + numberOfVars + 1; i++)
    {
        Console.WriteLine("_____");
    }
    Console.WriteLine();

    for (int i = 0; i < extendedMatrix.GetLength(0); i++)
    {
        for (int j = 0; j < extendedMatrix.GetLength(1); j++)
        {
            if (i == pivotRow && j == pivotColumn && !isFinalTableau())
            {
                Console.WriteLine(String.Format("{0,10:[0.###]}",
extendedMatrix[i, j]));
            }
            else
            {
                Console.WriteLine(String.Format("{0,10:0.###}",
extendedMatrix[i, j]));
            }
            Console.WriteLine(String.Format("{0,10:0.###}", lambda[i]));
            Console.WriteLine();
        }
        for (int i = 0; i < numberOfSlackVars + numberOfVars + 1; i++)
        {
            Console.WriteLine("_____");
        }
        Console.WriteLine();
        foreach (double indicator in indicators)
        {
            Console.WriteLine(String.Format("{0,10:0.###}", indicator));
        }

        Console.WriteLine(String.Format("{0,10:0.###}", targetValue));
    }

    public static void FindPivot()
    {
        pivotOfIndicators = pivotOfLambda = double.PositiveInfinity;

        for (int i = 0; i < indicators.Length; i++)
        {
            if (indicators[i] < 0 && indicators[i] < pivotOfIndicators)
            {
                pivotOfIndicators = indicators[i];
                pivotColumn = i;
            }
        }

        for (int i = 0; i < lambda.Length; i++)
        {
            double ratio = lambda[i] / extendedMatrix[i, pivotColumn];
            if (ratio > 0 && ratio < pivotOfLambda)

```

```

        {
            pivotOfLambda = ratio;
            pivotRow = i;
        }
    }

    pivot = extendedMatrix[pivotRow, pivotColumn];
}

public static void ChangeTableau()
{
    for (int i = 0; i < extendedMatrix.GetLength(0); i++)
    {
        if (i != pivotRow)
        {
            double delta = -(extendedMatrix[i, pivotColumn] / pivot);
            lambda[i] += delta * lambda[pivotRow];
            for (int j = 0; j < extendedMatrix.GetLength(1); j++)
            {
                extendedMatrix[i, j] += delta * extendedMatrix[pivotRow, j];
            }
        }
    }
    //Change indicator
    double deltaLambda = -indicators[pivotColumn] / pivot;
    for (int j = 0; j < indicators.Length; j++)
    {
        indicators[j] += deltaLambda * extendedMatrix[pivotRow, j];
    }
    targetValue += deltaLambda * lambda[pivotRow];
}

public static bool isFinalTableau()
{
    foreach (double indicator in indicators)
    {
        if (indicator < 0) return false;
    }
    return true;
}

public static bool isSolutionExist()
{
    for (int i = 0; i < lambda.Length; i++)
    {
        if (lambda[i] / extendedMatrix[i, pivotColumn] > 0) return true;
    }
    return false;
}

public static void FindVars()
{
    Console.WriteLine("\n*** Final Solution ***");
    if (isMax == true)
    {
        vars = new double[numberOfVars + numberOfSlackVars];
        bool[] checkValue = new bool[numberOfVars + numberOfSlackVars];
        for (int i = 0; i < indicators.Length; i++)
        {
            if (indicators[i] != 0)
            {

```

```

        vars[i] = 0;
        checkValue[i] = true;
    }
}

for (int i = 0; i < extendedMatrix.GetLength(0); i++)
{
    for (int j = 0; j < extendedMatrix.GetLength(1); j++)
    {
        if (checkValue[j] == false && extendedMatrix[i, j] != 0)
        {
            vars[j] = lambda[i] / extendedMatrix[i, j];
            checkValue[j] = true;
        }
    }
}
}
else
{
    vars = new double[numberOfSlackVars];
    for (int i = numberOfVars; i < indicators.Length; i++)
    {
        vars[i - numberOfVars] = indicators[i];
    }
}

}

public static void FinalOutput()
{
    if (double.IsPositiveInfinity(targetValue))
    {
        Console.WriteLine("Objective function is unbounded. No solution
exist !");
    }
    else
    {
        int countVarName = 1;
        foreach (double value in vars)
        {
            Console.Write(String.Format("{0,10:0.###}", "x" + countVarName +
" = " + value));
            countVarName++;
        }
        Console.WriteLine(String.Format("{0,18:0.###}", "f(" + ((isMax ==
true) ? "max) = " : "min) = ") + targetValue));
    }
}

public static void SolveProcedure()
{
    GenerateInfomation();
    ShowProblem();
    Console.WriteLine("\nTableau #" + countTable);
    ShowTableau();
    while (!isFinalTableau())
    {
        if (!isSolutionExist())
        {
            Console.WriteLine("No solution exists !");
            break;
        }
    }
}

```

```
        }
        countTable++;
        Console.WriteLine("\nTableau #" + countTable);
        FindPivot();
        ChangeTableau();
        ShowTableau();
    }
    if (countTable == 1) Console.WriteLine("No Solution exists !");
    else
    {
        FindVars();
        FinalOutput();
    }
}

public static void ConsoleToTextFile()
{
    FileStream fs = File.Create(pathOutput);
    TextWriter tmp = Console.Out;
    StreamWriter sw = new StreamWriter(fs);
    Console.SetOut(sw);
    SolveProcedure();
    Console.SetOut(tmp);
    sw.Close();
}

public Form1()
{
    InitializeComponent();
    radioButtonOpen.Checked = true;
}

private void radioButtonGetInput_CheckedChanged(object sender, EventArgs e)
{
    buttonOpen.Enabled = false;
    numericUpDownVars.Enabled = true;
    numericUpDownConstraints.Enabled = true;
    buttonOK.Enabled = true;
}

private void radioButtonOpen_CheckedChanged(object sender, EventArgs e)
{
    buttonOpen.Enabled = true;
    numericUpDownVars.Enabled = false;
    numericUpDownConstraints.Enabled = false;
    buttonOK.Enabled = false;
    buttonGenerate.Enabled = false;
}

private void buttonSolve_Click(object sender, EventArgs e)
{
    try
    {
        richTextBox1.Text = File.ReadAllText(pathOutput);
        dataGridView1.AutoGenerateColumns = false;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

```

private void buttonReset_Click(object sender, EventArgs e)
{
    Application.Restart();
    Environment.Exit(0);
}

private void buttonOK_Click(object sender, EventArgs e)
{
    try
    {
        buttonGenerate.Enabled = true;
        buttonOK.Enabled = false;
        int cols = int.Parse(numericUpDownVars.Value.ToString());
        int rows = int.Parse(numericUpDownConstraints.Value.ToString());
        dataGridView1.Columns.Add("x", "Objective func");
        for (int i = 0; i < cols + 2; i++)
        {
            if (i < cols)
            {
                dataGridView1.Columns.Add("x" + (i + 1), "x" + (i + 1));
            }
            else dataGridView1.Columns.Add("x" + (i + 1), " ");
        }

        for (int i = 0; i < rows + 3; i++)
        {
            if (i <= 2)
                dataGridView1.Rows.Add(" ");
            else dataGridView1.Rows.Add("Constraint " + (i - 2));
        }

        dataGridView1.Rows[2].Cells[0].Value = "Constraints";
        for (int i = 0; i < cols; i++)
        {
            dataGridView1.Rows[2].Cells[i + 1].Value = "x" + (i + 1);
        }
        dataGridView1.Columns[0].DefaultCellStyle.BackColor =
Color.LightYellow;
        dataGridView1.Columns[0].DefaultCellStyle.BackColor =
Color.LightYellow;
        dataGridView1.Columns[0].ReadOnly = true;
        dataGridView1.Rows[1].ReadOnly = true;
        dataGridView1.Rows[1].DefaultCellStyle.BackColor = Color.Gray;
        dataGridView1.Rows[2].Cells[cols + 1].Value = "Sign";
        dataGridView1.Rows[2].Cells[cols + 2].Value = "λ";
        dataGridView1.Rows[2].ReadOnly = true;
        dataGridView1.Rows[2].DefaultCellStyle.BackColor = Color.Orange;
        foreach (DataGridViewColumn dgvc in dataGridView1.Columns)
        {
            dgvc.SortMode = DataGridViewColumnSortMode.NotSortable;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void buttonGenerate_Click(object sender, EventArgs e)
{
    try
    {

```

```

        int cols = int.Parse(numericUpDownVars.Value.ToString());
        int rows = int.Parse(numericUpDownConstraints.Value.ToString());

        List<string> myList = new List<string>();
        int countLine = rows + 3;

        string line3 = "";
        for (int i = 0; i < cols; i++)
        {
            line3 += dataGridView1.Rows[0].Cells[i + 1].Value.ToString() + "
";
        }
        myList.Add(comboBoxType.SelectedItem.ToString());
        myList.Add(cols.ToString() + " " + rows.ToString());
        myList.Add(line3);

        for (int i = 0; i < rows; i++)
        {
            string myLine = "";
            for (int j = 0; j < cols + 2; j++)
            {
                myLine += dataGridView1.Rows[i + 3].Cells[j +
1].Value.ToString() + " ";
            }
            myList.Add(myLine);
        }

        FileStream fs = File.Create(pathInput);
        StreamWriter sw = new StreamWriter(fs);
        foreach (string s in myList)
        {
            sw.WriteLine(s);
        }
        sw.Close();
        MessageBox.Show(fs.Name);
        ReadFile(fs.Name);
        ConsoleToFile();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void buttonOpen_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog1 = new OpenFileDialog();

    openFileDialog1.InitialDirectory = "c:\\";
    openFileDialog1.Filter = "txt files (*.txt)|*.txt";
    openFileDialog1.FilterIndex = 2;
    openFileDialog1.RestoreDirectory = true;
    openFileDialog1.Multiselect = false;

    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        try
        {
            ReadFile(openFileDialog1.FileName);
            ConsoleToFile();
            int cols = int.Parse(lines[1].Split(' ')[0]);
            int rows = int.Parse(lines[1].Split(' ')[1]);

```

```

        numericUpDownVars.Value = cols;
        numericUpDownConstraints.Value = rows;
        dataGridView1.Columns.Add("x", "Objective func");

        //Configure Rows 0
        for (int i = 0; i < cols + 2; i++)
        {
            if (i < cols)
            {
                dataGridView1.Columns.Add("x" + (i + 1), "x" + (i +
1));
            }
            else dataGridView1.Columns.Add("x" + (i + 1), " ");
        }
        for (int i = 0; i < rows + 3; i++)
        {
            if (i <= 2)
                dataGridView1.Rows.Add(" ");
            else dataGridView1.Rows.Add("Constraint " + (i - 2));
        }

        //Configure Row 1
        string[] templine = lines[2].Split(' ');
        dataGridView1.Rows[0].Cells[0].Value =
(lines[0].ToString().Contains("max") ? "Maximize " : "Minimize ") + "of";
        for (int i = 0; i < templine.Length; i++)
        {
            dataGridView1.Rows[0].Cells[i + 1].Value = templine[i];
        }

        //Configure Rows 3
        dataGridView1.Rows[2].Cells[0].Value = "Constraints";
        for (int i = 0; i < cols; i++)
        {
            dataGridView1.Rows[2].Cells[i + 1].Value = "x" + (i + 1);
        }
        dataGridView1.Rows[2].Cells[cols + 1].Value = "Sign";
        dataGridView1.Rows[2].Cells[cols + 2].Value = "λ";

        for (int i = 0; i < rows; i++)
        {
            string[] tempCons = lines[i + 3].Split(' ');
            for (int j = 0; j < cols; j++)
            {
                dataGridView1.Rows[i + 3].Cells[j + 1].Value =
tempCons[j];
            }
            dataGridView1.Rows[i + 3].Cells[cols + 1].Value =
tempCons[cols];
            dataGridView1.Rows[i + 3].Cells[cols + 2].Value =
tempCons[cols + 1];
        }

        dataGridView1.Rows[1].ReadOnly = true;
        dataGridView1.Rows[1].DefaultCellStyle.BackColor = Color.Gray;
        dataGridView1.Rows[2].DefaultCellStyle.BackColor = Color.Orange;
        dataGridView1.Columns[0].DefaultCellStyle.BackColor =
Color.LightYellow;
        foreach (DataGridViewColumn dgvc in dataGridView1.Columns)
        {
            dgvc.SortMode = DataGridViewColumnSortMode.NotSortable;
        }
    
```



```
        }  
        catch (Exception ex)  
        {  
            MessageBox.Show(ex.Message);  
        }  
    }  
}
```