ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI:

QUY HOẠCH TUYẾN TÍNH VÀ MỘT SỐ ỨNG DỤNG

MÔN HỌC: ĐẠI SỐ TUYẾN TÍNH

LỚP L13 – NHÓM 13 – HỌC KỲ 251

Giảng viên hướng dẫn: ThS. Nguyễn Xuân Mỹ

STT	Họ và tên	MSSV	Điểm số	Chữ ký
1	Trương Quốc Học	2510606		
2	Trương Tuấn Khôi	2510708		
3	Võ Duy Vũ Hoàng	2510604		
4	Võ Thị Thảo	2413214		
5	Võ Thu Hằng	2510561		
6	Võ Thừa Hiếu	2510587		
7	Võ Trương Gia Huấn	2510608		
8	Võ Việt Hoàng	2510605		
9	Võ Việt Khuê	2510710		
10	Trương Thiện Hùng	2510615		
11	Vương Quốc Hùng	2510616		

TP. Hồ Chí Minh, tháng 11 năm 2025

Bảng phân công

STT	Họ và tên	MSSV	Phân công nhiệm vụ
1	Trương Quốc Học	2510606	Mục 1
2	Trương Tuấn Khôi	2510708	Mục 2.1
3	Võ Duy Vũ Hoàng	2510604	Mục 2.1
4	Võ Thị Thảo	2413214	Mục 2.2
5	Võ Thu Hằng	2510561	Mục 2.3 - 2.4 - 2.5
6	Võ Thừa Hiếu	2510587	Mục 3.1 - 3.2.1 - 3.4
7	Võ Trương Gia Huấn	2510608	Mục 3.2.1 - 3.2.2 - 3.3.1 - Tổng hợp
8	Võ Việt Hoàng	2510605	Mục 4.1
9	Võ Việt Khuê	2510710	Mục 3.5 - 4.2
10	Trương Thiện Hùng	2510615	Mục 3.2.3 - 3.3.2
11	Vương Quốc Hùng	2510616	Mục 5

LỜI NÓI ĐẦU

Chúng em – tập thể nhóm 13 xin được gửi lời cảm ơn sâu sắc và chân thành nhất đến Cô **Nguyễn Xuân Mỹ**, hiện đang công tác tại Trường Đại học Bách Khoa – Đại học Quốc gia Thành phố Hồ Chí Minh, người đã trực tiếp giảng dạy và hướng dẫn chúng em trong suốt quá trình học tập và thực hiện đề tài của môn học Đại số tuyến tính.

Chúng em ý thức được rằng, trong quá trình thực hiện đề tài, dù đã cố gắng hết sức nhưng không tránh khỏi những thiếu sót về nội dung, cách trình bày hay phương pháp triển khai. Vì vậy, chúng em rất mong nhận được những lời nhận xét, đánh giá và góp ý quý báu từ Cô để nhóm có thể rút kinh nghiệm và hoàn thiện hơn trong những dự án học thuật sau này. Mỗi ý kiến đóng góp của Cô đều là một bài học quý giá, giúp chúng em tiến bộ không ngừng trên con đường học tập và nghiên cứu.

Môn học Đại số tuyến tính là một trong những học phần đại cương quan trọng, đóng vai trò làm nền tảng cho các môn học chuyên ngành thuộc khối khoa học kỹ thuật – công nghệ. Đối với nhóm em nói riêng và sinh viên Trường Đại học Bách Khoa - Đại học Quốc gia Thành phố Hồ Chí Minh nói chung, việc nắm vững kiến thức của môn học này giúp chúng em trang bị tư duy khoa học, logic và khả năng phân tích – giải quyết vấn đề trong thực tiễn. Nhận thức được tầm quan trọng ấy, nhóm chúng em đã dành một lượng lớn thời gian và công sức để nghiên cứu, học tập và thực hành, từ đó xây dựng nền tảng vững chắc phục vụ cho việc học các môn chuyên ngành trong tương lai. Một lần nữa, chúng em xin chân thành cảm ơn Cô vì tất cả những gì Cô đã mang đến cho chúng em trong suốt quá trình học tập. Kính chúc Cô luôn mạnh khỏe, công tác tốt và tiếp tục truyền cảm hứng học tập, nghiên cứu cho nhiều thế hệ sinh viên Bách khoa trong những năm tới.

Chúng em xin chân thành cảm ơn!

Toàn thể sinh viên nhóm L13

TP. Hồ Chí Minh, ngày 30 tháng 11 năm 2025

Mục lục

1 Khái niệm và bài toán thực tế

1.1 Khái niệm

Quy hoạch tuyến tính - linear programming (LP) là một thuật toán nhằm tìm ra phương án tối ưu (hoặc kế hoạch tối ưu) từ vô số các phương án quyết định. Phương án tối ưu là phương án thoả mãn được các mục tiêu đề ra của một hãng, phụ thuộc vào các hạn chế và các ràng buộc. Thuật toán LP đề cập đến vấn đề phân bổ nguồn lực khan hiếm giữa các hoạt động cạnh tranh trong một phương án tối ưu để đạt được hiệu quả cao nhất, lãi gộp cao nhất hay doanh thu hoặc chi phí thấp nhất. Mô hình quy hoạch tuyến tính gồm 2 thành phần cơ bản:

- **Hàm mục tiêu:** Biểu thị giá trị mà ta muốn tối ưu hóa (tối đa hóa hoặc tối thiểu hóa) để đạt được mục tiêu cụ thể.
- **Điều kiện ràng buộc:** Các ràng buộc dưới dạng các hạn chế về sự sẵn có của nguồn lực hay thoả mãn các yêu cầu tối thiểu.

$\Rightarrow$ Do đó có thể hiểu đơn giản quy hoạch tuyến tính là lĩnh vực toán học nghiên cứu các bài toán tối ưu mà **hàm mục tiêu** (vấn đề được quan tâm) và các ràng buộc (điều kiện của bài toán) đều là hàm và các phương trình hoặc bất phương trình tuyến tính.

1.2 Các thành phần của quy hoạch tuyến tính

Mô tả mô hình quy hoạch tuyến tính gồm 3 thành phần cơ bản:

- **Biến quyết định (Decision Variables):** Là các ẩn số của bài toán, đại diện cho các đại lượng mà chúng ta muốn tối ưu hóa hoặc điều chỉnh. Chúng thường được ký hiệu là các biến (vd: $x_1, x_2, \ldots, x_n$)
- **Hàm mục tiêu (Objective Function):** Là một hàm tuyến tính của các biến quyết định, biểu thị chính xác và cụ thể mục tiêu cần được tối đa hóa (như lợi nhuận, doanh thu) hoặc tối thiểu hóa (như chi phí, thời gian). Hàm mục tiêu có dạng:

  $$Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

  Trong đó c_i là những hằng số (hệ số của hàm mục tiêu)
- **Các ràng buộc (Constraints):** Là các phương trình hoặc bất phương trình tuyến tính mà các biến quyết định phải thỏa mãn. Chúng mô tả các giới hạn về nguồn lực (nguyên liệu,

thời gian, vốn), yêu cầu kỹ thuật hoặc điều kiện thị trường và thường có dạng:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq, =, \geq b_i$$

1.3 Các đặc điểm của quy hoạch tuyến tính

1. **Tính tuyến tính (Linearity):**
   - Tất cả các hàm toán học trong mô hình (cả hàm mục tiêu và các ràng buộc) đều phải là các biểu thức tuyến tính.
   - Ví dụ: $3x_1 + 2x_2 \leq 100$, chứ không có x_1^2, $\ln(x_2)$, x_1x_2, ...
2. **Tính chắc chắn (Certainty):**
   - Các hệ số trong hàm mục tiêu (lợi nhuận/đơn vị, ...) và các ràng buộc (nguyên liệu/đơn vị, ...) đều được biết rõ, xác định trước và cố định.
   - Không có yếu tố ngẫu nhiên hay thay đổi theo xác suất (nếu có thì thuộc dạng quy hoạch ngẫu nhiên).
3. **Tính phụ gia (Additivity):**
   - Ảnh hưởng của từng biến đến hàm mục tiêu và ràng buộc là cộng dồn, độc lập. Nói cách khác, các biến quyết định không tương tác hay ảnh hưởng lẫn nhau.
   - Ví dụ: Tổng chi phí = chi phí từ x_1 + chi phí từ x_2 → không có tác động tương hỗ giữa chúng.
4. **Tính khả phân (Divisibility):**
   - Các biến quyết định có thể nhận giá trị thực bất kỳ (không nhất thiết nguyên) → tài nguyên có thể được chia liên tục.
   - Một số bài toán yêu cầu nghiệm nguyên (số người, số phương tiện, ...) → trở thành quy hoạch nguyên.
5. **Tính phi âm (Non-negativity):**
   - Hầu hết các bài toán quy hoạch tuyến tính đều có điều kiện ràng buộc $x \geq 0$ (biểu diễn các đại lượng như thời gian, nguyên liệu, sản phẩm, ...).

1.4 Ứng dụng của quy hoạch tuyến tính

Quy hoạch tuyến tính có được ứng dụng trong nhiều lĩnh vực như vận tải, sản xuất, tài chính, ...

Ví dụ như trong sản xuất

- **Hoạch định sản xuất (Production Planning):** Xác định số lượng từng loại sản phẩm cần sản xuất để **tối đa hóa lợi nhuận** hoặc tối ưu hóa chi phí, dưới các ràng buộc về nguyên vật liệu, lao động, thời gian máy ...
- **Bố trí nguồn lực (Resource Allocation):** Phân bổ nguyên liệu, máy móc, nhân công hợp lý để đạt được mục tiêu tốt nhất.
- **Lập lịch sản xuất (Scheduling):** Sắp xếp thứ tự công việc và bố trí lao động hợp lý để đồng thời đáp ứng nhu cầu sản xuất nhưng vẫn tối ưu chi phí nhân công.

1.5 Ưu điểm và nhược điểm của quy hoạch tuyến tính

Ưu điểm:

- Tính toán nhanh, hiệu quả cao.
- Dễ hiểu, dễ mô hình hóa.
- Ứng dụng rộng rãi.
- Độ chính xác cao.
- Có khả năng giải được các bài toán lớn với hàng trăm, thậm chí hàng triệu biến và ràng buộc.
- Đối với các bài toán tuyến tính, đảm bảo tìm được giải pháp tối ưu toàn cục (nếu tồn tại), không chỉ tối ưu cục bộ.

Nhược điểm:

- Giới hạn về dạng tuyến tính.
- Chỉ áp dụng được cho các bài toán tuyến tính.
- Nhạy cảm với dữ liệu không chính xác.
- Khó khăn khi giải quyết các bài toán lớn không còn tuyến tính rõ ràng.
- Nếu không tồn tại giải pháp tối ưu (vd: do ràng buộc không thể đạt được), quy hoạch tuyến tính có thể không cung cấp thông tin hữu ích hoặc chỉ ra rằng vấn đề không có giải pháp.

2 Cơ sở lý thuyết và thuật toán tối ưu

2.1 Sơ lược về lý thuyết

2.1.1 Cơ sở lý thuyết

Cơ sở lý thuyết của quy hoạch tuyến tính nằm ở việc giải các bài toán tối ưu hóa mà cả hàm mục tiêu (lợi ích hoặc chi phí) và các điều kiện ràng buộc đều là các hàm hoặc bất phương trình tuyến tính. Điều này có nghĩa là các mối quan hệ giữa các biến số là tuyến tính và mục tiêu là tìm một tập hợp các biến số (giải pháp) sao cho hàm mục tiêu đạt giá trị lớn nhất hoặc nhỏ nhất, đồng thời thỏa mãn tất cả các ràng buộc đã cho.

2.1.2 Các dạng quy hoạch tuyến tính

a. Dạng tổng quát:

1. $f(x) = c_1x_1 + c_2x_2 + \cdots + c_nx_n \to \max(\min)$
2. $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \,[\leq, \geq, =]\, b_i, \quad i = 1,2,\ldots,m$
3. $x_j \,[\geq 0, \leq 0 \text{ tùy ý}], \quad j = 1,2,\ldots,n$

Trong đó:

- (1) là **hàm mục tiêu**.
- (2) là **hệ ràng buộc chính**.
- (3) là **ràng buộc dấu**.
- (2) và (3) được gọi chung là **hệ ràng buộc của bài toán**.

Khi ấy:

- Mỗi vector $x = (x_1, x_2, \ldots, x_n)$ thỏa (2), (3) được gọi là một phương án của bài toán.
- Mỗi phương án x thỏa (1), nghĩa là tại đó hàm mục tiêu đạt giá trị nhỏ nhất (lớn nhất) trên tập các phương án được gọi là một phương án tối ưu (**PATU**) của bài.
- Giải một bài toán quy hoạch tuyến tính là đi tìm một **PATU** hoặc chứng minh bài vô nghiệm (nghĩa là không có **PATU**).

b. Dạng chính tắc:

1. $f(x) = c_1x_1 + c_2x_2 + \cdots + c_nx_n \to \max(\min)$
2. $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i, \quad i = 1,2,\ldots,m$
3. $x_j \geq 0, \quad j = 1,2,\ldots,n$

Nhận xét: Bài toán QHTT dạng chính tắc là bài toán QHTT dạng tổng quát mà:

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

c. Dạng chuẩn:

Bài toán QHTT dạng chuẩn là bài toán QHTT dạng chính tắc

1. $f(x) = c_1x_1 + c_2x_2 + \cdots + c_nx_n \to \max(\min)$
2. $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i, \quad i = 1,2,\ldots,m$
3. $x_j \geq 0, \quad j = 1,2,\ldots,n$

Trong đó:

- Các hệ số tự do đều không âm.
- Trong ma trận hệ số tự do có đủ m vector cột đơn vị: $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m$.

$$\mathbf{e}_1^T = \begin{pmatrix} 1 & 0 & \ldots & 0 \end{pmatrix}, \mathbf{e}_2^T = \begin{pmatrix} 0 & 1 & \ldots & 0 \end{pmatrix}, \ldots, \mathbf{e}_m^T = \begin{pmatrix} 0 & 0 & \ldots & 1 \end{pmatrix}$$

Khi đó:

- Các ẩn ứng với các cột vector đơn vị là ẩn cơ bản. Cụ thể ẩn ứng với vector cột đơn vị $\mathbf{e}_k$ là ẩn cơ bản thứ k.
- Một phương án mà các ẩn cơ bản đều bằng 0 được gọi là phương án cơ bản.
- Một phương án cơ bản có đủ m thành phần dương được gọi là không suy biến.
- Ngược lại, một phương án cơ bản có ít hơn m thành phần dương được gọi là suy biến.

Chú ý: Tổng quát, trong bài toán QHTT dạng chuẩn bất kì, khi cho ẩn cơ bản thứ k bằng hệ số tự do thứ k ($k = 1,2,\ldots,m$), còn các ẩn không cơ bản bằng 0, ta được một phương án cơ bản của bài toán. Ta gọi đây là phương án cơ bản ban đầu của bài toán.

2.1.3 Biến phụ và biến thừa

Biến phụ và biến thừa là hai khái niệm liên quan đến việc đưa các bài toán quy hoạch tuyến tính về **dạng chuẩn tắc**, giúp giải quyết dễ dàng hơn bằng các thuật toán như thuật toán đơn hình. Biến thừa được thêm vào để xử lý các bất đẳng thức dạng "lớn hơn hoặc bằng"($\geq$), còn biến phụ được thêm vào để xử lý các bất đẳng thức dạng "nhỏ hơn"($<$) hoặc "nhỏ hơn hoặc bằng"($\leq$).

- **Biến thừa (Surplus variable):**
  - **Mục đích:** Để chuyển các bất đẳng thức dạng $a_1x_1 + \cdots + a_nx_n \geq b$ thành phương trình $a_1x_1 + \cdots + a_nx_n - s = b$.
  - **Đặc điểm:** là một biến có giá trị *không âm* được *trừ đi* khỏi vế trái để cân bằng 2 vế.
  - **Ví dụ:** $2x_1 + 3x_2 \geq 10$ có thể viết thành $2x_1 + 3x_2 - s_1 = 10$, trong đó s_1 là biến thừa và $s_1 \geq 0$.
- **Biến phụ (Slack variable):**
  - **Mục đích:** Để chuyển các bất đẳng thức dạng $a_1x_1 + \cdots + a_nx_n \leq b$ thành phương trình $a_1x_1 + \cdots + a_nx_n + s = b$.
  - **Đặc điểm:** là một biến có giá trị *không âm* được *cộng vào* vế trái để cân bằng 2 vế.
  - **Ví dụ:** $x_1 + 3x_2 \leq 5$ có thể viết thành $x_1 + 3x_2 + s_1 = 5$, trong đó s_1 là biến phụ và $s_1 \geq 0$.

2.2 Miền nghiệm và Tính chất Hình học

Miền nghiệm của một bài toán Quy hoạch Tuyến tính (QHTT) là tập hợp tất cả các vector $X = (x_1, x_2, \ldots, x_n)$ thỏa mãn tất cả các ràng buộc. Miền nghiệm này luôn có một cấu trúc hình học đặc biệt và là nền tảng cho *phương pháp Simplex.*

2.2.1 Khái niệm Đa diện Lồi (Convex Polyhedron)

Miền nghiệm của một bài toán QHTT luôn là một tập hợp lồi (Convex Set). Cụ thể hơn, nó là một Đa diện Lồi (Convex Polyhedron).

- **Định nghĩa Tập hợp Lồi:** Một tập hợp S được gọi là lồi nếu với mọi hai điểm x1 và x2 thuộc S, đoạn thẳng nối hai điểm đó cũng nằm hoàn toàn trong S.
- **Định nghĩa Đa diện:** Một tập hợp được định nghĩa bởi hữu hạn số lượng các bất phương trình và phương trình tuyến tính.

- **Đa diện Lồi:** Miền nghiệm là giao của các nửa không gian (tạo bởi các bất phương trình) và các siêu phẳng (tạo bởi các phương trình), luôn tạo thành một Đa diện Lồi.

2.2.2 Khái niệm Đỉnh Cực Trị (Extreme Point)

Tính chất hình học quan trọng nhất của đa diện lồi là vai trò của các đỉnh (*Vertices*) trong việc tìm kiếm giá trị tối ưu của Hàm mục tiêu.

- **Định nghĩa Đỉnh Cực Trị (hay Điểm Cực Biên/Vertex):** Là một điểm trong đa diện lồi không thể biểu diễn dưới dạng tổ hợp lồi của hai điểm khác nhau thuộc đa diện lồi. Nói cách khác, các đỉnh là các góc của miền nghiệm đa diện lồi.
- **Mối liên hệ với Nghiệm Cơ sở:** Trong không gian n chiều, một đỉnh cực trị tương ứng với một **Nghiệm Cơ sở Khả thi** (*Basic Feasible Solution* - BFS) của hệ ràng buộc.

Định lý Cơ bản của QHTT:

Nếu bài toán QHTT có nghiệm tối ưu, thì ít nhất một nghiệm tối ưu phải xảy ra tại một đỉnh cực trị của miền nghiệm đa diện lồi.

$\Rightarrow$ Định lý này là cơ sở lý thuyết cho phương pháp Simplex, vì nó cho phép chúng ta tìm kiếm nghiệm tối ưu bằng cách chỉ duyệt qua các đỉnh thay vì phải tìm kiếm trong toàn bộ miền nghiệm.

2.2.3 Minh họa bằng Đồ thị 2D: Đa diện Lồi và Đỉnh Cực Trị

Để trực quan hóa khái niệm Đa diện Lồi và Đỉnh Cực Trị, ta xét một bài toán Quy hoạch Tuyến tính (QHTT) với chỉ hai biến quyết định (x1, x2). Khi đó, miền nghiệm có thể được biểu diễn trên mặt phẳng Oxy.

Bài toán ví dụ:

$$\text{Tối đa hóa } Z = 3x_1 + 2x_2$$

Ràng buộc:

1. $x_1 + 2x_2 \leq 6$ (Ràng buộc 1)
2. $x_1 \leq 4$ (Ràng buộc 2)
3. $x_2 \leq 2$ (Ràng buộc 3)
4. $x_1, x_2 \geq 0$ (Ràng buộc không âm)

Lời giải

1. Xác định Miền nghiệm Đa diện Lồi

Miền nghiệm được xác định bằng cách vẽ từng đường biên (biến bất phương trình thành phương trình) và xác định nửa mặt phẳng thỏa mãn điều kiện.

1. **Vẽ các đường biên:**
   - $L_1 : x_1 + 2x_2 = 6$
   - $L_2 : x_1 = 4$
   - $L_3 : x_2 = 2$
   - Các trục: $x_1 = 0$ và $x_2 = 0$.
2. **Miền nghiệm khả thi (Feasible Region):** Vùng thỏa mãn tất cả các bất phương trình cùng một lúc. Vùng này luôn tạo thành một hình đa giác lồi (nếu miền nghiệm bị chặn).

2. Xác định các Đỉnh Cực Trị (Vertices)

Các Đỉnh Cực Trị (tương ứng với các Nghiệm Cơ Sở Khả thi) là các giao điểm tạo nên các góc của đa diện lồi. Đây là các điểm mấu chốt mà thuật toán Simplex sẽ kiểm tra.

Đỉnh	**Tọa độ** (x_1, x_2)	**Ràng buộc giao nhau**
O	$(0,0)$	$x_1 = 0$ và $x_2 = 0$
A	$(4,0)$	$x_1 = 4$ và $x_2 = 0$
B	$(4,1)$	$x_1 = 4$ và $x_2 = 1$
C	$(2,2)$	$x_1 = 2$ và $x_2 = 2$
D	$(0,2)$	$x_1 = 0$ và $x_2 = 2$

3. Tìm Nghiệm Tối ưu

Theo Định lý Cơ bản của QHNT, ta chỉ cần đánh giá Hàm Mục tiêu Z tại các đỉnh cực trị:

Đỉnh	**Tọa độ** (x_1, x_2)	$Z = 3x_1 + 2x_2$
O	$(0,0)$	$Z = 3.(0) + 2.(0) = 0$
A	$(4,0)$	$Z = 3.(4) + 2.(0) = 12$
B	$(4,1)$	$Z = 3.(4) + 2.(1) = 14$
C	$(2,2)$	$Z = 3.(2) + 2.(2) = 10$
D	$(0,2)$	$Z = 3.(0) + 2.(2) = 4$

$\rightarrow$ Kết luận hình học :
Giá trị tối đa là $Z_{\max} = 14$ xảy ra tại đỉnh $B(4,1)$. Minh họa này khẳng định rằng nghiệm tối ưu luôn nằm tại một Đỉnh Cực Trị của đa diện lồi, điều này tạo cơ sở vững chắc cho thuật toán Simplex.

2.3 Giới thiệu Phương pháp Đơn hình (Simplex)

Phương pháp Simplex, được phát triển bởi George Dantzig vào năm 1947, là thuật toán nền tảng và hiệu quả nhất để giải các bài toán Quy hoạch Tuyến tính (QHTT). Nó dựa trên tính chất hình học rằng nghiệm tối ưu luôn nằm tại một đỉnh cực trị của miền nghiệm đa diện lồi.

2.3.1 Ý tưởng Cơ bản của Thuật toán Simplex:

Ý tưởng của Simplex là thực hiện một **quá trình lặp (iterative)** để tìm kiếm nghiệm tối ưu bằng cách đi từ đỉnh cực trị này sang đỉnh cực trị kề (liền kề) khác theo hướng làm cải thiện giá trị của hàm mục tiêu.

1. **Khởi tạo:** Bắt đầu từ một Nghiệm Cơ sở Khả thi (Basic Feasible Solution - BFS) ban đầu, thường là đỉnh $(0,0,\ldots,0)$.
2. **Cải thiện:** Tại mỗi bước lặp, thuật toán Simplex xác định xem có bất kỳ cạnh nào đi ra từ đỉnh hiện tại có thể dẫn đến một đỉnh kề mới với giá trị hàm mục tiêu tốt hơn (lớn hơn đối với $\max Z$, nhỏ hơn đối với $\min Z$) hay không.
3. **Di chuyển:** Nếu có, thuật toán di chuyển dọc theo cạnh có tiềm năng cải thiện tốt nhất, thực hiện một phép biến đổi hàng cơ bản (phép xoay trục - pivoting) để chuyển từ BFS hiện tại sang BFS kề mới.

4. **Kiểm tra:** Lặp lại quá trình cho đến khi tất cả các hướng di chuyển đều không làm cải thiện (hoặc làm giảm) giá trị hàm mục tiêu. Khi đó, đỉnh hiện tại chính là nghiệm tối ưu toàn cục.

2.3.2 Quy trình Tổng quát của Thuật toán Simplex:

Thuật toán Simplex được thực hiện thông qua việc biến đổi một ma trận mở rộng gọi là *Bảng Đơn hình (Simplex Tableau).*

1. Dạng Chuẩn (Standard Form)

Trước khi giải, bài toán QHTT phải được đưa về **Dạng Chuẩn**:

- Hàm mục tiêu: Ở dạng Tối đa hóa ($\max Z$).
- Ràng buộc: Tất cả các ràng buộc là phương trình (bằng cách thêm biến bù s_i, hoặc trừ biến dư e_i).
- Điều kiện không âm: Tất cả các biến đều không âm ($x_i, s_i, e_i \geq 0$).

2. Khởi Tạo Bảng Đơn hình

- Thiết lập Bảng Đơn hình ban đầu. Các biến bù/nhân tạo sẽ tạo thành cơ sở (Basic Variables) ban đầu, cung cấp một nghiệm cơ sở khả thi.

3. Điều kiện Tối ưu (Dừng)

- Kiểm tra dòng chỉ số (Index Row) hay dòng Hàm Mục tiêu (Z):
  - **Tối đa hóa** ($\max Z$)**:** Nếu tất cả các hệ số của biến quyết định và biến bù/dư trong dòng Z đều không âm (≥ 0), thì nghiệm hiện tại là tối ưu. Thuật toán dừng.

4. Chọn Biến vào Cơ sở (Entering Variable)

- Chọn cột có hệ số **âm lớn nhất** (giá trị tuyệt đối) trong dòng Z làm Cột Trụ (Pivot Column). Biến tương ứng với cột này là biến sẽ vào cơ sở.

5. Chọn Biến ra khỏi Cơ sở (Leaving Variable)

- Thực hiện Quy tắc Tỷ số Nhỏ nhất (Minimum Ratio Test) bằng cách chia giá trị RHS cho phần tử tương ứng của Cột Trụ.
- Tỷ số = $\dfrac{\text{RHS}}{\text{Phần tử Cột Trụ (với giá trị dương)}}$
- Chọn hàng có tỷ số dương nhỏ nhất làm Hàng Trụ (Pivot Row). Biến cơ sở tương ứng với hàng này là biến sẽ ra khỏi cơ sở.

6. Biến đổi Bảng Đơn hình (Pivoting)

- Thực hiện các phép biến đổi hàng để biến Phần tử Trụ (giao của Hàng Trụ và Cột Trụ) thành 1, và tất cả các phần tử khác trong Cột Trụ (bao gồm dòng Z) thành 0.
- Quay lại Bước 3 và lặp lại.

Ta hãy cùng xem xét ví dụ sau đây:

$$\text{Tối đa hóa } Z = 3x_1 + 5x_2$$

Ràng buộc:

1. $x_1 + 2x_2 \leq 4$
2. $3x_1 + 4x_2 \leq 12$
3. $x_1, x_2 \geq 0$

<u>Bước 1:</u> Dạng Chuẩn : Thêm biến bù s_1 và s_2

$$\begin{cases} x_1 + 2x_2 + s_1 & = 4 \\ 3x_1 + 4x_2 + s_2 & = 12 \\ Z - 3x_1 - 5x_2 & = 0 \end{cases}$$

Bước 2: Bảng Đơn hình Ban đầu (Pivot 0)

Cơ sở	Z	x_1	x_2	s_1	s_2	RHS	Tỷ số
s_1	0	1	2	1	0	4	$4/2 = 2$
s_2	0	3	4	0	1	12	$12/4 = 3$
Z	1	-3	-5	0	0	0	

Bước 3: Lặp 1 – Tìm và Xoay Trục

- **Chọn Cột Trụ:** Giá trị âm lớn nhất là -5 (Cột x_2), x_2 là biến vào cơ sở.
- **Chọn Hàng Trụ:** $\min(4/2, 12/4) = \min(2,3) = 2$. Hàng s_1 là Hàng Trụ. s_1 là biến ra khỏi cơ sở.
- **Phần tử Trụ:** 2.

Phép Biến đổi:

1. $R_1 \leftarrow R_1/2$ (Chuẩn hóa Hàng Trụ): Biến phần tử trụ thành 1.
2. $R_2 \leftarrow R_2 - 4.R_1^{\text{mới}}$ (Để x_2 trong R_2 thành 0).
3. $R_3 \leftarrow R_3 + 5.R_1^{\text{mới}}$ (Để x_2 trong R_3 thành 0).

Bảng Đơn hình (Pivot 1):

Cơ sở	Z	x_1	x_2	s_1	s_2	RHS	Tỷ số
x_2	0	$1/2$	1	$1/2$	0	2	$2/(1/2) = 4$
s_2	0	1	0	-2	1	4	$4/1 = 4$
Z	1	$-1/2$	0	$5/2$	0	10	

Bước 4: Lặp 2 – Tìm và Xoay Trục

- **Kiểm tra Tối ưu:** Dòng Z còn giá trị âm: $-1/2$ (Cột x_1).
- **Chọn Cột Trụ:** Cột x_1, x_1 là biến vào cơ sở.

- **Chọn Hàng Trụ:** $\min(4/(1/2), 4/1) = \min(4,4) = 4$. Có thể chọn tùy ý. Chọn Hàng s_2 là Hàng Trụ, s_2 là biến ra khỏi cơ sở.
- **Phần tử Trụ:** 1.

Phép Biến đổi:

1. $R_2 \leftarrow R_2/1$ (Hàng Trụ đã là 1).
2. $R_1 \leftarrow R_1 - (1/2)R_2^{\text{mới}}$ (Để x_1 trong R_1 thành 0).
3. $R_3 \leftarrow R_3 + (1/2)R_2^{\text{mới}}$ (Để x_1 trong R_3 thành 0).

Bảng Đơn hình (Pivot 2):

Cơ sở	**Z**	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{s_1}$	$\mathbf{s_2}$	**RHS**
x_2	0	0	1	$3/2$	$-1/2$	0
x_1	0	1	0	-2	1	4
Z	1	0	0	$3/2$	$1/2$	12

Bước 5: Điều kiện Dừng và Kết luận

Kiểm tra Tối ưu: Dòng Z không còn giá trị âm. Thuật toán dừng.
Nghiệm Tối ưu:

- Biến Cơ sở: x_1 và x_2.
- $x_1 = 4$
- $x_2 = 0$
- $Z_{\text{max}} = 12$

Kết luận: Nghiệm tối ưu là $x_1 = 4, x_2 = 0$, cho giá trị hàm mục tiêu tối đa $Z = 12$.

2.4 Cấu trúc bảng đơn hình

Bảng đơn hình là công cụ trung tâm của phương pháp Simplex, dùng để thực hiện các phép biến đổi đại số.

- **Cột ẩn cơ bản:** danh sách biến cơ bản.
- **Cột hệ số:** hệ số của các biến cơ bản trong hàm mục tiêu.
- **Cột phương án:** giá trị vế phải của các ràng buộc (còn gọi là cột nghiệm).
- **Các cột biến quyết định & biến phụ:** biểu diễn hệ số của các biến trong từng ràng buộc.
- **Hàng $\Delta_j = Z_j - C_j$:** thể hiện mức độ cải thiện của từng biến nếu được đưa vào cơ sở.

$$Z_j = \sum C_b.a_{ij}$$

Bảng đơn hình ban đầu:

Hệ số	Ẩn cơ bản	Phương án	x1	x2	x3
c_1	x_1	b_1	a_{11}	a_{12}	a_{13}
c_2	x_2	b_2	a_{21}	a_{22}	a_{23}

Các tình huống đặc biệt có thể xảy ra:

- **Không giới hạn (không bị chặn):** Nếu trong bước chọn biến ra, không có hệ số a_{ij} dương nào trong cột biến vào (tức tất cả $a_{ij} \leq 0$) thì ta không thể xác định được tỉ số $\lambda = \frac{b_i}{a_{ij}}$ hữu hạn. Điều đó có nghĩa giá trị hàm mục tiêu có thể tăng lên vô hạn mà vẫn thỏa mãn ràng buộc.
- **Nhiều nghiệm tối ưu:** Nếu ở bảng đơn hình tối ưu có một biến không cơ sở nào đó có $\Delta_j = 0$ thì có thêm nghiệm tối ưu khác (tức nghiệm cùng tối ưu). Đó là khi biến không cơ sở đó có thể vào cơ sở mà không làm thay đổi giá trị hàm mục tiêu.
- **Vô nghiệm (không có nghiệm khả thi):** Nếu dùng phương pháp khởi tạo (như Big-M) tạo các biến giả, sau khi loại bỏ hết và đi đến bảng tối ưu mà có biến giả vẫn lớn hơn 0 thì bài toán vô nghiệm khả thi (không tồn tại nghiệm thỏa mãn).

Xử lý ràng buộc và phương pháp Big-M

- Với ràng buộc $\leq$: thêm biến bù (*slack variable*), $s_i \geq 0$.
- Với ràng buộc $\geq$: trừ đi biến thừa (*surplus variable*), $s_i \geq 0$.

- Với ràng buộc $=$, hoặc ràng buộc $\geq$: thêm biến giả để đảm bảo có nghiệm cơ bản ban đầu.

Trong **hàm mục tiêu**, các biến giả (a_i) được thêm vào với hệ số tương ứng rất lớn M (Big-M).

- Với **max**: $-Ma_i$
- Với **min**: $+Ma_i$

$\rightarrow$ **Định lý:** Nếu phương án tối ưu của bài toán mở rộng có **ít nhất một biến giả khác 0** thì bài toán gốc **không có phương án tối ưu** (tức vô nghiệm khả thi). Như vậy, thuật toán Simplex mở rộng (Big-M) đảm bảo bài toán luôn có thể bắt đầu từ một nghiệm cơ bản khả thi.

Ta hãy cùng xem xét ví dụ sau đây:
Một công ty nước giải khát muốn sản xuất sản phẩm mới mang tên Oranj với dung tích tiêu chuẩn là 10 ounce cho mỗi chai. Sản phẩm được pha chế từ hai thành phần chính:

- x_1: số ounce nước ngọt cam (orange soda) trong một chai Oranj.
- x_2: số ounce nước cam nguyên chất (orange juice) trong một chai Oranj.

Trong quá trình pha chế, công ty cần tuân thủ các điều kiện sau:

1. **Giới hạn về đường:** Lượng đường tối đa trong mỗi chai không vượt quá 4 ounce. Biết rằng mỗi ounce soda cung cấp 0.5 ounce đường, còn mỗi ounce nước cam cung cấp 0.25 ounce đường.
2. **Hàm lượng vitamin C:** Mỗi chai phải chứa ít nhất 20 đơn vị vitamin C. Trong đó, mỗi ounce soda cung cấp 1 đơn vị và mỗi ounce nước cam cung cấp 3 đơn vị vitamin C.
3. **Dung tích tiêu chuẩn:** Tổng lượng soda và nước cam trong mỗi chai đúng bằng 10 ounce.

Mục tiêu của công ty là tối thiểu hóa chi phí pha chế, với giả định chi phí cho mỗi ounce soda là 2 đơn vị và cho mỗi ounce nước cam là 3 đơn vị.

Mô hình toán học

$$\min z = 2x_1 + 3x_2$$

$$\begin{cases} 0.5x_1 + 0.25x_2 \leq 4 & (\textit{ràng buộc về đường}) \\ x_1 + 3x_2 \geq 20 & (\textit{ràng buộc về vitamin C}) \\ x_1 + x_2 = 10 & (\textit{ràng buộc về dung tích}) \\ x_1, x_2 \geq 0 \end{cases}$$

Thêm biến bù, biến thừa để đưa về dạng phương trình: s_1, e_2

$$\begin{cases} 0.5x_1 + 0.25x_2 + s_1 = 4 \\ x_1 + 3x_2 - e_2 + a_2 = 20 \\ x_1 + x_2 + a_3 = 10 \\ x_1, x_2, s_1, e_2, a_2, a_3 \geq 0 \end{cases}$$

Vì chưa đủ cột đơn vị, ta cần thêm 2 biến giả: a_2, a_3

$$f(x) = 2x_1 + 3x_2 + Ma_2 + Ma_3$$

$$\begin{cases} 0.5x_1 + 0.25x_2 + s_1 = 4 \\ x_1 + 3x_2 - e_2 + a_2 = 20 \\ x_1 + x_2 + a_3 = 10 \\ x_1, x_2, s_1, e_2, a_2, a_3 \geq 0 \end{cases}$$

Tạo bảng đơn hình

Biến vào: x_2 (Δ lớn nhất).Biến ra: a_2 (λ nhỏ nhất).

Hệ số	Ẩn cơ bản	Phương án	x_1	x_2	s_1	e_2	a_2	a_3	$\lambda(x_2)$
0	s_1	4	$\frac{1}{2}$	$\frac{1}{4}$	1	0	0	0	16
M	a_2	20	1	3	0	-1	1	0	$\frac{20}{3}$
M	a_3	10	1	1	0	0	0	1	10
Δ_j			$2M-2$	$4M-3$	0	$-M$	0	0	

Biến vào: x_1 (Δ lớn nhất).Biến ra: a_3 (λ nhỏ nhất).

Hệ số	Ẩn cơ bản	Phương án	x_1	x_2	s_1	e_2	a_2	a_3	$\lambda(x_1)$
0	s_1	$\frac{7}{3}$	$\frac{5}{12}$	0	1	$\frac{1}{12}$	$-\frac{1}{12}$	0	5.6
3	x_2	$\frac{20}{3}$	$\frac{1}{3}$	1	0	$-\frac{1}{3}$	$\frac{1}{3}$	0	20
M	a_3	$\frac{10}{3}$	$\frac{2}{3}$	0	0	$\frac{1}{3}$	$-\frac{1}{3}$	1	5
Δj			$\frac{2M-3}{3}$	0	0	$\frac{M-3}{3}$	$\frac{3-4M}{3}$	0	

Hệ số	Ẩn cơ bản	Phương án	x_1	x_2	s_1	e_2	a_2	a_3	λ
0	s_1	1/4	0	0	1	$-1/8$	$1/8$	$-5/8$	
3	x_2	5	0	1	0	$-1/2$	$1/2$	$-1/2$	
2	x_1	5	1	0	0	$1/2$	$-1/2$	$3/2$	
Δ_j			0	0	0	$-1/2$	$(1-2M)/2$	$(3-2M)/2$	

Phương án tối ưu của bài toán mở rộng:

$$x^2 = (x_1; x_2; s_1; e_2; a_1; a_2; a_3) = (5; 5; 1/4; 0; 0; 0)$$

Giá trị tối ưu của bài toán mở rộng:

$$f(x^2) = 25$$

Vì phương án tối ưu của bài toán mở rộng có 2 biến giả $a_2 = a_3 = 0$ nên bài toán gốc có phương án tối ưu là:

$$x^2 = (x_1; x_2; s_1; e_2) = (5; 5; 1/4; 0)$$

$\rightarrow$ Giá trị tối ưu của bài toán:

$$f(x^2) = 25$$

2.5 Ưu điểm và hạn chế của phương pháp Simplex

Ưu điểm

- **Hiệu quả trong thực tiễn:** Phương pháp Simplex thường giải quyết được hầu hết các bài toán lập trình tuyến tính thực tế với số bước lặp tương đối nhỏ so với kích thước bài toán, giúp tiết kiệm thời gian tính toán.
- **Phạm vi ứng dụng rộng:** Simplex có thể xử lý các bài toán với nhiều biến và ràng buộc, đồng thời áp dụng trong nhiều lĩnh vực như kinh tế, kỹ thuật, quản lý kinh doanh và nghiên cứu vận hành.
- **Quy trình rõ ràng, có cấu trúc:** Thuật toán cung cấp các bước thực hiện cụ thể, tuần tự, giúp người dùng dễ triển khai, giám sát và giải thích kết quả.

Hạn chế

- **Vòng lặp và suy biến:** Trong một số trường hợp đặc biệt, thuật toán có thể lặp lại cùng một nghiệm hoặc "dừng"do hiện tượng suy biến. Tuy nhiên, các quy tắc chống vòng lặp có thể giảm thiểu vấn đề này.
- **Bất ổn về số học:** Các cài đặt chuẩn của Simplex đôi khi gặp lỗi làm tròn, đặc biệt với những bài toán nhạy cảm với sai số. Một số phiên bản cải tiến đã được phát triển để tăng tính ổn định.
- **Không đảm bảo thời gian đa thức cho mọi trường hợp:** Mặc dù Simplex hoạt động rất nhanh trong thực tế, nó không có chứng minh chạy trong thời gian đa thức cho tất cả các đầu vào, khác với một số thuật toán tối ưu khác.
- **Chỉ áp dụng cho bài toán tuyến tính:** Simplex không thể giải trực tiếp các bài toán phi tuyến tính và trong các bài toán phi tuyến không lồi, thuật toán có thể chỉ tìm được cực trị cục bộ thay vì cực trị toàn cục.

3 Ứng dụng vào bài toán vận tải - Phần mềm MATLAB

3.1 Sơ lược về mô hình bài toán vận tải

Bài toán đặt ra: Tìm phương án vận chuyển hàng từ m kho hàng đến n nơi nhận sao cho **tổng chi phí là bé nhất(tối ưu).**

Ta có thể **tổng quát hóa** bài toán như sau:

- Gọi a_i là số hàng hóa chứa ở kho thứ i với $i = \overline{1,m}$
- Gọi b_j là số hàng hóa cần nhận ở nơi nhận thứ j với $j = \overline{1,n}$
- Gọi chi phí vận chuyển 1 đơn vị hàng hóa từ kho thứ i sang nơi nhận thứ j là c_{ij} với $i = \overline{1,m}$, $j = \overline{1,n}$
- Gọi lượng hàng vận chuyển từ kho hàng thứ i sang nơi nhận thứ j là x_{ij} với $i = \overline{1,m}$, $j = \overline{1,n}$

Với chú ý rằng: Trong các bài toán này người ta thường phát biểu **chi phí vận chuyển** (c_{ij}) và **lượng hàng vận chuyển** (x_{ij}) bằng ngôn ngữ toán học **ma trận(Matrix)**

$$\text{C} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ c_{m1} & c_{m2} & \dots & c_{mn} \end{bmatrix} \quad \text{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}$$

Ta sẽ giả sử z là **tổng chi phí** (Ta cần tìm phương án để z **min**). Khi đó, mô hình **quy hoạch tuyến tính** của bài toán này là:

$$\boxed{\text{Hàm mục tiêu:} \quad z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}} \quad (1)$$

Với các điều kiện ràng buộc sau:

$$\sum_{j=1}^{n} x_{ij} \leq a_i \quad (i = 1, 2, \dots, m) \quad \text{(Ràng buộc cung)} \quad (2)$$

$$\sum_{i=1}^{m} x_{ij} \geq b_j \quad (j = 1, 2, \dots, n) \quad \text{(Ràng buộc cầu)} \quad (3)$$

$$x_{ij} \geq 0 \quad (i = 1, 2, \dots, m;\ j = 1, 2, \dots, n)$$

Chú ý rằng với (2) và (3) ta thường chỉ xét xảy ra dấu bằng.

Bài toán sẽ được gọi là **cân bằng** nếu như tổng cung bằng tổng cầu và khi ấy bài toán sẽ có nghiệm tối ưu

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$$

3.2 Cơ sở lý thuyết giải quyết bài toán

Để giải quyết bài toán vận tải trong đồ án này, quy trình thực hiện bao gồm 2 giai đoạn chính:

1. **Giai đoạn 1:** Tìm phương án cơ bản khả thi ban đầu.
2. **Giai đoạn 2:** Kiểm tra tính tối ưu và cải thiện phương án để tìm ra chi phí vận chuyển nhỏ nhất.

3.2.1 Lựa chọn phương pháp giải quyết bài toán

Để giải quyết bài toán vận tải, quy trình tổng quát bao gồm hai giai đoạn chính với các thuật toán tiêu biểu sau:

1. **Giai đoạn 1: Tìm phương án cơ bản khả thi ban đầu** Các phương pháp thường được sử dụng bao gồm:
    - Phương pháp Góc Tây Bắc (North-West Corner Method - NWCM).
    - Phương pháp Cước phí nhỏ nhất (Least Cost Method - LCM).
    - Phương pháp Xấp xỉ Vogel (Vogel's Approximation Method – VAM).
2. **Giai đoạn 2: Kiểm tra và tối ưu hóa phương án** Để tìm ra nghiệm tối ưu (chi phí thấp nhất), ta có thể dùng:
    - Phương pháp Bước đệm (Stepping Stone).
    - Phương pháp Phân phối cải tiến (Modified Distribution Method - MODI).

Kết luận lựa chọn: Trong phạm vi bài báo cáo này, nhóm em lựa chọn kết hợp NWCM cho giai đoạn 1 và MODI cho giai đoạn 2.

Lý do lựa chọn:

- **Về phương pháp Góc Tây Bắc (NWCM):** Đây là giải thuật đơn giản nhất để tìm nghiệm ban đầu. Ưu điểm lớn nhất là thuật toán dễ cài đặt (coding), tốn ít tài nguyên tính toán và độ dài mã nguồn ngắn, phù hợp với giới hạn của bài báo cáo.
- **Về phương pháp MODI:** Sau khi có nghiệm từ NWCM (thường chưa tối ưu), MODI cung cấp một quy trình đại số chặt chẽ để tìm ra nghiệm tối ưu. Việc kết hợp này giúp cân bằng giữa sự đơn giản khi khởi tạo và tính chính xác tuyệt đối khi kết thúc chương trình.

3.2.2 Phương pháp Góc Tây Bắc (North-West Corner Method)

Đây là phương pháp xác định phương án cơ bản ban đầu dựa trên vị trí địa lý của các ô trong bảng vận tải. Quy trình thực hiện:

- **Bước 1:** Bắt đầu phân bổ từ ô ở góc trên cùng bên trái (góc Tây Bắc) của bảng, tức là ô $(1,1)$.
- **Bước 2:** Phân bổ lượng hàng tối đa có thể cho ô chọn: $x_{ij} = \min(a_i, b_j)$.

- **Bước 3:** Điều chỉnh lượng cung/cầu:
  - Nếu nguồn cung a_i hết, loại bỏ dòng i và chuyển xuống ô ngay phía dưới.
  - Nếu nhu cầu b_j hết, loại bỏ cột j và chuyển sang ô ngay bên phải.
- **Bước 4:** Lặp lại quá trình cho đến khi toàn bộ lượng hàng cung và cầu được phân bổ hết.

3.2.3 Phương pháp MODI (Modified Distribution Method)

Sau khi có bảng phân phối ban đầu từ phương pháp Góc Tây Bắc, ta sử dụng MODI để kiểm tra và tối ưu hóa. Quy trình chi tiết gồm 7 bước như sau:

- **Bước 1: Xác định phương án khả thi ban đầu** - Sử dụng kết quả từ phương pháp Góc Tây Bắc. Đảm bảo bảng phân phối thỏa mãn:
  - Tổng hàng = Tổng cột (Cân bằng cung cầu).
  - Tất cả các ô phân phối đều ≥ 0.
  - Số ô chọn (ô cơ sở) phải bằng $m+n-1$ (với m là số hàng, n là số cột).
- **Bước 2: Tính hệ số thế vị** u_i, v_j Áp dụng công thức cho các ô đã phân phối (ô cơ sở):

  $$c_{ij} = u_i + v_j$$

  Trong đó:
  - c_{ij}: Chi phí đơn vị tại ô (i, j).
  - u_i, v_j: Các biến thế vị hàng và cột (chọn tùy ý một biến bằng 0, thường là $u_1 = 0$).
- **Bước 3: Tính hệ số cải thiện (dấu hiệu tối ưu)** - Áp dụng công thức cho các ô chưa phân phối (ô trống):

  $$\Delta_{ij} = c_{ij} - (u_i + v_j)$$

  Kiểm tra điều kiện:
  - Nếu tất cả $\Delta_{ij} \geq 0 \rightarrow$ Bảng hiện tại đã là Tối ưu. Dừng thuật toán.
  - Nếu tồn tại $\Delta_{ij} < 0$, chọn ô có Δ âm nhất để đưa vào cơ sở (gọi là ô điều chỉnh).
- **Bước 4: Tạo vòng khép kín** - Từ ô điều chỉnh đã chọn ở Bước 3, vẽ một đường khép kín đi qua các ô đã có phân phối (các đỉnh của vòng phải là các ô cơ sở). Đường đi chỉ được rẽ nhánh vuông góc (theo hàng hoặc cột).

- **Bước 5: Đặt dấu cho các đỉnh của vòng** - Đánh dấu cộng $(+)$ và trừ $(-)$ luân phiên tại các góc của vòng, bắt đầu từ:

$$\text{Ô khởi đầu (ô điều chỉnh)} \rightarrow \text{dấu } (+)$$

- **Bước 6: Tìm lượng điều chỉnh θ và cập nhật**
  - Tìm $\theta = \min(\text{lượng hàng tại các ô có dấu } -)$.
  - Cập nhật lại lượng hàng cho các ô trên vòng:
    - Ô có dấu $(+) \rightarrow$ cộng thêm θ.
    - Ô có dấu $(-) \rightarrow$ trừ đi θ.
- **Bước 7: Lặp lại** - Cập nhật lại bảng vận tải mới với lượng hàng đã thay đổi. Quay lại Bước 2 và tiếp tục quy trình cho đến khi tất cả $\Delta_{ij} \geq 0$.

3.3 Giải bài toán vận tải

3.3.1 Tìm phương án cơ bản ban đầu bằng phương pháp Tây Bắc

Bảng 1: Bảng chi phí và lượng Cung/Cầu

T / P	X1	X2	X3	X4	Phát
P1	19	30	50	10	7
P2	70	30	40	60	9
P3	40	8	70	20	18
Thu	5	8	7	14	34

Bước 1: Phân phối tại ô góc Tây Bắc (P_1, X_1)

- Tại ô (P_1, X_1): So sánh nguồn phát $P_1 = 7$ và nhu cầu $X_1 = 5$.
- Phân bổ: $x_{11} = \min(7,5) = \mathbf{5}$.
- Cập nhật:
  - Nhu cầu X_1 đã đủ $(0) \rightarrow$ **Xóa cột** X_1.
  - Nguồn phát P_1 còn dư: $7 - 5 = 2$.

P \T	X1	X2	X3	X4	Phát
P1	**5** (19)	30	50	10	7-**2**
P2	70	30	40	60	9
P3	40	8	70	20	18
Thu	5-**0**	8	7	14	**34**

Bước 2: Phân phối tại ô góc Tây Bắc mới (P_1, X_2)

- Tại ô (P_1, X_2): Nguồn P_1 còn 2, nhu cầu X_2 là 8.
- Phân bổ: $x_{12} = \min(2,8) = \mathbf{2}$.
- Cập nhật:
  - Nguồn P_1 đã hết $(0) \rightarrow$ **Xóa dòng** P_1.
  - Nhu cầu X_2 còn thiếu: $8 - 2 = 6$.

P \T	X1	X2	X3	X4	Phát
P1	**5** (19)	**2** (30)	50	10	2-**0**
P2	70	30	40	60	9
P3	40	8	70	20	18
Thu	0	8-**6**	7	14	**34**

Bước 3: Phân phối tại ô góc Tây Bắc mới (P_2, X_2)

- Tại ô (P_2, X_2): Nguồn P_2 là 9, nhu cầu X_2 còn 6.
- Phân bổ: $x_{22} = \min(9,6) = \mathbf{6}$.
- Cập nhật:
  - Nhu cầu X_2 đã đủ $(0) \rightarrow$ **Xóa cột** X_2.
  - Nguồn P_2 còn dư: $9 - 6 = 3$.

P \T	X1	X2	X3	X4	Phát
P1	**5** (19)	**2** (30)	50	10	0
P2	70	**6** (30)	40	60	9-**3**
P3	40	8	70	20	18
Thu	0	6-**0**	7	14	**34**

Bước 4: Phân phối tại ô góc Tây Bắc mới (P_2, X_3)

- Tại ô (P_2, X_3): Nguồn P_2 còn 3, nhu cầu X_3 là 7.
- Phân bổ: $x_{23} = \min(3, 7) = \mathbf{3}$.
- Cập nhật:
  - Nguồn P_2 đã hết (0) $\rightarrow$ **Xóa dòng** P_2.
  - Nhu cầu X_3 còn thiếu: $7 - 3 = 4$.

P \T	X1	X2	X3	X4	Phát
P1	**5** (19)	**2** (30)	50	10	0
P2	70	**6** (30)	**3** (40)	60	3-**0**
P3	40	8	70	20	18
Thu	0	0	7-**4**	14	**34**

Bước 5: Phân phối tại ô góc Tây Bắc mới (P_3, X_3)

- Tại ô (P_3, X_3): Nguồn P_3 là 18, nhu cầu X_3 còn 4.
- Phân bổ: $x_{33} = \min(18, 4) = \mathbf{4}$.
- Cập nhật:
  - Nhu cầu X_3 đã đủ (0) $\rightarrow$ **Xóa cột** X_3.
  - Nguồn P_3 còn dư: $18 - 4 = 14$.

P \T	X1	X2	X3	X4	Phát
P1	**5** (19)	**2** (30)	50	10	0
P2	70	**6** (30)	**3** (40)	60	0
P3	40	8	**4** (70)	20	18-**14**
Thu	0	0	4-**0**	14	**34**

Bước 6: Phân phối cuối cùng tại ô (P_3, X_4)

- Tại ô (P_3, X_4): Nguồn P_3 còn 14, nhu cầu X_4 là 14.
- Phân bổ: $x_{34} = \min(14, 14) = \mathbf{14}$.
- Cả nguồn và cầu đều được thỏa mãn. Quá trình phân phối kết thúc.

P \T	X1	X2	X3	X4	Phát
P1	**5** (19)	**2** (30)	50	10	**0**
P2	70	**6** (30)	**3** (40)	60	**0**
P3	40	8	**4** (70)	**14** (20)	**0**
Thu	**0**	**0**	**0**	**0**	**34**

Kết luận

Phương án vận chuyển ban đầu theo phương pháp góc Tây Bắc là:

$$X = \begin{pmatrix} 5 & 2 & 0 & 0 \\ 0 & 6 & 3 & 0 \\ 0 & 0 & 4 & 14 \end{pmatrix}$$

Tổng chi phí vận chuyển:

$$Z = (5 \times 19) + (2 \times 30) + (6 \times 30) + (3 \times 40) + (4 \times 70) + (14 \times 20)$$

$$Z = 95 + 60 + 180 + 120 + 280 + 280 = \mathbf{1015} \text{ (đơn vị tiền tệ)}$$

3.3.2 Tối ưu hóa bằng phương pháp MODI

Từ phương án cơ bản ban đầu (Góc Tây Bắc), ta có tổng chi phí là **1015**. Ta tiến hành kiểm tra tối ưu và cải thiện phương án.

Kết quả từ phương pháp Tây Bắc, ta có:

Các ô cơ sở (có phân phối) là: $(1,1),(1,2),(2,2),(2,3),(3,3),(3,4)$.
Số ô cơ sở = $m+n-1 = 3+4-1 = 6$. (Điều kiện không suy biến thỏa mãn).

Lần lặp 1: Tính thế vị và kiểm tra tối ưu

Tính hệ số thế vị u_i, v_j (Chọn $u_1 = 0$)

- $x_{11} > 0 \Rightarrow u_1 + v_1 = 19 \Rightarrow 0 + v_1 = 19 \Rightarrow \mathbf{v_1 = 19}$.
- $x_{12} > 0 \Rightarrow u_1 + v_2 = 30 \Rightarrow 0 + v_2 = 30 \Rightarrow \mathbf{v_2 = 30}$.
- $x_{22} > 0 \Rightarrow u_2 + v_2 = 30 \Rightarrow u_2 + 30 = 30 \Rightarrow \mathbf{u_2 = 0}$.
- $x_{23} > 0 \Rightarrow u_2 + v_3 = 40 \Rightarrow 0 + v_3 = 40 \Rightarrow \mathbf{v_3 = 40}$.
- $x_{33} > 0 \Rightarrow u_3 + v_3 = 70 \Rightarrow u_3 + 40 = 70 \Rightarrow \mathbf{u_3 = 30}$.
- $x_{34} > 0 \Rightarrow u_3 + v_4 = 20 \Rightarrow 30 + v_4 = 20 \Rightarrow \mathbf{v_4 = -10}$.

Tính hệ số cải thiện $\Delta_{ij} = c_{ij} - (u_i + v_j)$ **cho ô trống**

ui \vj	19	30	40	-10	
0	**5** (19)	**2** (30)	$\Delta_{13} = 10$	$\Delta_{14} = 20$	**P1**
0	$\Delta_{21} = 51$	**6** (30)	**3** (40)	$\Delta_{24} = 70$	**P2**
30	$\Delta_{31} = -9$	$\Delta_{32} = 8 - 60 = -52$	**4** (70)	**14** (20)	**P3**

Có $\Delta_{32} = -52$ là giá trị âm nhất $\rightarrow$ **Chọn ô (3,2) đưa vào cơ sở**.

Tạo vòng và điều chỉnh

- Vòng: $(3,2) \rightarrow (3,3) \rightarrow (2,3) \rightarrow (2,2) \rightarrow (3,2)$.
- Dấu: $(3,2)[+] \rightarrow (3,3)[-] \rightarrow (2,3)[+] \rightarrow (2,2)[-]$.
- Lượng điều chỉnh: $\theta = \min(x_{33}, x_{22}) = \min(4,6) = 4$.
- Cập nhật:
  - $x_{32} = 0 + 4 = 4$.
  - $x_{33} = 4 - 4 = 0$ (Loại khỏi cơ sở).
  - $x_{23} = 3 + 4 = 7$.
  - $x_{22} = 6 - 4 = 2$.

Lần lặp 2: Kiểm tra bảng mới

Tương tự như trên, ta có Δ_{ij}

ui \vj	19	30	40	42	
0	**5** (19)	**2** (30)	10	$\Delta_{14} = 10 - 42 = -32$	**P1**
0	51	**2** (30)	**7** (40)	18	**P2**
-22	43	**4** (8)	52	**14** (20)	**P3**

Có $\Delta_{14} = -32$ là âm nhất $\rightarrow$ Chọn ô (1,4) đưa vào cơ sở.

Tạo vòng và điều chỉnh

- Vòng: $(1,4) \rightarrow (3,4) \rightarrow (3,2) \rightarrow (1,2) \rightarrow (1,4)$. (Chú ý: không qua (2,2) vì cần tạo chu trình đóng với các ô cơ sở).
- Dấu: $(1,4)[+] \rightarrow (3,4)[-] \rightarrow (3,2)[+] \rightarrow (1,2)[-]$.
- Lượng điều chỉnh: $\theta = \min(x_{34}, x_{12}) = \min(14,2) = 2$.

- Cập nhật:
  - $x_{14} = 0 + 2 = 2$.
  - $x_{34} = 14 - 2 = 12$.
  - $x_{32} = 4 + 2 = 6$.
  - $x_{12} = 2 - 2 = 0$ (Loại khỏi cơ sở).

Lần lặp 3 (Kết quả tối ưu)

Tương tự, ta có Δ_{ij}

ui \vj	**19**	**-2**	**8**	**10**	
0	**5**	$\Delta = 32$	$\Delta = 42$	**2**	**P1**
32	$\Delta = 19$	**2**	**7**	$\Delta = 18$	**P2**
10	$\Delta = 11$	**6**	$\Delta = 52$	**12**	**P3**

Lưu ý: Các giá trị ô trống $\Delta_{ij} \geq 0$, phương án đã đạt tối ưu.

Kết luận cuối cùng

Phương án vận chuyển tối ưu là:

$$X_{opt} = \begin{pmatrix} 5 & 0 & 0 & 2 \\ 0 & 2 & 7 & 0 \\ 0 & 6 & 0 & 12 \end{pmatrix}$$

Tổng chi phí tối thiểu:

$$Z_{min} = (5 \times 19) + (2 \times 10) + (2 \times 30) + (7 \times 40) + (6 \times 8) + (12 \times 20)$$

$$Z_{min} = 95 + 20 + 60 + 280 + 48 + 240 = \mathbf{743} \text{ (đơn vị tiền tệ)}$$

So với phương án Góc Tây Bắc ($Z = 1015$), phương pháp MODI đã tiết kiệm được: $1015 - 743 = 272$ đơn vị.

3.4 Kiểm tra kết quả bằng MATLAB

3.4.1 Triển khai giải thuật

```
clear all;
clc;
format compact;

% 1. NHẬP DỮ LIỆU
Cost = input('Nhập ma trận chi phí: ');
A = input('Cung: ');
B= input('Cầu: ');

% 2. KHỞI TẠO
[m, n] = size(Cost);  % m = số hàng, n = số cột
X = zeros(m, n);      % Ma trận kết quả (phân bổ)
ICost = Cost;         % Sao lưu ma trận chi phí gốc để tính tiền

% 3. TÌM PABĐ BẰNG GÓC TÂY BẮC
A_temp = A; % Tạo bản sao Cung để NWC sử dụng
B_temp = B; % Tạo bản sao Cầu
i = 1; % Bắt đầu ở hàng 1
j = 1; % Bắt đầu ở cột 1
while (i <= m && j <= n)
    alloc_amount = min(A_temp(i), B_temp(j));
    X(i, j) = alloc_amount; % Gán vào ma trận kết quả
    A_temp(i) = A_temp(i) - alloc_amount;
    B_temp(j) = B_temp(j) - alloc_amount;

    if A_temp(i) < 1e-6 % Nếu Cung ở hàng i đã hết -> di chuyển XUỐNG
        i = i + 1;
    else % Nếu Cầu ở cột j đã hết -> di chuyển PHẢI
        j = j + 1;
    end
end
fprintf('\nBảng phân bổ:\n');
disp(X);
InitialCost = sum(sum(ICost .* X)); % Tính tổng chi phí
fprintf('==> Tổng Chi Phí: %.2f\n', InitialCost);
```

```
36
37  % 5. GỌI HÀM MODI
38  fprintf('\n[2] Phương Án Tối Ưu (từ MODI):\n');
39  % Gọi hàm modi (Hàm này được định nghĩa ở dưới)
40  [OptimalAllocation, OptimalCost] = modi(ICost, A, B, X, m, n);
41  fprintf('Bảng phân bổ Tối Ưu:\n');
42  disp(OptimalAllocation);
43  fprintf('==> Tổng Chi Phí Tối Ưu: %.2f\n', OptimalCost);
44
45
46  % HÀM PHỤ TRỢ (HELPER FUNCTIONS)
47  function [Allocation, TotalCost] = modi(Costs, Supply, Demand,
    InitialAllocation, m, n)
48      Allocation = InitialAllocation;
49      iter = 1;
50      while iter <= (m*n)
51          % fprintf(' MODI: Lần lặp %d...\n', iter);
52          % Bước 1: Xử lý suy biến (nếu cần)
53          basic_cells = (Allocation > 1e-6);
54          num_basics = sum(basic_cells(:));
55          if num_basics < m + n - 1
56              fprintf('Phát hiện suy biến! Thêm ô cơ sở 0 ảo.\n');
57              non_basic_indices = find(Allocation <= 1e-6);
58              [~, sort_idx] = sort(Costs(non_basic_indices));
59              for k = 1:length(sort_idx)
60                  if num_basics >= m + n - 1
61                      break;
62                  end
63                  idx_to_add = non_basic_indices(sort_idx(k));
64                  if ~basic_cells(idx_to_add)
65                      basic_cells(idx_to_add) = 1;
66                      num_basics = num_basics + 1;
67                  end
68              end
69          end
70          % Bước 2: Tính u, v
71          [r_basic, c_basic] = find(basic_cells);
72          u = NaN(m, 1);
73          v = NaN(1, n);
```

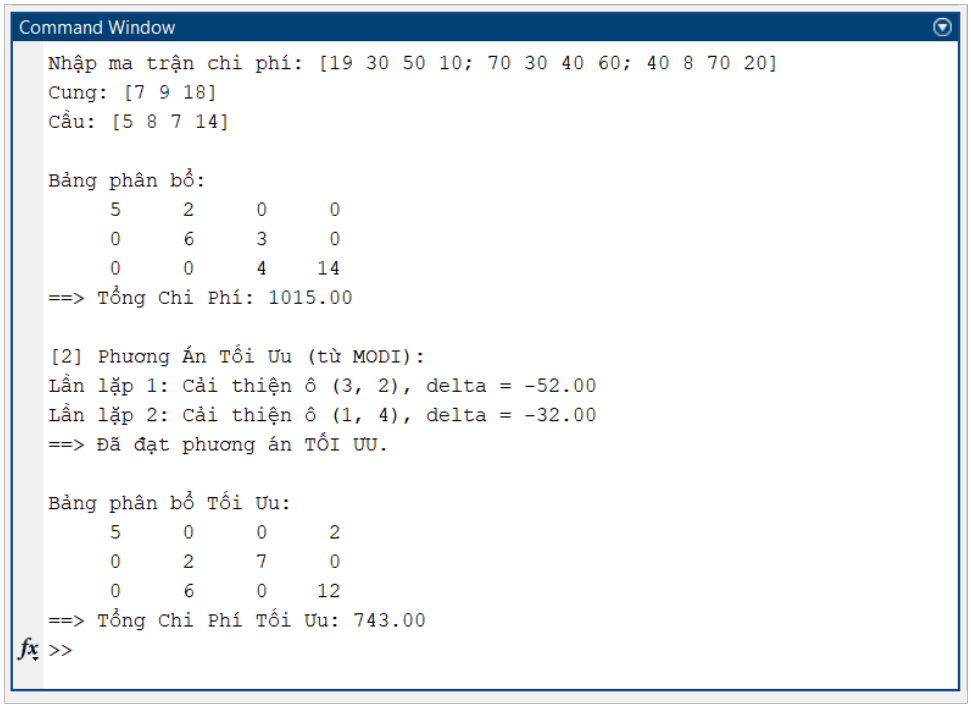
```
        u(1) = 0;
        for k = 1:(m + n)
            for i = 1:length(r_basic)
                r = r_basic(i);
                c = c_basic(i);
                if ~isnan(u(r)) && isnan(v(c))
                    v(c) = Costs(r, c) - u(r);
                elseif isnan(u(r)) && ~isnan(v(c))
                    u(r) = Costs(r, c) - v(c);
                end
            end
        end
        if any(isnan(u)) || any(isnan(v))
            u(isnan(u)) = 0;
            v(isnan(v)) = 0;
        end
        % Bước 3: Tính chi phí cơ hội & Kiểm tra tối ưu
        OppCosts = Inf(m, n);
        non_basic_cells_logical = (Allocation <= 1e-6) & ~basic_cells;
        non_basic_indices = find(non_basic_cells_logical);
        for idx = 1:length(non_basic_indices)
            r = mod(non_basic_indices(idx)-1, m) + 1;
            c = floor((non_basic_indices(idx)-1) / m) + 1;
            OppCosts(r, c) = Costs(r, c) - u(r) - v(c);
        end
        if all(OppCosts(:) >= -1e-6)
            fprintf('==> Đã đạt phương án TỐI ƯU.\n\n');
            break; % Đã tối ưu
        end
        % Bước 4: Tìm ô vào
        [min_delta, enter_idx] = min(OppCosts(:));
        [r_enter, c_enter] = ind2sub([m, n], enter_idx);
        fprintf('Lần lặp %d: Cải thiện ô (%d, %d), delta = %.2f\n', iter,
        r_enter, c_enter, min_delta);
        % Bước 5: Tìm vòng lặp (DFS)
        [loop_path, success] = find_loop(basic_cells, [r_enter, c_enter]);
        if ~success
            fprintf('Lỗi: Không tìm thấy vòng lặp. Dừng lại.\n');
            break;
```

```
        end
        % Bước 6: Tái phân bổ (Theta)
        minus_cells_idx = 2:2:size(loop_path, 1);
        minus_allocations = [];
        for i = 1:length(minus_cells_idx)
            r = loop_path(minus_cells_idx(i), 1);
            c = loop_path(minus_cells_idx(i), 2);
            minus_allocations = [minus_allocations, Allocation(r, c)];
        end
        theta = min(minus_allocations(minus_allocations > -1e-6));
        if isempty(theta)
            theta = 0;
        end
        for i = 1:size(loop_path, 1)
            r = loop_path(i, 1);
            c = loop_path(i, 2);
            if mod(i, 2) == 1
                Allocation(r, c) = Allocation(r, c) + theta;
            else
                Allocation(r, c) = Allocation(r, c) - theta;
            end
        end
        iter = iter + 1;
    end
    Allocation(Allocation < 1e-6) = 0;
    TotalCost = sum(sum(Allocation .* Costs));
end

% HÀM CON 3: TÌM VÒNG LẶP (DFS)
function [loop_path, success] = find_loop(basic_cells, start_node)
    % Hàm này tìm vòng lặp kín từ start_node, chỉ đi qua các ô basic_cells
    [r_start, c_start] = deal(start_node(1), start_node(2));
    search_cells = basic_cells;
    search_cells(r_start, c_start) = 1; % Thêm ô bắt đầu vào để tìm
    % Thử tìm theo cột trước
    [path_col, success_col] = search_path(search_cells, [r_start, c_start],
    'col');
    if success_col
        loop_path = [r_start, c_start; path_col];
```

```
        success = true;
        return;
    end
    % Thử tìm theo hàng trước
    [path_row, success_row] = search_path(search_cells, [r_start, c_start],
    'row');
    if success_row
        loop_path = [r_start, c_start; path_row];
        success = true;
        return;
    end
    loop_path = [];
    success = false;
end

function [path, success] = search_path(basic_cells, current_path_nodes,
direction)
    % Hàm đệ quy DFS để tìm vòng lặp
    current_node = current_path_nodes(end, :);
    [r, c] = deal(current_node(1), current_node(2));
    start_node = current_path_nodes(1, :);
    [r_start, c_start] = deal(start_node(1), start_node(2));
    if strcmp(direction, 'col') % Tìm trong cột 'c'
        [m, ~] = size(basic_cells);
        possible_rows = find(basic_cells(:, c) & (1:m)' ~= r);
        for i = 1:length(possible_rows)
            new_r = possible_rows(i);
            % Kiểm tra đã quay lại hàng bắt đầu (và vòng lặp có ít nhất 4 ô)
            if new_r == r_start && size(current_path_nodes, 1) >= 3
                path = [new_r, c];
                success = true;
                return;
            end
            % Tránh quay lại ô vừa đi qua
            if size(current_path_nodes, 1) > 1 && all([new_r, c] ==
            current_path_nodes(end-1, :))
                continue;
            end
            % Tiếp tục tìm kiếm (đổi hướng)
```

```
            [path_found, found] = search_path(basic_cells,
            [current_path_nodes; new_r, c], 'row');
            if found
                path = [new_r, c; path_found];
                success = true;
                return;
            end
        end
    else % Tìm trong hàng 'r'
        [~, n] = size(basic_cells);
        possible_cols = find(basic_cells(r, :) & (1:n) ~= c);
        for i = 1:length(possible_cols)
            new_c = possible_cols(i);
            % Kiểm tra đã quay lại cột bắt đầu
            if new_c == c_start && size(current_path_nodes, 1) >= 3
                path = [r, new_c];
                success = true;
                return;
            end
            % Tránh quay lại ô vừa đi qua
            if size(current_path_nodes, 1) > 1 && all([r, new_c] ==
            current_path_nodes(end-1, :))
                continue;
            end
            % Tiếp tục tìm kiếm (đổi hướng)
            [path_found, found] = search_path(basic_cells,
            [current_path_nodes; r, new_c], 'col');
            if found
                path = [r, new_c; path_found];
                success = true;
                return;
            end
        end
    end
    % Không tìm thấy đường
    path = [];
    success = false;
end
```

3.4.2 Kiểm tra thuật toán



Hình 1: Output hoàn toàn trùng với lời giải ở trên

3.5 Bảng mô tả các hàm MATLAB

Lệnh	Cú pháp sử dụng	Ý nghĩa
Lệnh cơ bản		
‘clear all‘	‘clear all‘	Xóa tất cả các biến khỏi bộ nhớ (Workspace).
‘clc‘	‘clc‘	Xóa nội dung hiển thị trên cửa sổ lệnh (Command Window).
‘format‘	‘format compact‘	Định dạng cách hiển thị kết quả (ví dụ ‘compact‘ để bỏ các dòng trống thừa).
‘input‘	‘Cost = input(’Nhập...’)‘	Hiển thị một chuỗi văn bản (prompt) và chờ người dùng nhập dữ liệu từ bàn phím.

‘fprintf‘	‘fprintf(’Chi phí: %.2f’, Cost)‘	In ra màn hình hoặc file một chuỗi văn bản đã được định dạng. (Lưu ý: dùng %)
‘disp‘	‘disp(X)‘	Hiển thị giá trị của một biến ra màn hình (không hiển thị tên biến).
Hàm xử lý Ma trận/Vector		
‘size‘	‘[m, n] = size(Cost)‘	Trả về kích thước của ma trận. ‘m‘ là số hàng, ‘n‘ là số cột.
‘zeros‘	‘X = zeros(m, n)‘	Tạo một ma trận mới có ‘m‘ hàng, ‘n‘ cột, chứa toàn giá trị 0.
‘sum‘	‘sum(sum(X))‘	Tính tổng các phần tử. ‘sum(X)‘ tính tổng theo cột. ‘sum(sum(X))‘ tính tổng toàn bộ ma trận.
‘find‘	‘indices = find(Allocation <= 1e-6)‘	Trả về các chỉ số (indices) của các phần tử trong ma trận/vector thỏa mãn một điều kiện logic.
‘sort‘	‘[∼, sort_idx] = sort(Costs(. . .))‘	Sắp xếp các phần tử của một vector. Cú pháp ‘[∼, idx]‘ chỉ lấy vector chỉ số (vị trí) sau khi sắp xếp. (Lưu ý: dùng _ và ∼)
‘length‘	‘length(sort_idx)‘	Trả về độ dài (số phần tử) của một vector.
‘min‘	‘min(A_temp(i), B_temp(j))‘	Trả về giá trị nhỏ nhất trong các đối số đầu vào (hoặc trong một vector).
‘min‘	‘[val, idx] = min(OppCosts(:))‘	Trả về giá trị nhỏ nhất (‘val‘) và chỉ số tuyến tính (‘idx‘) của giá trị đó trong ma trận. ‘OppCosts(:)‘ biến ma trận thành vector cột.
‘ind2sub‘	‘[r, c] = ind2sub([m, n], idx)‘	Chuyển đổi một chỉ số tuyến tính (‘idx‘) của ma trận kích thước ‘[m, n]‘ thành chỉ số hàng (‘r‘) và cột (‘c‘).

‘isempty‘	‘if isempty(theta)‘	Kiểm tra xem một ma trận hoặc vector có rỗng hay không (không chứa phần tử nào).
‘deal‘	‘[r, c] = deal(node(1), node(2))‘	Gán các giá trị đầu vào cho các biến đầu ra tương ứng. Tương đương ‘r = node(1)‘, ‘c = node(2)‘.
Hàm xử lý Số/Logic		
‘NaN‘	‘u = NaN(m, 1)‘	Tạo ra một vector/ma trận chứa giá trị ‘NaN‘ (Not a Number), thường dùng để khởi tạo giá trị chưa biết.
‘isnan‘	‘if isnan(u(r))‘	Kiểm tra xem một phần tử có phải là ‘NaN‘ hay không.
‘Inf‘	‘OppCosts = Inf(m, n)‘	Tạo ra một ma trận chứa giá trị ‘Inf‘ (Infinity - Vô cực).
‘any‘	‘if any(isnan(u))‘	Kiểm tra xem *có bất kỳ* phần tử nào trong một vector logic là ‘true‘ (khác 0) hay không.
‘all‘	‘if all(OppCosts(:) >= -1e-6)‘	Kiểm tra xem *tất cả* các phần tử trong một vector logic là ‘true‘ (khác 0) hay không.
‘mod‘	‘r = mod(index-1, m) + 1‘	Phép toán Modulo (chia lấy phần dư).
‘floor‘	‘c = floor((index-1) / m) + 1‘	Làm tròn một số xuống số nguyên gần nhất (ví dụ: ‘floor(3.9)‘ là ‘3‘).
‘strcmp‘	‘if strcmp(direction, ’col’)‘	So sánh hai chuỗi ký tự (String Compare). Trả về ‘true‘ nếu hai chuỗi giống hệt nhau.

4 KẾT LUẬN

4.1 Tổng kết kết quả đạt được

Trong phạm vi đồ án này, nhóm đã hoàn thành các mục tiêu đề ra:

- **Về lý thuyết:** Hệ thống hóa thành công các khái niệm cốt lõi của Quy hoạch tuyến tính và thuật toán Đơn hình (Simplex).
- **Về mô hình hóa:** Xây dựng được mô hình toán học cho bài toán vận tải dựa trên ma trận cung - cầu và chi phí.
- **Về thực hành:** Lập trình thành công công cụ giải bài toán vận tải trên MATLAB, kết hợp phương pháp *Góc Tây Bắc* (tìm phương án đầu) và *MODI* (tối ưu hóa).
- **Về kiểm thử:** Chương trình hoạt động ổn định và cho kết quả chính xác trên các bộ dữ liệu thử nghiệm.

4.2 Đánh giá và Khả năng ứng dụng

Ưu điểm:

- Giải pháp toàn diện kết hợp chặt chẽ giữa cơ sở toán học và công cụ tính toán.
- Quy trình giải thuật rõ ràng, dễ hiểu, phù hợp cho mục đích học tập và nghiên cứu.

Hạn chế:

- Hiệu suất tính toán chưa tối ưu bằng các hàm chuyên dụng có sẵn (như `linprog`).
- Mô hình còn đơn giản, chưa xét đến các ràng buộc phức tạp (thời gian, sức chứa xe...).
- Chưa có giao diện đồ họa (GUI), việc nhập liệu còn thủ công.

Ứng dụng thực tế: Giải pháp có thể áp dụng cho các doanh nghiệp vừa và nhỏ trong lĩnh vực logistics để lập kế hoạch vận chuyển hàng hóa nhằm tối thiểu hóa chi phí.

5 Tài liệu tham khảo

Tài liệu

[1] Sitanggang, E. A., & Ahyaningsih, F. (2022). IMPLEMENTATION OF NORTH WEST CORNER METHOD AND MODIFIED DISTRIBUTION METHOD IN OPTIMIZING FISH DISTRIBUTION COSTS USING MATLAB PROGRAM. *ZERO: Jurnal Sains, Matematika dan Terapan - Jurnal UINSU*, **6**(2). (Tài liệu trực tuyến: `http://jurnal.uinsu.ac.id/index.php/zero/article/view/14556`)

[2] Bertsimas, D. & Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA. (Nguồn trực tuyến: `https://archive.org/details/introductiontolinearoptimizationber` - Internet Archive)

[3] Goemans, M. (2014). *Linear Programming* (Lecture Notes for Course 18.438). Massachusetts Institute of Technology (MIT). (Tài liệu tham khảo: `http://web.mit.edu/goemans/www/lpnotes310.pdf`)

[4] Shameem, S. (2022). *An experiment with simplex method for solving linear programming problems* (Doctoral dissertation, Brac University).

[5] Shamir, R. (1987). The Efficiency of the Simplex Method: A Survey. *Management Science*, 33, 301–334.

[6] Qi, H., & Hall, J. (2015). Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10, 119–142.

[7] Inayatullah, S., Touheed, N., & Imtiaz, M. (2015). A Streamlined Artificial Variable Free Version of Simplex Method. *PLoS ONE*, 10.

[8] Cunningham, W. (1979). Theoretical Properties of the Network Simplex Method. *Math. Oper. Res.*, 4, 196–208.

[9] Cerdà, V., De La Cerda, J., & Idris, A. (2016). Optimization using the gradient and simplex methods. *Talanta*, 148, 641–648.

[10] Nelder, J., & Mead, R. (1965). A Simplex Method for Function Minimization. *Comput. J.*, 7, 308–313.

[11] Brooks/Cole. (2003). *The Big M Method (Chapter 04)* [Lecture slides]. Thomson Learning, Inc.