VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

MATHEMATICAL MODELING (CO2011)

Assignment

Stochastic Programming and Applications

Advisor: Nguyen Tien Thinh
Students: Nguyen Thai Son - 2112198.
Vu Lam Hoang Dai - 2110992.
Vu Linh Cuong - 2110890.
Hoang Minh Hai Dang - 2110120.
Phan Duc Dat - 2113152.

HO CHI MINH CITY, JANUARY 2024

Contents

1 Member list & Workload

No.	Fullname	Student ID	Problems	Percentage of work
1	Nguyễn Thái Sơn	2112198	- Cơ sở lý thuyết - Thực hiện code PROBLEM 1 - Soạn báo cáo	100%
2	Vũ Lâm Hoàng Đại	2110992	- Cơ sở lý thuyết - Thực hiện code PROBLEM 1 - Soạn báo cáo	100%
3	Vũ Linh Cường	2110890	- Cơ sở lý thuyết - Mô hình cho PROBLEM 1 - Soạn báo cáo	100%
4	Hoàng Minh Hải Đăng	2110120	- Xây dựng mô hình - Giải thuật SLP-EPDR - Soạn báo cáo	100%
5	Phan Đức Đạt	2113152	- Xây dựng mô hình - Giải thuật SLP-EPDR - Soạn báo cáo	100%

2 Problem 1

2.1 Giới thiệu

Với mục tiêu chung là nghiên cứu và ứng dụng quy hoạch tuyến tính ngẫu nhiên để giải quyết các vấn đề phức tạp trong quản lý và quy hoạch, trong dự án này, nhóm đặc biệt tập trung vào nghiên cứu và thiết lập mô hình quy hoạch tuyến tính ngẫu nhiên hai giai đoạn có can thiệp (2-SLPWR). Dưới đây là những gì có thể đạt được khi thực hiện dự án này:

- **Hiểu biết các khái niệm trong quy hoạch ngẫu nhiên:** Dự án giúp cung cấp sự hiểu biết sâu sắc về các khái niệm cơ bản và ứng dụng của quy hoạch ngẫu nhiên, đặc biệt là trong ngữ cảnh quy hoạch tuyến tính ngẫu nhiên.
- **Mô hình hóa dữ liệu và thiết lập bài toán quy hoạch ngẫu nhiên:** Thông qua việc mô phỏng dữ liệu và xây dựng bài toán quy hoạch ngẫu nhiên, nhóm có cơ hội học cách tạo ra các mô hình xác suất và áp dụng chúng vào thế giới thực, đặc biệt là trong lĩnh vực quản lý và quy hoạch.
- **Phân tích tình huống và giải bài toán quy hoạch ngẫu nhiên:** Dự án đặt ra nhiệm vụ phân tích các tình huống thực tế và giải quyết bài toán quy hoạch ngẫu nhiên hai giai đoạn có can thiệp. Điều này giúp nhóm xây dựng kỹ năng trong việc áp dụng lý thuyết vào thực tế.
- **Xây dựng ứng dụng:** Nhóm sẽ có cơ hội xây dựng ứng dụng thực tế từ các mô hình và kết quả thu được, đưa quy hoạch tuyến tính ngẫu nhiên từ khía cạnh lý thuyết sang ứng dụng thực tế, làm nổi bật tính ứng dụng của kiến thức đã học.

2.2 Cơ sở lý thuyết

2.2.1 One-Stage Stochastic linear programming - No recourse (1-SLP)

DEFINITION

Quy hoạch tuyến tính ngẫu nhiên một giai đoạn (không có sự can thiệp từ các yếu tố khác bên ngoài).

Xét một chương trình $LP(\alpha)$ được tham số hóa bởi vector ngẫu nhiên α:

$$\begin{gathered}\text{Minimize } Z = g(x) = f(x) = c^T \cdot x = \sum_{j=1}^{n} c_j \cdot x_j \\ \text{s.t } Ax = b, \text{ (certain constraints)} \\ \text{and } Tx \geq h \text{ (stochastic constraints)}\end{gathered}$$

Với giả định như sau

1. Ma trận $T = T(\alpha)$ và vector $h = h(\alpha)$ biểu diễn sự không chắc chắn thông qua các ràng buộc ngẫu nhiên.

$$T(\alpha)x \geq h(\alpha) \Longleftrightarrow \alpha_1 x_1 + ... + \alpha_n x_n \geq h(\alpha)$$

2. Các bộ giá trị (T, h) là không xác định: Chúng chỉ được biết khi có một trường hợp của mô hình xảy ra, $h(\alpha)$ chỉ phụ thuộc vào một biến ngẫu nhiên α_j nào đó.

3. **Sự không chắc chắn** được biểu diễn thông qua phân phối xác xuất của tham số ngẫu nhiên $(\alpha_j) = \alpha$ nên do đó quy hoạch tuyến tính là một trường hợp suy biến của quy hoạch tuyến tính ngẫu nhiên khi α_j là một hằng số.

APPROACH 1: use Chain constraint and Acceptable risk

- Thay thế $Tx \geq h$ bằng ràng buộc xác xuất $\mathbb{P}[Tx \geq h] \geq p$ với độ tin cậy trong khoảng $p \in (.5, 1)$ (được quyết định bởi người xử lí vấn đề).
  Chương trình quy hoạch tuyến tính được định nghĩa phía trên với tham số ngẫu nhiên $\alpha = [\alpha_1, \alpha_2, ...]$ được gọi là chương trình quy hoạch tuyến tính với ràng buộc xác suất (hoặc là 1-SLP).

- Risk (rủi ro) được quan tâm một cách rõ ràng, nếu được định nghĩa như sau:

$$\text{acceptable risk } r_x := \mathbb{P}[Not(Tx \geq h)] = \mathbb{P}[Tx \leq h] \leq 1 - p$$

  Khi đó $(1 - p)$ là acceptable risk lớn nhất (rủi ro lớn nhất có thể chấp nhận được). Ràng buộc xác suất $Tx \leq h$ mang ý nghĩa rằng rủi ro chấp nhận được r_x nhỏ hơn một giá trị lớn nhất được cho trước $1 - p \in (0, 1)$.

APPROACH 2: for stochastic constraint $T(\alpha)x \leq h(\alpha)$

Sử dụng phân tích kịch bản của $T(\alpha)x \leq h(\alpha)$. Với mỗi kịch bản $(T^s; h^s), s = 1, ..., S$, ta giải quyết bài toán:

$$\text{Minimize } \{f(x) = c^T.x; \quad \text{s.t.} \quad Ax = b, \quad T^s x \leq h^s$$

Loại chương trình này nhắm đến một đối tượng tuyến tính cụ thể trong khi tính toán cho một hàm xác suất có mối liên hệ với nhiều kịch bản khác. Bằng cách này, có thể tìm được giải pháp tổng quát bằng cách chọn lựa giữa các hướng giải quyết của từng kịch bản $x^s (s = 1, ..., S)$.
Ưu điểm: Mỗi một kịch bản là một chương trình quy hoạch tuyến tính.
Nhược điểm: Phân phối rời rạc $\rightarrow$ mô hình mixed-integer LP.

2.2.2 Generic Stochastic Programming (GSP) with RECOURSE

DEFINITION

Chương trình ngẫu nhiên 2 giai đoạn (bài toán 2-SP tổng quát). Được mở rộng từ định nghĩa của 1-SLP có dạng:

$$\text{2-SP:} \quad \min_x g(x) \quad \text{với } g(x) = f(x) + E_\omega[\upsilon(\chi, \omega)]$$

tại đó, $x = (x_1, x_2, ..., x_n)$ là biến quyết định của giai đoạn 1, $f(x)$ có thể là một hàm tuyến tính hoặc không, là một phần của hàm mục tiêu tổng $g(x)$

Giá trị kỳ vọng $Q(x) := E_\omega[\upsilon(\chi, \omega)]$ của hàm

$$\upsilon : \mathbb{R}^n \times \mathbb{R}^S \rightarrow \mathbb{R}$$

Dựa trên ảnh hưởng của các kịch bản $\omega.Q(x)$ là giá trị tối ưu cho giai đoạn 2 cụ thể.

$$\min_{y \in \mathbb{R}^P} q\ y \mid \text{subject to } T \cdot x + W \cdot y = h$$

Các vector $\alpha = \alpha(\omega)$ và $y = y(\omega)$ được đặt tên correction, tuning hoặc recourse biến quyết định, chỉ được biết sau kiểm chứng kinh nghiệm e.

Tóm lại, chúng ta sẽ tối ưu hóa (tối thiểu hóa) tổng chi phí dự kiến $g(x) = f(x) + Q(x)$, đồng thời thỏa mãn được:

$$W \cdot y(\omega) = h(\omega) - T(\omega) \cdot x$$

Ở đây, W là ma trận can thiệp $m \times p$, bắt đầu với một trường hợp đơn giản $m = 1$, q là vector chi phí đơn vị can thiệp, có cùng miền với y và $y = y(\omega) \in \mathbb{R}^p$.

2.2.3 Two-Stage Stochastic linear programming (2-SLP)

Two-stage SLP Recourse model - (simple form)
DEFINITION

Quy hoạch tuyến tính ngẫu nhiên hai giai đoạn có sự can thiệp: 2-SLPWR, hoặc chính xác hơn là hành động khắc phục có chi phí phạt được đặc tả như sau:

$$\text{2-SLP}: \quad \min_{x \in X} c^T \cdot x + \min_{y(\omega) \in Y} E_\omega[q \cdot y]$$

hoặc ở dạng tổng quát

$$\text{2-SLP}: \min_{x \in X, y(\omega) \in Y} E_\omega[c^T \cdot x + \upsilon(\chi, \omega)]$$

với $\upsilon(\chi, \omega) := q \cdot y$
phụ thuộc vào

$$Ax = b \quad \text{First Stage Constraint,}$$
$$T(\omega) \cdot x + W \cdot y(\omega) = h(\omega) \text{ Second Stage Constraints}$$
$$\text{or shortly } W \cdot y = h(\omega) - T(\omega) \cdot x$$

Chương trình SLP này chỉ định chương trình trước đó 2-SP đến một mục tiêu cụ thể - một hàm tổng mục tiêu ngẫu nhiên $g(x)$ có

1. Hàm quyết định $f(x)$ - vừa tính toán, vừa là hàm tuyến tính
2. Hàm xác suất $\upsilon(\chi, \omega)$ có liên hệ với nhiều kịch bản ω khác.

$y = y(\chi, \omega) \in R^p_+$ được gọi là biến hành động can thiệp để quyết định x và hiện thực hóa ω. Hành động can thiệp được xem là hành động **Penalize corrective** trong SLP. Penalize correction được biểu hiện thông qua kì vọng $Q(x) = E_\omega[\upsilon(\chi, \omega)]$.

Two-stage SLP Recourse model - (canonical form)
DEFINITION

Chương trình quy hoạch tuyến tính có can thiệp (2-SLPWR). Dạng chuẩn của quy hoạch tuyến tính ngẫu nhiên 2 giai đoạn có can thiệp được xây dựng như sau:

$$\text{2-SLP}: \quad \min_x g(x) \quad with$$
$$g(x) := c^T \cdot x + \upsilon(y)$$
$$\text{subject to (s.t.) } Ax = b \text{ where } x \in X \subset \mathbb{R}^n, x \geq 0$$
$$\upsilon(z) := \min_{y \in \mathbb{R}^p_+} q\ y \text{ subject to } W \cdot y = h(\omega) - T(\omega) \cdot x =: z$$

Tại đó $\upsilon(y) := \upsilon(\chi, \omega)$ là giá trị của hàm giai đoạn 2, và $y = y(x, \omega) \in \mathbb{R}^p_+$ là hành động can thiệp quyết định x và hiện thực hóa ω.

1. Chi phí can thiệp dự kiến của quyết định χ là $Q(\chi) := E_\omega[\upsilon(\chi, \omega)]$. Do đó về tổng quát chúng ta cần giảm thiểu tổng chi phí dự kiến $\min_{x\in\mathbb{R}^n, y\in\mathbb{R}^p_+} c^T \cdot x + Q(\chi)$.

2. Thiết kế một biến quyết định thứ 2 $y(\omega)$, như vậy chúng ta có thể chỉnh sửa cũng như phản ứng với ràng buộc ban đầu theo một cách logic (tối ưu).

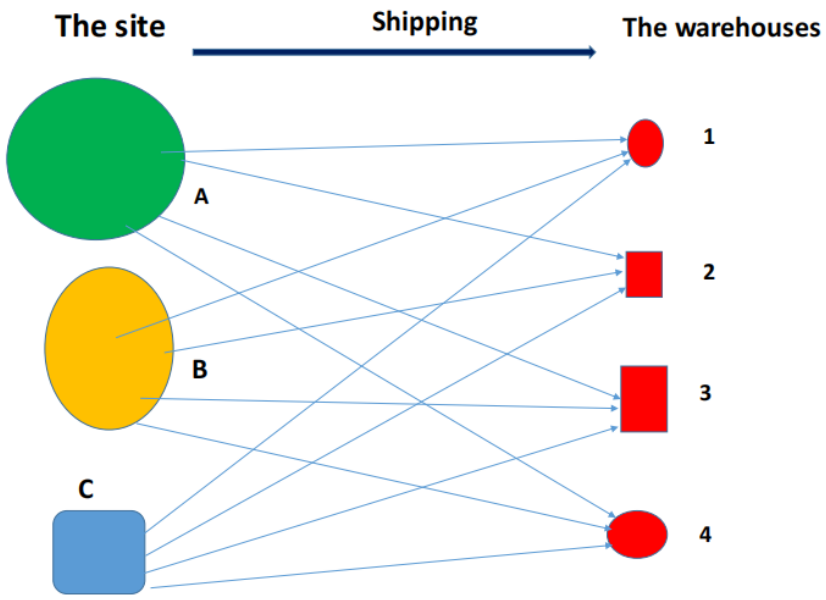
$$\chi - - - - - - T, h, \omega - - - - - - - \rightarrow y$$

3. Giá trị tối ưu của quy hoạch tuyến tính ngẫu nhiên giai đoạn 2 là $\upsilon_* = \upsilon(y^*)$, với $y^* = y^*(\chi, \omega)$ là nghiệm tối ưu của nó, ở đây là $y^* \in \mathbb{R}^p_+$. Giá trị tối ưu tổng là $c^T \cdot x^* + \upsilon(y^*)$.

2.2.4 Industry - Manufacturing

Xét một công ty F có nhà sản xuất sản xuất ra n sản phẩm.

Có những bộ phận(linh kiện nhỏ) được đặt hàng bởi m bên thứ 3 (the site). Hình dưới đây minh họa cho kế hoạch vận chuyển hàng hóa giữa công ty F với $m = 3$ nhà cung cấp và $n = 4$ địa điểm sản xuất(products, warehouse).



Một đơn vị sản xuất i yêu cầu $a_{i,j} \geq 0$ đơn vị j, trong đó $i = 1, ..., n$ và $j = 1, ..., m$. Yêu cầu của sản phẩm được mô hình bởi vector $\omega = D = (D_1, D_2, ..., D_n)$.

Vấn đề giai đoạn hai

Với giá trị thực tế quan sát được $d = (d_1, d_2, ..., d_n)$ của vector nhu cầu ngẫu nhiên D, có thể hoạch định được giải pháp tốt nhất về mặt cung ứng sản phẩm bằng cách giải bài toán quy hoạch tuyến tính ngẫu nhiên (SLP) với biến quyết định $z = (z_1, z_2, ..., z_n)$ - số lượng đơn vị đã sản xuất, mặt khác, biến quyết định khác là $y = (y_1, y_2, ..., y_n)$ - số linh kiện còn lại.

$$\text{LSP} : \min_{z,y} Z = \sum_{i=1}^{n} (l_i - q_i)\ z_i - \sum_{j=1}^{m} s_j y_j$$

Tại đó, $s_j < b_j$ (được định nghĩa là chi phí đặt hàng trước trên từng đơn vị linh kiện j) và $x_j, j = 1, ..., m$ là số linh kiện được đặt trước khi đưa vào sản xuất.

$$subject\ to \begin{cases} y_j = x_j - \sum_{i=1}^{n} a_{ij} z_i, \quad j = 1, ..., m \\ 0 \le z_i \le d_i, i = 1, ..., n; \quad y_j \ge 0, j = 1, ..., m. \end{cases}$$

Toàn bộ mô hình (của giai đoạn hai) có thể biểu diễn như sau

$$MODEL = \begin{cases} \min_{z,y} Z = c^T \cdot z - s^T \cdot y \\ \text{với } c = (c_i := l_i - q_i) \text{ là hệ số chi phí} \\ y = x - A^T z, \text{ tại đó } A = [a_{ij}] \text{ là một ma trận } n \times m \\ 0 \le z \le d, \quad y \ge 0. \end{cases}$$

Quan sát nghiệm của bài toán này, ta thấy các vector z, y phụ thuộc vào thực tế của vector d dựa trên nhu cầu ngẫu nhiên $\omega = D$ cũng như kết quả từ giai đoạn một $x = (x_1, x_2, ..., x_m)$

Vấn đề giai đoạn một

Toàn bộ mô hình 2-SLPWR được dựa trên một nguyên tắc phổ biến rằng: **sản xuất ≤ nhu cầu**.

Dựa trên phương hướng tiếp cận lấy phân phối làm cơ sở, đặt $Q(x) := E[Z(z, y)] = E_\omega[x, \omega]$ biểu thị cho giá trị tối ưu của vấn đề với $b = (b_1, b_2, ..., b_n)$ được xây dựng bởi chi phí hàng đặt trước b_j cho mỗi đơn vị của bộ phận j (trước khi biết được nhu cầu). Số lượng x_j được quyết định bởi phương trình tối ưu sau đây.

$$\min g(x, y, z) = b^T \cdot x + Q(x) = b^T \cdot x + E[Z(z)]$$

với $Q(x) = E[\omega Z] = \sum_{i=1}^{n} p_i c_i z_i$ được xem xét đối với phân phối xác suất của $\omega = D$.

Phần đầu tiên của hàm mục tiêu đại diện cho chi phí đặt hàng trước và x. Ngược lại, phần thứ hai đại diện cho chi phí kỳ vọng của kế hoạch sản xuất tối ưu , được xác định bởi số lượng hàng đã đặt z, đã chịu ảnh hưởng của nhu cầu ngẫu nhiên $D = d$ với mật độ xác suất của chúng.

Phân tích đề bài

Đề bài mô tả về một vấn đề trong lĩnh vực quản lý sản xuất công nghiệp, nơi một doanh nghiệp F sản xuất n sản phẩm và cần đặt hàng các bộ phận từ m nhà cung cấp bên thứ ba. Dưới đây là phân tích của đề bài:

- Mô tả chung:
  - Doanh nghiệp công nghiệp F sản xuất n sản phẩm.
  - Có m nhà cung cấp bên thứ ba cung cấp các bộ phận khác nhau.
  - Mô hình kế hoạch vận chuyển từ nhà cung cấp đến địa điểm sản xuất, với $m = 3$ nhà cung cấp và $n = 4$ địa điểm sản xuất.
- Quá trình sản xuất:
  - Mỗi sản phẩm i yêu cầu ít nhất aij đơn vị của bộ phận j.
  - Nhu cầu cho các sản phẩm được mô hình hóa như một vector ngẫu nhiên $\omega = D = (D_1, D_2, ..., D_n)$
- Yêu cầu đề bài:
  - Vấn Đề Giai Đoạn Thứ Hai (SLP):

* Tìm kế hoạch sản xuất tối ưu dựa trên nhu cầu ngẫu nhiên, thông qua giải quyết một vấn đề tuyến tính ngẫu nhiên với các biến quyết định là số lượng sản phẩm và bộ phận.

– Vấn Đề Giai Đoạn Thứ Nhất:

* Dựa trên quy tắc sản xuất $\leq$ nhu cầu và sử dụng phương pháp dựa trên phân phối để tối ưu hóa giá trị kế hoạch sản xuất.

2.3 Giải quyết vấn đề

2.3.1 Hiện thực mô hình

2.3.1.a Giới thiệu về thư viện GAMS

Khái niệm:
GAMS, hay General Algebraic Modeling System, là một hệ thống mô hình hóa toán học và ngôn ngữ lập trình được thiết kế để giải quyết các bài toán tối ưu hóa toán học và các vấn đề liên quan. GAMS cung cấp một cách linh hoạt và mạnh mẽ để mô hình hóa và giải quyết nhiều loại bài toán phức tạp trong nhiều lĩnh vực, bao gồm quản lý nguồn lực, kế hoạch sản xuất, tài chính, và nghiên cứu toán học.
Tính chất:

- **Đa Dạng Bài Toán:**GAMS hỗ trợ giải quyết một loạt các bài toán tối ưu hóa toán học, từ tối ưu hóa tuyến tính đến tối ưu hóa nguyên tuyến tính và nhiều dạng bài toán tối ưu hóa khác nhau.
- **Ngôn Ngữ Mô Hình Hóa:**GAMS cung cấp một ngôn ngữ mô hình hóa mạnh mẽ, dễ đọc và dễ hiểu, giúp người mô hình có thể diễn đạt một cách rõ ràng cấu trúc của bài toán.
- **Hỗ Trợ Nhiều Ngôn Ngữ:**GAMS không chỉ giới hạn trong việc sử dụng một ngôn ngữ lập trình cụ thể, mà còn hỗ trợ nhiều ngôn ngữ như C++, Java, Python, và nhiều ngôn ngữ khác, tạo thuận lợi cho việc tích hợp và sử dụng trong nhiều môi trường lập trình.
- **Hiệu Suất Cao:**GAMS được tối ưu hóa để đạt được hiệu suất cao trong việc giải quyết các bài toán lớn và phức tạp. Thuật toán tiên tiến được sử dụng để giải quyết các vấn đề tối ưu hóa một cách hiệu quả.

2.3.1.b Sử dụng GAMS (GamsPy)

- **Khai báo biến quyết định:**
  - x: Biến quyết định cho lượng hàng đặt.
  - y, z: Biến ẩn cho kế hoạch sản xuất và tồn kho của mỗi kịch bản.
- **Khai báo các tham số:** Dựa trên các dữ liệu tham số được mô phỏng, thực hiện khai báo các tham số tham gia trong mô hình
- **Định nghĩa ràng buộc:** Định nghĩa ràng buộc dựa trên mô hình Two-Stage SLP.
- **Xây dựng hàm mục tiêu:** Thực hiện xây dựng hàm mục tiêu dựa trên các phương trình được xây dựng sẵn theo mô hình tối ưu hóa tổng chi phí.
- **Tạo mô hình GAMS:**

- Xây dựng biến chưa mô hình problem1 = Container()
- Thực hiện xây dựng mô hình dựa trên các thông tin được định nghĩa và xây dựng

```
problem1 = Model(container=problem1, name='problem_1',
    equations=problem1.getEquations(), problem='MIP', sense=Sense.MIN,
    objective=objective_func)
```

2.3.2 Mô phỏng dữ liệu

2.3.2.a Giới thiệu về thư viện NumPy

Khái niệm
NumPy (Numerical Python) là một thư viện quan trọng trong ngôn ngữ lập trình Python, chủ yếu được sử dụng để thực hiện các phép toán số học và thống kê trên mảng đa chiều. NumPy cung cấp một đối tượng mảng nhanh và hiệu quả, cùng với nhiều hàm số và công cụ tính toán cho các tác vụ khoa học dữ liệu, máy học và tính toán khoa học.
Tính chất:

- **Đối Tượng Mảng Nhanh và Hiệu Quả:** NumPy tập trung vào việc cung cấp đối tượng mảng đa chiều, ndarray, giúp thực hiện các phép toán số học và thống kê một cách nhanh chóng và hiệu quả.
- **Tạo Dữ Liệu Ngẫu Nhiên:** NumPy cung cấp một mô-đun numpy.random cho phép tạo dữ liệu ngẫu nhiên dựa trên các phân phối xác suất khác nhau. Điều này làm cho NumPy trở thành một công cụ mạnh mẽ để sinh dữ liệu ngẫu nhiên để mô phỏng và kiểm thử trong các ứng dụng thống kê và máy học.
- **Hỗ Trợ Nhiều Loại Dữ Liệu:**NumPy hỗ trợ nhiều loại dữ liệu số học, bao gồm số nguyên, số thực và số phức. Điều này làm cho nó trở thành một công cụ mạnh mẽ cho xử lý và phân tích dữ liệu số.

2.3.2.b Sinh dữ liệu bằng thư viện NumPy

Ta mô phỏng các yếu tố sau của mô hình:

- Các vector b, l, q, s: Đại diện cho các chi phí và giá trị thu hồi liên quan đến quá trình sản xuất.
  - **Vector b: Chi phí đặt hàng trước**
    Vector này đại diện cho chi phí đặt hàng trước khi bắt đầu sản xuất. Mỗi phần tử của b tương ứng với chi phí đặt hàng trước cho một bộ phận hoặc thành phần cụ thể. Ví dụ: Nếu chúng ta sản xuất xe đạp, bj có thể đại diện cho chi phí đặt hàng một bộ lốp từ nhà cung cấp. Nếu $b = [2, 5]$, việc đặt hàng một bộ lốp sẽ tốn 2 đô la, và việc đặt hàng một bộ bàn đạp sẽ tốn 5 đô la.
  - **Vector l: Chi phí bổ sung để đáp ứng nhu cầu**
    Vector này chỉ ra chi phí bổ sung phát sinh cho mỗi sản phẩm để đáp ứng một đơn vị nhu cầu vượt quá chi phí đặt hàng trước.
    Ví dụ: Tiếp tục với ví dụ về xe đạp, nếu nhu cầu cho một mô hình xe đạp cụ thể tăng đột ngột, công ty có thể phải chi tiền gửi hàng nhanh hoặc chi phí làm thêm giờ để đáp ứng nhu cầu này. Nếu $l = [3, 4]$, điều này có nghĩa là chúng ta chi thêm 3 đô la cho mỗi đơn vị của mô hình xe đạp đầu tiên và 4 đô la cho mỗi đơn vị của mô hình thứ hai để đáp ứng nhu cầu vượt quá.

  - **Vector q: Giá bán mỗi đơn vị**
    Vector q chỉ ra giá bán mỗi đơn vị cho mỗi sản phẩm mà công ty sản xuất.
    Ví dụ: Nếu công ty bán hai mô hình xe đạp với giá khác nhau và $q = [200, 250]$, điều này có nghĩa là giá bán cho mô hình đầu tiên là 200 đô la mỗi đơn vị và 250 đô la mỗi đơn vị cho mô hình thứ hai.
  - **Vector s: Giá trị thu hồi**
    Vector này đại diện cho giá trị còn lại mỗi đơn vị của mỗi bộ phận sau khi đáp ứng nhu cầu. Nó phản ánh số tiền thu hồi được từ việc bán lại hoặc tái chế tồn kho dư thừa.
    Ví dụ: Nếu có lốp và bàn đạp dư thừa sau khi đáp ứng nhu cầu, và giá trị còn lại là $s = [1, 0.50]$, công ty có thể thu hồi 1 đô la cho mỗi bộ lốp thừa và 0.50 đô la cho mỗi bộ pedal thừa thông qua một cuộc giảm giá hoặc bán chúng như làm phụ tùng thay thế.

- Vector c (Hệ số chi phí sản xuất): Vector này đại diện cho các hệ số chi phí, chẳng hạn như chi phí bổ sung và giá bán, đối với mỗi sản phẩm. Các thành phần của c có thể được tính bằng cách trừ giá bán (q) từ chi phí bổ sung (l), chỉ ra chi phí cần phải chi trả để sản xuất mỗi sản phẩm.
- Ma trận A: Ghi lại mối quan hệ giữa các bộ phận và sản phẩm.
- Vector $\omega = D = (D_1, D_2, ..., D_n)$: Đại diện cho nhu cầu không chắc chắn của mỗi sản phẩm.

2.4 Code

```
import numpy as np
import pandas as pd
from gamspy import Container, Set, Parameter, Variable, Equation, Model, Sum, Sense

# User Input Parameters
m = int(input("Enter the number of sites (m): "))
n = int(input("Enter the number of warehouse (n): "))
noScen = 2
warehouses = ['Warehouse ' + str(i) for i in range(1, n+1)]
scenarios = ['Scenario ' + str(k) for k in range(1, noScen+1)]

# Generate Datas
cost_l = np.random.randint(50, 200, n)
cost_q = np.random.randint(100, 500, n)
cost_b = np.random.randint(0, 20, m)
cost_s = np.random.randint(0, 10, m)
req = np.random.randint(0, 10, n * m)
needs = np.random.randint(0, 15, size=(n * noScen))

# Reshape Datas
req_reshaped = req.reshape((n, m))
needs_reshaped = needs.reshape((n, noScen))

# Condition Check (sj < bj)
for j in range(m):
```

```
    if cost_s[j] >= cost_b[j]:
        cost_s[j] = cost_b[j] - 10

# Create Model
problem1 = Container()

# Define Sets
i = Set(container=problem1, name='i', description='Warehouses', records=['Warehouse ' +
    str(i) for i in range(1, n+1)])
j = Set(container=problem1, name='j', description='Sites', records=['Site ' + str(j)
    for j in range(1, m+1)])
k = Set(container=problem1, name='k', description='Scenarios', records=['Scenario ' +
    str(k) for k in range(1, noScen+1)])

# Set Parameters
d = Parameter(container=problem1, name='d', domain=[i, k], description='Products
    needs', records=needs_reshaped)
b = Parameter(container=problem1, name='b', domain=[j], description='Preorder cost',
    records=cost_b)
l = Parameter(container=problem1, name='l', domain=[i], description='Additional cost',
    records=cost_l)
q = Parameter(container=problem1, name='q', domain=[i], description='Selling price',
    records=cost_q)
s = Parameter(container=problem1, name='s', domain=[j], description='Salvage values',
    records=cost_s)
c = Parameter(container=problem1, name='c', domain=[i], description='Cost
    coefficients', records=cost_l-cost_q)
prob = Parameter(container=problem1, name='p', domain=[k], description='Scenario
    probabilities', records=np.array([1 / 2, 1 / 2]))
A = Parameter(container=problem1, name='A', description='Requirements of products',
    domain=[i, j], records=req_reshaped)

# Set Variables
x = Variable(container=problem1, name='x', type='positive', domain=j,
    description='Number of parts to be ordered before production')
y = Variable(container=problem1, name='y', type='positive', domain=[j, k],
    description='Number of parts left in inventory')
z = Variable(container=problem1, name='z', type='positive', domain=[i, k],
    description='Number of units produced')

# Build Equations
need_constraint = Equation(container=problem1, name='need_constraint', domain=[j, k])
need_constraint[j, k] = y[j, k] == x[j] - Sum(i, A[i, j] * z[i, k])

non_negative_order_constraint = Equation(container=problem1,
    name='non_negative_order_constraint', domain=j)
non_negative_order_constraint[j] = x[j] >= 0

non_negative_inventory_constraint = Equation(container=problem1,
    name='non_negative_inventory_constraint', domain=[j, k])
non_negative_inventory_constraint[j, k] = y[j, k] >= 0

non_negative_production_constraint = Equation(container=problem1,
```

```
    name='non_negative_production_constraint', domain=[i, k])
non_negative_production_constraint[i, k] = z[i, k] >= 0

production_need_constraint = Equation(container=problem1,
    name='production_need_constraint', domain=[i, k])
production_need_constraint[i, k] = z[i, k] <= d[i, k]

inventory_salvage_constraint = Equation(container=problem1,
    name='inventory_salvage_constraint', domain=[j, k])
inventory_salvage_constraint[j, k] = y[j, k] >= 0

production_cost = Sum(i, c[i] * z[i, k])
salvage_value = Sum(j, s[j] * y[j, k])
objective_expression = production_cost - salvage_value
objective_func = Sum(j, b[j] * x[j]) + Sum(k, objective_expression * prob[k])
# GAMS Model
problem1 = Model(container=problem1, name='problem_1',
    equations=problem1.getEquations(), problem='MIP', sense=Sense.MIN,
    objective=objective_func)

# Solve Model
problem1.solve()

# Print Results
print("Objective Value:", problem1.objective_value)
print("The number of units produced:")
print(z.records.set_index(["i", "k"]))
```

- Với m = 5 và n = 8 (theo yêu cầu đề bài)

```
Enter the number of sites (m): 5
Enter the number of warehouse (n): 8
Objective Value: -2786.166666666667
The number of units produced:
                          level    marginal  lower  upper  scale
i           k
Warehouse 1 Scenario 1  12.000000    0.000000    0.0    inf    1.0
            Scenario 2  11.000000    0.000000    0.0    inf    1.0
Warehouse 2 Scenario 1   0.000000   66.666667    0.0    inf    1.0
            Scenario 2   0.000000  147.333333    0.0    inf    1.0
Warehouse 3 Scenario 1   0.000000    3.833333    0.0    inf    1.0
            Scenario 2   0.000000  217.166667    0.0    inf    1.0
Warehouse 4 Scenario 1   1.000000    0.000000    0.0    inf    1.0
            Scenario 2  10.000000    0.000000    0.0    inf    1.0
Warehouse 5 Scenario 1   0.000000   82.333333    0.0    inf    1.0
            Scenario 2   0.000000  106.666667    0.0    inf    1.0
Warehouse 6 Scenario 1   0.000000   80.000000    0.0    inf    1.0
            Scenario 2   0.000000  265.000000    0.0    inf    1.0
Warehouse 7 Scenario 1   7.000000    0.000000    0.0    inf    1.0
            Scenario 2   3.222222    0.000000    0.0    inf    1.0
Warehouse 8 Scenario 1   0.000000   70.666667    0.0    inf    1.0
            Scenario 2   0.000000  226.333333    0.0    inf    1.0
```

- Với m = 4 và n = 6 (một ví dụ khác)

```
Enter the number of sites (m): 4
Enter the number of warehouse (n): 6
Objective Value: -9139.555555555555
The number of units produced:
                          level  marginal  lower  upper  scale
i           k
Warehouse 1 Scenario 1  9.000000       0.0    0.0    inf    1.0
            Scenario 2  6.000000       0.0    0.0    inf    1.0
Warehouse 2 Scenario 1 14.000000       0.0    0.0    inf    1.0
            Scenario 2 14.000000       0.0    0.0    inf    1.0
Warehouse 3 Scenario 1 12.000000       0.0    0.0    inf    1.0
            Scenario 2  4.000000       0.0    0.0    inf    1.0
Warehouse 4 Scenario 1  8.000000       0.0    0.0    inf    1.0
            Scenario 2 10.000000       0.0    0.0    inf    1.0
Warehouse 5 Scenario 1  8.000000       0.0    0.0    inf    1.0
            Scenario 2  8.000000       0.0    0.0    inf    1.0
Warehouse 6 Scenario 1  0.000000      54.0    0.0    inf    1.0
            Scenario 2  6.222222       0.0    0.0    inf    1.0
```

3 Stochastic Linear Program for Evacuation Planning In Disaster Responses (SLP-EPDR)

3.1 Giới thiệu vấn đề

Các thảm họa tự nhiên (động đất, bão, lũ lụt,...) và phi tự nhiên (chiến tranh, xung đột chính trị, khủng bố,...) thường xảy ra mà không có dấu hiệu báo trước và gây ra nhiều thiệt hại, thương vong cho con người. Vì vậy, đòi hỏi phải có các kế hoạch ứng phó khẩn cấp để giảm thiểu thiệt hại khi xảy ra các sự kiện này. Mục tiêu chính của các kế hoạch ứng phó khẩn cấp là cung cấp nơi trú ẩn và cứu trợ những người bị ảnh hưởng trong vùng thảm họa sớm nhất có thể. Để đạt được mục tiêu này, một vài nghiên cứu về kế hoạch ứng phó khẩn cấp thường tập trung vào việc tìm cách để người dân ở khu vực an toàn có thể được hỗ trợ các vật phẩm cứu trợ cần thiết và cách xác định vị trí của các trung tâm nguồn lực. Bên cạnh đó, một vài nghiên cứu khác tập trung vào việc tìm cách đưa ra kế hoạch rõ ràng để sơ tán những người bị ảnh hưởng từ khu vực nguy hiểm đến khu vực an toàn. Đây cũng chính là vấn đề mà chúng ta sẽ nghiên cứu trong phần này bằng cách xây dựng một khung mô hình chung khi xảy ra thảm họa để giải quyết bài toán lập kế hoạch sơ tán những người bị ảnh hưởng.

Nhóm sẽ cố gắng xây dựng một mô hình dựa trên bài báo **A two-stage stochastic programming framework for evacuation planning in disaster responses (2020), Li Wang** trong đó có các trường hợp thể hiện những yếu tố ngẫu nhiên để thể hiện mức độ tiềm ẩn của thảm họa(cụ thể, một phần đường giao thông có thể bị phá hủy gây ra thời gian di chuyển và tải trên mỗi con đường là ngẫu nhiên).

3.2 Xây dựng mô hình

3.2.1 Bảng kí hiệu

Ký hiệu	Định nghĩa
V	Tập hợp các đỉnh
A	Tập hợp các đường đi
i, j	Chỉ số của các đỉnh, $i, j \in V$
(i, j)	Chỉ số của các đường đi có hướng, $(i, j) \in A$
s	Chỉ số của các kịch bản
S	Tổng số lượng của các kịch bản
v	Giá trị của cung cấp của đỉnh nguồn
$\widetilde{T}$	Ngưỡng thời gian khi nắm bắt được thông tin về thiên tai
T	Tổng số khoảng thời gian dùng để sơ tán
$u_{ij}(t)$	Dung lượng đường đi trên (i, j)
$u^s_{ij}(t)$	Dung lượng đường đi trên (i, j) trong kịch bản s tại thời điểm t
c^s_{ij}	Thời gian di chuyển trên (i, j) trong kịch bản s tại thời điểm t
μ_s	Xác suất xảy ra kịch bản s

Bảng 1: Chỉ số và tham số dùng trong xây dựng mô hình

Biến quyết định	Định nghĩa
x_{ij}	Luồng đường đi trên (i, j)
$y_{ij}^s(t)$	Luồng đường đi trên (i, j), trong kịch bản s tại thời điểm t

Bảng 2: Biến quyết định dùng trong xây dựng mô hình

3.2.2 Biến quyết định

Có 2 loại biến quyết định dùng để xây dựng mô hình sơ tán 2 giai đoạn. Ở giai đoạn 1, biến x_{ij} dùng để biểu thị luồng đường đi trên (i, j). Ở giai đoạn 2, biến $y_{ij}^s(t)$ dùng để biểu thị luồng đường đi trên (i, j) trong kịch bản s tại thời điểm t.

Ràng buộc hệ thống

Giai đoạn đầu tiên (The first stage)

Ở giai đoạn này, một kế hoạch sơ tán khả thi nên được xác định từ super-source đến super-sink. Luồng (flow) trên mỗi đường đi (link) phải đáp ứng cân bằng luồng như sau.

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i \tag{1}$$

Trong đó d_i là một tham số với các định nghĩa như sau:

$$d_i = \begin{cases} v, & \text{nếu } i = s \text{ (source)} \\ -v, & \text{nếu } i = t \text{ (sink, target)} \\ 0, & \text{trường hợp khác} \end{cases}$$

Nghĩa là nếu nó đang ở đỉnh nguồn (source) thì chỉ có luồng ra dẫn tới hiệu giữa luồng ra và luồng vào bằng v. Ngược lại, nếu đang ở đỉnh đích (sink) thì chỉ có luồng vào dẫn tới giá trị của hiệu này bằng $-v$.

Đồng thời luồng trên mỗi đường đi cũng phải thỏa mãn ràng buộc về mặt dung lượng:

$$0 \le x_{ij} \le u_{ij}, \quad \forall (i, j) \in A \tag{2}$$

Nghĩa là giá trị luồng (flow) đường đi trên (i, j) không được vượt quá dung lượng (capacity) đường đi trên (i, j)

Ngoài ra, ràng buộc về cân bằng luồng có thể tạo ra một số đường đi có vòng lặp, hoặc chu trình con nếu nó tiềm năng trong mạng lưới sơ tán. Để loại bỏ các vòng lặp trên đường đi thực tế của quá trình sơ tán, một biến thể hiện giá trị phạt $p_{ij}, (i, j) \in A$ sẽ được dùng để tính toán tổng giá trị phạt.

$$\begin{aligned} f(\mathbf{X}) &= \sum_{(i,j)\in A} p_{ij} x_{ij} \\ \mathbf{X} &:= \{x_{ij}\}_{(i,j)\in A} \quad \text{(vector luồng)} \end{aligned} \tag{3}$$

Nghĩa là hàm $f(\mathbf{X})$ sẽ biểu thị tổng giá trị phạt trong đó p_{ij} là giá trị phạt cho mỗi lần tạo ra vòng lặp trên mỗi luồng đường đi x_{ij}.

Giai đoạn thứ 2 (The second stage)

Giai đoạn 2 là giai đoạn đánh giá lại giai đoạn đầu tiên để đạt được kế hoạch sơ tán tối ưu. Trong giai đoạn này, người bị ảnh hưởng từ động đất sẽ nhận được hướng dẫn để di chuyển trên những tuyến đường với thời gian sơ tán là nhanh nhất dựa trên những thông tin thiên tai tại thời điểm đó. Trước ngưỡng thời gian nắm bắt được thông tin về thiên tai ($\widetilde{T}$) mọi người sẽ được sơ tán như kế hoạch sơ tán đã được đề ra trong giai đoạn 1. Từ đó một cặp ràng buộc về kế hoạch sơ tán cho mỗi kịch bản trước ngưỡng thời gian ($\widetilde{T}$) có thể được xây dựng như sau:

$$y_{ij}^s(t) = x_{ij}, \quad \forall t \leq \widetilde{T}, \quad (i,j) \in A, \quad s = 1, 2, ...S. \tag{4}$$

Cặp ràng buộc thể hiện mỗi quan hệ giữa tuyến đường đi với thời gian trong kế hoạch sơ tán, nghĩa là, nếu luồng lưu thông trên đường đi (i,j) trong giai đoạn 1 ví dụ $x_{ij} = 2$ thì luồng lưu thông trên trên mỗi đường đi (i,j) trước ngưỡng thời gian ($\widetilde{T}$) trong mỗi kịch bản cũng phải bằng 2 cụ thể $y_{ij}^s(t) = 2$. Tóm lại trước ngưỡng thời gian ($\widetilde{T}$) kế hoạch sơ tán cho giai đoạn thứ 2 giống với kế hoạch sơ tán ở giai đoạn đầu.

Mô hình 2 giai đoạn (a second-stage model) được xây dựng với mục tiêu là tối ưu tổng thời gian để sơ tán người chịu ảnh hưởng bởi thiên tai từ khu vực nguy hiểm đến các khu vực an toàn trong mọi kịch bản. Hàm thể hiện chi phí (tổng thời gian sơ tán) của mọi kịch bản được xây dựng như sau:

$$Q(Y, s) = \min \sum_{(i,j) \in A_s} c_{ij}^s(t) \cdot y_{ij}^s(t) \tag{5}$$

Với điều kiện:
Ràng buộc về cân bằng luồng:

$$\sum_{(i_t, j_t') \in A_s} y_{ij}^s(t) - \sum_{(j_t', i_t) \in A_s} y_{ij}^s(t') = d_t^s(t), \quad \forall i \in V, \quad t \in \{0, 1, 2, ..., T\}, \quad s = 1, 2, ..., S \tag{6}$$

Ràng buộc về dung lượng tải của đường đi:

$$0 \leq y_{ij}^s(t) \leq u_{ij}^s(t), \quad \forall (i,j) \in A, \quad t \in \{0, 1, ..., T\}, \quad s = 1, 2, ..., S \tag{7}$$

Cặp ràng buộc đảm bảo kế hoạch sơ tán trước thời gian ngưỡng ($\widetilde{T}$) của mỗi kịch bản trong giai đoạn 2 giống giai đoạn 1:

$$y_{ij}^s(t) = x_{ij}, \quad \forall t \leq \widetilde{T}, \quad (i,j) \in A, \quad s = 1, 2, ...S. \tag{8}$$

3.3 Mô hình sơ tán ngẫu nhiên 2 giai đoạn

Mục đích chính của thành lập mô hình sơ tán ngẫu nhiên 2 giai đoạn thực chất là xây dựng một kế hoạch sơ tán mạnh mẽ ở giai đoạn thứ nhất dựa trên đánh giá những đường đi tương ứng với mỗi kịch bản trong giai đoạn 2. Để làm được điều đó ta giả sử xác suất xuất hiện của mỗi kịch bản s là μ_s, $s = 1, 2, 3, ...S$. Để tối ưu hóa tổng giá trị phạt ($f(\mathbf{X})$) trong giai đoạn 1 và thời gian sơ tán tổng thể dự kiến ($Q(Y, s)$) trong giai đoạn 2, một mô hình sơ tán 2 giai đoạn

phụ thuộc vào thời gian và các kịch bản ngẫu nhiên được xây dựng như sau:

$$\begin{cases} \min \sum_{(i,j)\in A} p_{ij} \cdot x_{ij} + \sum_{s=1}^{S} \mu_s \cdot Q(Y,s) \\ \text{s.t.} \\ \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i, \quad \forall i \in V \\ 0 \le x_{ij} \le u_{ij}, \quad \forall (i,j) \in A \\ \text{trong đó,} \\ Q(Y,s) = \min \sum_{(i,j)\in A_s} c_{ij}^s(t) \cdot y_{ij}^s(t) \\ \text{s.t.} \\ \sum_{(i_t,j_t')\in A_s} y_{ij}^s(t) - \sum_{(j_t',i_t)\in A_s} y_{ij}^s(t') = d_t^s(t), \quad \forall i \in V, \\ t \in \{0,1,2,...,T\}, \quad s = 1,2,...,S \\ 0 \le y_{ij}^s(t) \le u_{ij}^s(t), \quad \forall (i,j) \in A, \quad t \in \{0,1,...,T\}, \quad s = 1,2,...,S \\ y_{ij}^s(t) = x_{ij}, \quad \forall t \le \widetilde{T}, \quad (i,j) \in A, \quad s = 1,2,...S. \end{cases} \tag{9}$$

Trong đó:

- $\sum_{(i,j)\in A} p_{ij} \cdot x_{ij} + \sum_{s=1}^{S} \mu_s \cdot Q(Y,s)$ là hàm mục tiêu cần phải tối thiểu hóa
- $\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i, \quad \forall i \in V$ là ràng buộc cân bằng luồng ở giai đoạn đầu tiên.
- $0 \le x_{ij} \le u_{ij}, \quad \forall (i,j) \in A$ là ràng buộc về dung lượng tải của đường đi ở giai đoạn đầu.
- $Q(Y,s)$ đại diện cho tổng thời gian sơ tán người chịu ảnh hưởng từ khu vực bị ảnh hưởng bởi thiên tai đến khu vực an toàn.
- p_{ij} là chi phí phạt.
- $\sum_{(i_t,j_t')\in A_s} y_{ij}^s(t) - \sum_{(j_t',i_t)\in A_s} y_{ij}^s(t') = d_t^s(t), \quad \forall i \in V$, là ràng buộc về cân bằng luồng ở giai đoạn thứ 2 của kịch bản s tại thời điểm t.
- $0 \le y_{ij}^s(t) \le u_{ij}^s(t), \quad \forall (i,j) \in A, \quad t \in \{0,1,...,T\}, \quad s = 1,2,...,S$ là ràng buộc về dung lượng tải của đường đi ở giai đoạn thứ 2 của kịch bản s tại thời điểm t.
- $y_{ij}^s(t) = x_{ij}, \quad \forall t \le \widetilde{T}, \quad (i,j) \in A, \quad s = 1,2,...S.$ là cặp ràng buộc để kế hoạch sơ tán của mỗi kịch bản trong giai đoạn 2 trước ngưỡng thời gian $(\widetilde{T})$ giống với giai đoạn 1.

Mô hình trên là mô hình sơ tán ngẫu nhiên 2 giai đoạn phụ thuộc vào thời gian. Bởi vì, ở giai đoạn 2 số lượng kịch bản xảy ra có giới hạn nên mô hình trên có thể tương đương với một mô hình sơ tán một giai đoạn sau:

$$\begin{cases} \min \ \sum_{(i,j)\in A} p_{ij} \cdot x_{ij} + \sum_{s=1}^{S} (\mu_s \cdot \sum_{(i,j)\in A_s} c_{ij}^s(t) \cdot y_{ij}^s(t)) \\ \text{s.t.} \\ \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i, \quad \forall i \in V \\ 0 \le x_{ij} \le u_{ij}, \quad \forall (i,j) \in A \\ \sum_{(i_t,j_t')\in A_s} y_{ij}^s(t) - \sum_{(j_t',i_t)\in A_s} y_{ij}^s(t') = d_i^s(t), \quad \forall i \in V, \\ t \in \{0,1,2,...,T\}, \quad s = 1,2,...S \\ 0 \le y_{ij}^s(t) \le u_{ij}^s(t), \quad \forall (i,j) \in A, \quad t \in \{0,1,..,T\}, \quad s = 1,2,...,S \\ y_{ij}^s(t) = x_{ij}, \quad \forall t \le \widetilde{T}, \quad (i,j) \in A, \quad s = 1,2,...S. \end{cases} \tag{10}$$

3.4 Giải thuật

Mô hình (10) về cơ bản là một mô hình quy hoạch nguyên chứa hai loại biến quyết định là $\mathbf{X}: = \{x_{ij}\}_{(i,j)\in A}$ và $\mathbf{Y}: = \{y_{ij}^s(t)\}_{i,j\in A, t\in\{0,1,...,T\}, s=1,2,...,S}$. Trong mô hình này, ràng buộc ghép là một ràng buộc phức tạp làm cho mô hình không thể giải được trong thời gian đa thức. Do đó, phần này dự định nới lỏng ràng buộc ghép vào hàm mục tiêu bằng phương pháp nới lỏng Lagrange. Sau đây, phần 3.4.1 cung cấp cách tiếp cận kép dựa trên phương pháp nới lỏng Lagrange để phân rã bài toán ban đầu thành hai bài toán con dễ giải có giá trị mục tiêu là giới hạn cận dưới của giá trị tối ưu của mô hình (10). Cụ thể hơn, mô hình ban đầu sẽ được phân rã thành bài toán luồng với chi phí tối thiểu (min-cost flow) tiêu chuẩn và bài toán phụ thuộc vào thời gian. Đây đều là những bài toán có thể được giải quyết hiệu quả bằng những thuật toán chính xác.

3.4.1 Phân rã mô hình

Từ mô hình ban đầu, có thể thấy rằng ràng buộc (4) là một ràng buộc khó. Ràng buộc này đặc trưng cho mối quan hệ giữa việc lựa chọn một đường đi vật lý và các cung phụ thuộc vào thời gian dựa trên kịch bản tương ứng. Do đó, chúng ta sẽ sử dụng hệ số nhân Lagrange $\alpha_{ij}^s(t), (i,j) \in A, s = 1,2,...,S, t \leq \widetilde{T}$ cho ràng buộc ghép và sau đó ràng buộc này có thể được nới lỏng thành hàm mục tiêu dưới dạng sau:

$$\sum_{s=1}^{S}\sum_{t\leq\widetilde{T}}\sum_{(i,j)\in A}\alpha_{ij}^s(t)(y_{ij}^s(t)-x_{ij}) \tag{11}$$

Do đó, sau khi nới lỏng công thức (11) thành hàm mục tiêu, công thức nới lỏng của mô hình (10) có thể được xây dựng như sau:

$$\begin{cases} \min \sum_{(i,j)\in A} p_{ij}x_{ij} + \sum_{s=1}^{S}(\mu_s \cdot \sum_{(i,j)\in A_s} c_{ij}^s(t)\cdot y_{ij}^s(t)) + \\ \sum_{s=1}^{S}\sum_{t\leq\widetilde{T}}\sum_{(i,j)\in A}\alpha_{ij}^s(t)(y_{ij}^s(t)-x_{ij}) \\ \text{s.t.} \\ \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i, \quad \forall i \in V \\ 0 \leq x_{ij} \leq u_{ij}, \quad \forall (i,j) \in A \\ \sum_{(i_t,j_t')\in A_s} y_{ij}^s(t) - \sum_{(j_t',i_t)\in A_s} y_{ij}^s(t') = d_i^s(t), \quad \forall i \in V, \\ t \in \{0,1,...,T\}, \quad s = 1,2,...S \\ 0 \leq y_{ij}^s(t) \leq u_{ij}^s(t), \quad \forall (i,j) \in A, \quad t \in \{0,1,..,T\}, \quad s = 1,2,...,S \end{cases} \tag{12}$$

Cần lưu ý rằng các biến X và Y có thể tách biệt với nhau trong mô hình nới lỏng trên Nghĩa là bằng cách kết hợp các phần tương ứng, mô hình nới lỏng (12) được phân rã thành hai bài toán con như sau:

SubProblem 1: Bài toán luồng với chi phí tối thiểu

Bài toán con đầu tiên có thể được coi là bài toán luồng chi phí tối thiểu và dạng của nó được đưa ra như sau:

$$\begin{cases} \min \text{SP1}(\alpha) = \sum_{(i,j)\in A}\left(p_{ij} - \sum_{s=1}^{S}\sum_{t\leq\widetilde{T}}\alpha_{ij}^s(t)\right)x_{ij} \\ \text{s.t.} \\ \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i, \quad \forall i \in V \\ 0 \leq x_{ij} \leq u_{ij}, \quad \forall (i,j) \in A \end{cases} \tag{13}$$

Hàm mục tiêu của SubProblem 1 có thể được định nghĩa như sau: $g_{ij} := p_{ij} - \sum_{s=1}^{S} \sum_{t \leq \widetilde{T}} \alpha_{ij}^{s}(t)$ dùng để biểu diễn chi phí tổng quát của mỗi đường đi (link). Do đó, SubProblem 1 có thể giải được bằng thuật toán Successive shortest path sẽ được giới thiệu ở mục 3.5 trong phần sau. Để thuận tiện, ta ký hiệu giá trị mục tiêu tối ưu của bài toán con thứ nhất là $Z_{SP1}^{*}(\alpha)$. Thuật toán Successive shortest path cho SubProblem 1 sẽ được phát triển trong Giải thuật 1 bên dưới:

Giải thuật 1: Successive shortest path algorithm for min-cost flow problem.
Bước 1: Lấy biến x làm luồng khả thi giữa bất kỳ OD nào và nó có chi phí phân phối tối thiểu trong các luồng khả thi có cùng giá trị luồng.
Bước 2: Thuật toán sẽ kết thúc nếu giá trị luồng của x đạt v hoặc không có đường dẫn chi phí tối thiểu trong mạng phần dư (residual network) $(V, A(x), C(x), U(x), D)$; nếu không thì đường đi ngắn nhất với luồng tối đa sẽ được tính toán bằng thuật toán sửa nhãn (label-correcting), sau đó chuyển sang Bước 3. Các hàm $A(x), C(x), U(x)$ trong mạng phần dư có thể được định nghĩa như sau:

$$A(x) = \{(i,j) | (i,j) \in A, x_{ij} < u_{ij}\} \cup \{(j,i) | (j,i) \in A, x_{ij} > 0\}$$

$$C(x) = \begin{cases} c_{ij}, (i,j) \in A, x_{ij} < u_{ij} \\ -c_{ji}, (i,j) \in A, x_{ji} > 0 \end{cases}$$

$$U_{ij}(x) = \begin{cases} u_{ij}, (i,j) \in A, x_{ij} < u_{ij} \\ x_{ji}, (i,j) \in A, x_{ji} > 0 \end{cases}$$

Bước 3: Tăng luồng dọc theo đường đi có chi phí tối thiểu. Nếu giá trị luồng tăng lên không vượt quá v thì chuyển sang Bước 2.

SubProblem 2: Bài toán luồng với chi phí tối thiểu phụ thuộc vào thời gian
Bài toán con thứ hai của mô hình nới lỏng (12) liên quan đến biến quyết định Y với giá trị tối ưu của bài toán được ký hiệu là $Z_{SP2}^{*}(\alpha)$ được biểu diễn như sau:

$$\begin{cases} \min \text{SP2}(\alpha) = \sum_{s=1}^{S} \sum_{(i,j) \in A} \left(\sum_{t \in \{0,1,\ldots,T\}} \mu^{s} \cdot c_{ij}^{s}(t) + \sum_{t \leq \widetilde{T}} \alpha_{ij}^{s}(t) \right) y_{ij}^{s}(t) \\ \text{s.t.} \\ \sum_{(i_t, j_{t'}) \in A_s} y_{ij}^{s}(t) - \sum_{(j_{t'}, i_t) \in A_s} y_{ij}^{s}(t') = d_i^{s}(t), \quad \forall i \in V, \\ t \in \{0,1,\ldots,T\}, s = 1,2,\ldots,S \\ 0 \leq y_{ij}^{s}(t) \leq u_{ij}^{s}(t), \quad \forall (i,j) \in A, \quad t \in \{0,1,\ldots,T\}, \quad s = 1,2,\ldots S \end{cases} \tag{14}$$

SubProblem 2 có thể được phân tách thành tổng số S bài toán con, mỗi bài toán con trong số đó có thể được gọi là bài toán luồng với chi phí tối thiểu có thời gian và dung lượng di chuyển của đường đi phụ thuộc vào thời gian, cụ thể là:

$$\begin{cases} \min \text{SP2}(\alpha, s) = \sum_{(i,j) \in A} \left(\sum_{t \in \{0,1,\ldots,T\}} \mu^{s} \cdot c_{ij}^{s}(t) + \sum_{t \leq \widetilde{T}} \alpha_{ij}^{s}(t) \right) y_{ij}^{s}(t) \\ \text{s.t.} \\ \sum_{(i_t, j_{t'}) \in A_s} y_{ij}^{s}(t) - \sum_{(j_{t'}, i_t) \in A_s} y_{ij}^{s}(t') = d_i^{s}(t), \quad \forall i \in V, \quad t \in \{0,1,\ldots,T\} \\ 0 \leq y_{ij}^{s}(t) \leq u_{ij}^{s}(t), \quad \forall (i,j) \in A, \quad t \in \{0,1,\ldots,T\} \end{cases} \tag{15}$$

Với mỗi kịch bản $s \in \{1,2,\ldots,S\}$, bài toán con (15) có cấu trúc tương tự như bài toán con (13) với chi phí đường đi phụ thuộc vào thời gian $c_{ij}^{s}(t)$ và sức chứa đường đi $u_{ij}^{s}(t)$ và chi phí tổng quát $g_{ij}^{s}(t)$. Vì khoảng thời gian T được xem xét được chia thành hai giai đoạn thời gian

nên chi phí tổng quát được xác định là hàm từng phần:

$$g_{ij}^s(t) = \begin{cases} \mu_s \cdot c_{ij}^s(t) + \alpha_{ij}^s(t), t \leq \widetilde{T} \\ \mu_s \cdot c_{ij}^s(t), \qquad \widetilde{T} < t \leq T \end{cases}$$

Vì bài toán con (15) là bài toán luồng với chi phí tối thiểu phụ thuộc thời gian và do đó **Giải thuật 1** phải được sửa đổi ở Bước 2. Đầu tiên, các tham số $A(y(t)), C(y(t)), U(y(t))$ trong mạng phần dư $N(y(t))$ được định nghĩa như sau:

$A_s(y(t)) = \{(i_t, j_{t'})|(i_t, j_{t'}) \in A_s, y_{ij}^s < u_{ij}^s\} \cup \{(j_{t'}, i_t)|(j_{t'}, i_t) \in A_s, y_{ij}^s > 0\}, s = 1, 2, ...S$

$$c_{ij}^s(y(t)) = \begin{cases} c_{ij}^s(t), (i_t, j_{t'}) \in A_s, y_{ij}^s(t) < u_{ij}^s(t), \quad t \in \{1, 2, ..., T\} \\ -c_{ji}^s(t'), (j_{t'}, i_t) \in A_s, \quad \forall\{t' \in \{0, 1, ..., T\}|y_{ji}^s(t') > 0\}, \quad s = 1, 2, ..., S \\ T, \quad (j_{t'}, i_t) \in A_s, \quad \forall\{t' \in \{0, 1, ..., T\}|y_{ji}^s(t') = 0\} \end{cases}$$

Sau đó, thuật toán sửa nhãn đã được sửa đổi sẽ được áp dụng để tìm đường đi có chi phí tối thiểu phụ thuộc vào thời gian trong mạng phần dư.

Bằng cách giải bài toán con (13) và (14) với nghiệm nới lỏng X và Y, giá trị mục tiêu tối ưu Z_{LR}^* cho mô hình nới lỏng (12) với tập vector nhân Lagrange α đã cho có thể được biểu diễn như sau:

$$Z_{LR}^*(\alpha) = Z_{SP1}^*(\alpha) + Z_{SP2}^*(\alpha) \tag{16}$$

Có thể thấy giá trị mục tiêu tối ưu của mô hình nới lỏng (12) chính là giới hạn cận dưới của giá trị mục tiêu tối ưu của mô hình (10) ban đầu. Để có được giải pháp chất lượng cao, cần đạt được giới hạn cận dưới gần với giá trị mục tiêu tối ưu của mô hình ban đầu. Nghĩa là, phải đạt được giới hạn cận dưới lớn nhất có thể và biểu thức được đưa ra như sau:

$$Z_{LD}(\alpha^*) = max_{\alpha \geq 0} Z_{LR}(\alpha) \tag{17}$$

3.5 Successive shortest path algorithm

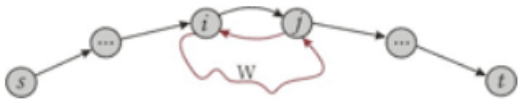
Thuật toán Successive shortest path là thuật toán để tìm maximum flow và tối ưu objective function một cách đồng thời. Vì vậy nó có thể giải quyết được vấn đề được gọi là max-flow-min-cost theo ý tưởng dưới đây:

Giả sử chúng ta có một mạng lưới giao thông G và yêu cầu là phải tìm một đường đi tối ưu trên nó. Ta sẽ thực hiện bằng cách chuyển đổi mạng lưới đó sang một mạng lưới khác tương đương tham khảo tại Finding a Solution bằng cách thêm 2 đỉnh s và t cùng với một số cạnh như sau. Với mỗi node i trong V nếu $d_i > 0$, ta thêm một cung (s, i) với dung lượng là d_i và cost 0. Với mỗi node i trong V nếu $d_i < 0$, ta thêm một cung (i, t) với dung lượng là $-d_i$ và cost 0.

Thuật toán này thay vì tìm maximum flow như thường lệ, ta sẽ gửi một luồng đi từ s đến t dọc theo đường đi ngắn nhất (tương ứng với chi phí cung). Tiếp theo, ta sẽ cập nhật mạng dư (residual network), tìm một đường đi ngắn nhất khác và tăng cường luồng lại, và tiếp tục lặp lai như vậy. Thuật toán sẽ dừng lại khi mạng dư không chứa đường đi từ s đến t nữa (luồng tối đa). Vì luồng là tối đa nên nó sẽ tương ứng với một giải pháp khả thi của vấn đề min-cost-flow ban đầu. Và hơn thế nữa, nó cũng sẽ là tối ưu (nhóm sẽ giải thích ở dưới sau).

Thuật toán successive shortest path được sử dụng khi mạng lưới G không bao gồm chu kì có chi phí âm. Nếu không, ta không thể nói chính xác rằng "the shortest path" nghĩa là gì.Bây giờ chúng ta sẽ làm rõ cách tiếp cận của thuật toán. Khi luồng hiện tại có giá trị là 0, mạng lưới giao thông G sẽ không có chu kì chi phí âm. Giả sử rằng sau một vài bước tăng cường ta có một

luồng x và G_x vẫn không chứa chu kì chi phí âm. Nếu x là luồng tối đa thì nó sẽ là luồng tối ưu, theo theorem 2. Nếu không, hãy kí hiệu đường đi ngắn nhất được tìm kiếm thấy tiếp trong theo mạng lưới G_x là P.



Hình 1: *Chu kì chi phí âm xuất hiện trong mạng dư*

Giả sử sau một vài bước tăng cường luồng hiện tại x dọc theo đường P, một chu kì chi phí âm W xuất hiện trong mạng dư. Lưu ý rằng, trước khi tăng cường không có chu kì chi phí âm nào. Điều này có nghĩa là có một cạnh (i, j) trong P (hoặc con đường $(i, ..., j)$ trong P) mà việc đảo ngược của nó (j, i) đã tạo ra chu trình W sau khi tăng. Rõ ràng, chúng ta có thể chọn một đường dẫn khác từ s đến t, đi từ s đến i sau đó từ i đến j dọc theo các cạnh của W sau đó từ j đến t. Hơn nữa, chi phí của đường dẫn này ít hơn chi phí của P. Chúng ta có một mâu thuẫn với giả định rằng P là đường dẫn ngắn nhất.

Một phân tích đơn giản cho thấy thuật toán thực hiện tối đa O(nB) lần tăng cường, ở đó B được gán là cận trên lớn nhất cho nguồn cung cấp của bất kì nút nào. Thật sự, mỗi lần tăng cường sẽ giảm mạnh mẽ dung lượng dư của một cung nguồn (được xem là bằng với nguồn cung cấp của nút tương ưng). Nhờ vào integrality property nó sẽ giảm đi ít nhất một đơn vị. Và bằng cách sử dụng một thuật toán có độ phức tạp là O(nm) để tìm đường đi ngắn nhất (có thể có cạnh âm), chúng ta sẽ đạt được độ phức tạp là $O(n^2mB)$ cho thuật toán successive shortest path.

Successive shortest path algorithm

1. Chuyển đổi mạng lưới giao thông G bằng cách thêm source node(s) và sink node(t).
2. Khởi tạo luồng x giá trị ban đầu là 0.
3. while (G_x còn bao gồm đường từ s đến t) do
   (a) Tìm bất kì đường đi ngắn nhất P nào từ s đến t.
   (b) Tăng cường luồng hiện tại x dọc theo P.
   (c) Cập nhật G_x.

Để xử lí các tình huống có tồn tại cạnh có chi phí âm, ta có thể làm nó trở nên không âm bằng cách sử dụng thuật toán Bellman-Ford's cùng với đó là sự xuất hiện của thuật ngữ tiềm năng của một nút (potential node). Bởi vì thực hiện thuật toán với chi phí dư không làm thay đổi đường đi ngắn nhất (theo định lí 2), và chúng ta có thể thực hiện thuật toán với mạng lưới đã được biến đổi và sử dụng thuật toán Dijkstra để tìm succesive shortest path hiệu quả hơn. Tuy nhiên, nó đòi hỏi cần phải giữ chi phí của mỗi cạnh là không âm ở mỗi vòng lặp, vì vậy ta cần phải cập nhật tiềm năng của nút và giảm chi phí ngay sau khi đường đi ngắn nhất được tìm thấy. Điều đó được thực hiện như đặc tả dưới đây:
Reduce Cost (π)

1. Với mỗi (i, j) trong E_x thực hiện

2. $c_{ij} \leftarrow c_{ij} + \pi_i - \pi_j$
3. $c_{rev(i,j)} \leftarrow 0$

Sau khi tìm thấy đường đi ngắn nhất liên tiếp, chúng ta cần cập nhật tiềm năng của các nút. Đối với mỗi nút i trong V, tiềm năng của nó bằng độ dài của đường đi ngắn nhất từ s đến t. Sau khi giảm chi phí của mỗi cung, chúng ta sẽ thấy rằng dọc theo đường đi ngắn nhất từ s đến i, các cung sẽ có chi phí bằng không trong khi các cung nằm ngoài đường đi ngắn nhất đến bất kỳ đỉnh nào khác sẽ có chi phí dương. Đó là lý do tại sao chúng ta gán chi phí bằng không cho bất kỳ cung đảo nào $c_{rev(i,j)}$.

Chúng ta kí hiệu chí phí cung đảo là $c_{rev(i,j)}$ thay vì c_{ji} là bởi vì mạng lưới có thể chứa cả 2 cung (i, j) và (j, i) (giả định 2). Đối với các cung khác (nằm ngoài đường đi tăng cường) việc gán bắt buộc này không làm gì cả, bởi vì các cung đảo của nó sẽ không xuất hiện trong mạng lưới dư.

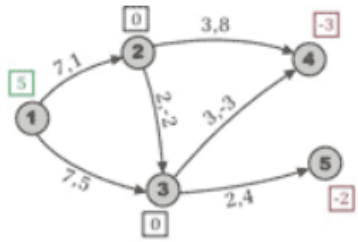
Successive shortest path algorithm with potential

1. Chuyển đổi mạng lưới giao thông G bằng cách thêm source node(s) và sink node(t).
2. Khởi tạo luồng x giá trị ban đầu là 0.
3. Sử dụng thuật toán Bellman-Ford để thiết lập tiềm năng π
4. Reduce Cost (π)
5. while (G_x còn bao gồm đường từ s đến t) do
   (a) Tìm bất kì đường đi ngắn nhất P nào từ s đến t.
   (b) Reduce Cost (π)
   (c) Tăng cường luồng hiện tại x dọc theo P.
   (d) Cập nhật G_x.

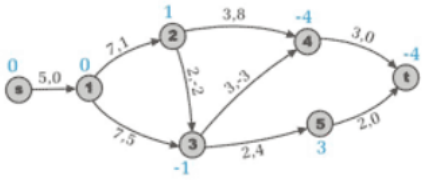
Phân tích độ phức tạp

Chúng ta chỉ sử dụng thuật toán Bellman-Ford một lần để tránh chi phí âm trên các cạnh. Nó mất thời gian O(nm). Sau đó, chúng ta sử dụng thuật toán Dijkstra O(nB) lần, mất thời gian O(n^2) (trường hợp thực hiện đơn giản) hoặc O(mlogn) (trường hợp thực hiện heap cho mạng thưa). Tổng cộng, chúng ta nhận được thời gian làm việc ước tính $O(n^3B)$ cho trường hợp thực hiện đơn giản và O(nmBlogn) nếu sử dụng heap.

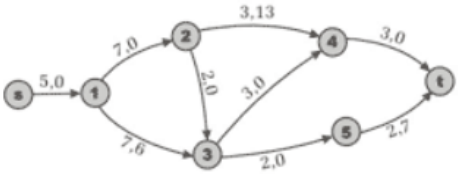
Ví dụ để làm rõ thuật toán:
Cho một mạng lưới được khởi tạo như hình sau:



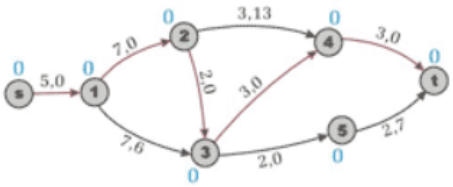
Thêm các node s, node t và thiết lập giá trị tiềm năng của node bằng thuật toán Bellman-Ford bằng cách khởi tạo giá trị tiềm năng source = 0, và các giá trị khác khởi tạo là ∞ và tiến hành cập nhật như sau với mỗi cung (i, j), nếu p(i) + cost < p(j) => cập nhật p(j) = p(i) + cost. Sau khi thực hiện ta được hình sau:



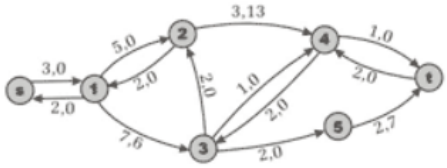
Tiếp theo, thực hiện giảm chi phí trên cạnh bằng cách cập nhật $c_{ij} \leftarrow c_{ij} + \pi_i - \pi_j$. Ta thu được hình sau:



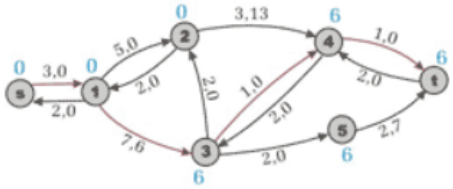
Sau đó, đường tăng cường đầu tiên được tìm thấy là s-1-2-3-4-t với dung lượng là 2 thông qua thuật toán Dijkstra's và giá trị tiềm năng của nút cũng được tính toán lại. Ta được hình sau:



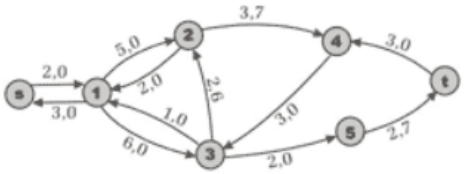
Thực hiện giảm chi phí trên cạnh ta thu được mạng lưới dư như sau:



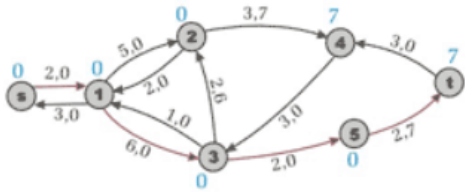
Tiếp theo, đường tăng cường tiếp theo được tìm thấy là s-1-3-4-t với dung lượng là 1, tiến hành tính toán tiềm năng cho các nút ta thu được hình sau:



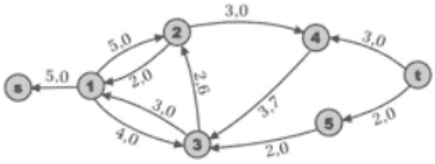
Thực hiện giảm chi phí trên cạnh ta thu được mạng lưới dư như sau:



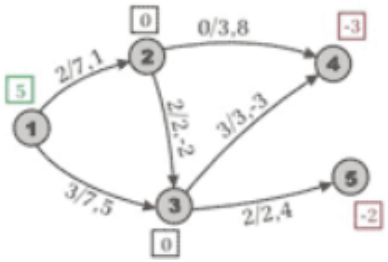
Đường tăng cường thứ 3 được tìm thấy là s-1-3-5-t và tính toán tiềm năng cho các nút mới ta thu được hình sau:



Lúc này mạng lưới dư mới sẽ còn đường tăng cường nữa :



Tái câu trúc lại mạng lưới giao thông ta sẽ thu được hình sau với luồng tối ưu với chi phí là 12.



References

[1] *MINIMUM COST FLOW PART TWO: ALGORITHMS* truy cập tại: https://www.topcoder.com/thrive/articles/Minimum%20Cost%20Flow%20Part%20Two:%20Algorithms

[2] *MINIMUM COST FLOW PART ONE: KEY CONCEPTS* truy cập tại: https://www.topcoder.com/thrive/articles/Minimum%20Cost%20Flow%20Part%20One:%20Key%20Concepts

[3] L. Wang, "A two-stage stochastic programming framework for evacuation planning in disaster responses," Computers & Industrial Engineering, vol. 145, p. 106458, 2020.

[4] *GAMS Stochastic Programming* truy cập tại: https://www.gams.com/latest/docs/UG_EMP_SP.html