ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

MÔ HÌNH HÓA TOÁN HỌC (CO2011)

Báo cáo Bài tập lớn

Stochastic Programming and Applications

GVHD: Th.S Mai Xuân Toàn
SV thực hiện: Phạm Ngọc Long - 2211894.
Phạm Văn Quốc Việt - 2213950.
Trần Thanh Phúc - 2212656.
Nguyễn Khắc Huy - 2211219.
Võ Phương Minh Nhật - 2212413.

TP. Hồ Chí Minh, Tháng 12/2023

Contents

1 Danh sách thành viên **2**

2 Cơ sở lý thuyết **2**
2.1 Giới thiệu về Stochastic Programming và Optimization . . . 2
2.1.1 Các thuật ngữ cơ bản . . . 2
2.1.2 Stochastic linear programming (SLP) . . . 2
2.2 One-Stage Stochastic linear programming - No recourse (1-SLP) . . . 3
2.3 Generic Stochastic Programming (GSP) with RECOURSE . . . 4
2.4 Two-Stage Stochastic linear programming (2-SLP) . . . 4
2.4.1 Two-stage SLP Recourse model (dạng đơn giản) . . . 5
2.4.2 Two-stage SLP Recourse model - (dạng chuẩn) . . . 5

3 Vấn đề 1 [Công nghiệp - Sản xuất] **6**
3.1 Tóm tắt bài toán . . . 6
3.2 Mô hình hóa bài toán . . . 6
3.3 Bài toán cụ thể . . . 7

4 Ứng dụng Stochastic Linear Program cho việc lập kế hoạch sơ tán khi ứng phó thiên tai **11**
4.1 Bối cảnh . . . 11
4.2 Phát biểu bài toán . . . 12
4.2.1 Bài toán sơ tán dưới dạng two-stage stochastic programming . . . 12
4.2.2 Mô tả bài toán min-cost flow trong tối ưu việc sơ tán . . . 13
4.3 Công thức, ký hiệu mô hình . . . 13
4.3.1 Ký hiệu . . . 13
4.3.2 Biến quyết định . . . 14
4.3.3 Các ràng buộc của bài toán . . . 14
4.3.4 Mô hình hóa bài toán sơ tán với two-stage stochastic programming . . . 16
4.4 Giải thuật . . . 17
4.4.1 Lagrangian relaxation . . . 17
4.4.2 Phân tách mô hình bài toán . . . 18
4.4.2.a Bài toán phụ 1: Bài toán min-cost flow . . . 19
4.4.2.b Bài toán phụ 2: Bài toán min-cost flow phụ thuộc thời gian 20
4.4.3 Xác định cận trên . . . 22
4.4.4 Giải thuật giải bài toán 2-SLP trong sơ tán người dân áp dụng Lagrangian relaxation và subgradient optimization. . . . 22
4.5 Hiện thực mô hình bài toán bằng mô phỏng trên máy tính. . . . 23

5 Tiến độ hoàn thành **28**

1 Danh sách thành viên

STT	Họ và tên	MSSV	Nội dung	Đóng góp
1	Phạm Ngọc Long	2211894	- Problem 2: Cơ sở lý thuyết & Coding	100%
2	Phạm Văn Quốc Việt	2213950	- Problem 2: Mô tả & phân tích bài toán - Nghiên cứu Python & GAMSPy	100%
3	Trần Thanh Phúc	2212656	- Problem 1: Cơ sở lý thuyết - Viết báo cáo	100%
4	Nguyễn Khắc Huy	2211219	- Problem 1: Phân tích bài toán - Problem 2: Cơ sở lý thuyết	100%
5	Võ Phương Minh Nhật	2212413	- Problem 1: Cơ sở lý thuyết & Coding	100%

2 Cơ sở lý thuyết

2.1 Giới thiệu về Stochastic Programming và Optimization

2.1.1 Các thuật ngữ cơ bản

- Mathematical Optimization là về việc đưa ra các quyết định nhằm tìm ra giá trị tối ưu nhất, chủ yếu ứng dụng các phương pháp toán học.

- Stochastic Programming (SP) là việc đưa ra các quyết định dưới những điều kiện chưa biết trước. Có thể coi nó là Mathematical Optimization với tham số ngẫu nhiên.

- Stochastic linear programs là các bài toán quy hoạch tuyến tính (bài toán có hàm mục tiêu là hàm tuyến tính) mà một số dữ liệu có thể chưa được biết trước.

- Recourse programs là các bài toán với một số quyết định hay cập nhật được đưa ra khi đã biết hết tất cả các dữ liệu.

2.1.2 Stochastic linear programming (SLP)

- Định nghĩa 1: Stochastic linear programming (SLP) là bài toán

$$\text{Minimize } Z = g(x) = f(x) = c^T.x, \quad \text{điều kiện } Ax = b \text{ và } Tx \geq h$$

với $x = (x_1, x_2, .., x_n)$ (các biến quyết định),
Ma trận A và vectơ b cho trước (với các ràng buộc xác định),
và các tham số ngẫu nhiên T, h với $Tx \geq h$ (với các ràng không xác định)

- Ví dụ 1: Ta xét bài toán tối ưu sau

$$\text{Minimize } z = x_1 + x_2 \quad \text{điều kiện } x_1 \geq 0,\ x_2 \geq 0$$
$$\begin{cases} \omega_1 x_1 + x_2 \geq 7; \\ \omega_2 x_1 + x_2 \geq 4; \end{cases}$$

Và ω_1, ω_2 là các biến ngẫu nhiên thỏa mãn phân phối đều Uniform(a,b) với $\omega_1 \sim$ Uniform(1,4), $\omega_2 \sim$ Uniform(1/3,1)

- Khi $\omega_1 = \omega_2 = 1$ thì 2 điều kiện trở thành $x_1 + x_2 = 4$ (đường màu đỏ) và $x_1 + x_2 = 7$ (đường màu xanh) và ta dễ dàng giải được bài toán. Liệu ta có thể giải bài toán với $\omega_1 = \omega_2 = 1$ là các biến ngẫu nhiên ?

- Phương pháp **wait-and-see**: Giả sử có thể tìm được các biến quyết định $x = [x_1, x_2]$ sau khi biết được giá trị các biến ngẫu nhiên $\omega = [\omega_1, \omega_2]$. Liệu ta có thể giải bài toán nếu không có waiting.

- Ta có thể giải bài toán mà không có waiting, tức là tìm $x = [x_1, x_2]$ trước khi biết $\omega = [\omega_1, \omega_2]$. Có 2 cách là (a) Guess at uncertainty và (b) Probabilistic Constraints

- **(a) Guess at uncertainty**: Ta đoán các giá trị hợp lí cho ω. Có 3 cách sau, mỗi cách cho ta thấy mỗi mức độ rủi ro:

- Unbiased: Chọn giá trị trung bình của ω
- Pessimistic: Chọn trường hợp xấu nhất của ω
- Optimistic: Chọn trường hợp tốt nhất của ω

Ví dụ, với **Unbiased**, phân phối đều Uniform(a,b) có mean value là (a+b)/2, nên $\hat{\omega} = (\frac{5}{2}, \frac{3}{2})$. Bài toán trở thành:

$$\text{Minimize } z = x_1 + x_2$$

có giá trị tối ưu $z_1 = \frac{50}{11}$ tại điểm $\hat{x} = (\hat{x_1}, \hat{x_2}) = 18/11, 32/11$ nếu thỏa mãn: $\begin{cases} \frac{5}{2}x_1 + x_2 \geq 7; & \frac{2}{3}x_1 + x_2 \geq 4; \\ x_1 \geq 0, & x_2 \geq 0 \end{cases}$

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

Định nghĩa 2 **(SLP with one-stage (No recourse) : 1-SLP)**. Xét bài toán $LP(\alpha)$ được tham số hóa bởi vectơ ngẫu nhiên α:

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

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

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

2. Giá trị (T, h) không biết trước khi xuất hiện mô hình, $h(\alpha)$ chỉ phụ thuộc vào các biến ngẫu nhiên α_j.
3. **Tính không chắc chắn** được biểu diễn thông qua phân phối xác suất của các tham số ngẫu nhiên $(\alpha_j) = \alpha$ nên bài toán LP tất định là trường hợp suy biến của SLP với α_j là hằng số

- Ta chỉ giải quyết bài toán với $x = (x_1, x_2, .., x_n) \in \chi$ là các biến quyết định cần phải xác định trước khi biết các vectơ tham số $\alpha \in \Omega$
- Ta thường đặt cận trên và cận dưới cho x qua miền $\chi = \{x \in \mathbb{R}^n : l \leq x \leq u\}$

Hướng tiếp cận: The Scenario Analysis - Giả sử rằng chỉ có hữu hạn ngữ cảnh cho bài toán. Với mỗi ngữ cảnh ta tìm nghiệm tối ưu, từ đó tìm nghiệm tối ưu tổng quát.

Ta dùng Scenario Analysis cho $T(\alpha)x \leq h(\alpha)$

Với mỗi ngữ cảnh $(T^s; h^s), s = 1, .., S$, ta giải bài toán

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

Ta sẽ tối thiểu hàm mục tiêu ứng với mỗi ngữ cảnh và xác suất của ngữ cảnh đó. Từ đó tìm nghiệm tổng quát từ các nghiệm ngữ cảnh $x^s(s = 1, .., S)$

Ưu điểm: mỗi ngữ cảnh là một bài toán LP
Nhược điểm: không dùng được cho các mô hình rời rạc (có thể là các mô hình không lồi nói chung)

2.3 Generic Stochastic Programming (GSP) with RECOURSE

Định nghĩa 3 **Stochastic program in two stages (generic 2-SP problem)**. Bài toán two-stage stochastic mở rộng từ định nghĩa 2 có dạng

$$2 - SP: \qquad \min_x g(x) = f(x) + \mathbf{E}_x[v(x, \omega)]$$

với $x = (x_1, x_2, .., x_n)$ là biến quyết định ở first stage. $f(x)$ có thể tuyến tính hoặc không và là một phần của hàm mục tiêu chính $g(x)$.

Giá trị trung bình $Q(x) := \mathbf{E}_\omega[v(x, \omega)]$ của hàm $v : \mathbb{R}^n \times \mathbb{R}^s \to \mathbb{R}$ phụ thuộc vào ngữ cảnh ω. $Q(x)$ là giá trị tối ưu của bài toán second-stage

$$\min_{\mathbf{y} \in \mathbb{R}^p} \mathbf{q} \ \mathbf{y} \ \mid \ subject \ to \quad T.x + W.y = h$$

Vectơ $\alpha = \alpha(\omega)$ and $y = y(\omega)$ được gọi là recourse decision variables, chỉ biết được sau experiment **E**.

Rõ ràng ta minimize tổng chi phí kì vọng $g(x) = f(x) + Q(x)$ với điều kiện $W.y(\omega) = h(\omega) - T(\omega).x$

Ở đây W được gọi là $m \times p$ recourse matrix, ta ta bắt đầu với trường hợp đơn giản $m = 1$, q là vectơ recourse đơn vị, cùng kích cỡ với y, và $y = y(\omega) \in \mathbb{R}^p$

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

Bây giờ ta xử lí bài toán two-stage SLP với recourse

2.4.1 Two-stage SLP Recourse model (dạng đơn giản)

Định nghĩa 4: Two-stage Stochastic linear program With Recourse (2-SLPWR) có dạng

$$2-SLP: \quad \min_{x\in X} c^T.x + \min_{y(\omega)\in Y} \mathbf{E}_\omega[\mathbf{q.y}]$$

hoặc tổng quát là

$$2-SLP: \quad \min_{x\in X, y(\omega)\in Y} \mathbf{E}_\omega[c^T.x + v(x,\omega)]$$

với $v(x,\omega) := \mathbf{q.\ y}$ và

. $Ax = b$ **First Stage Constraints**,

. $T(\omega).x + W.y = h(\omega)$ **Second Stage Constraints**

. or shortly $W.y = h(\omega) - T(\omega).x$

Bài toán SLP này là trường hợp cụ thể của bài toán 2-SP ở phần 3, hàm mục tiêu chính **(grand object function)** là $g(x)$ có

1. Hàm tất định $f(x)$ - **tuyến tính**

2. Hàm xác suất $v(x,\omega)$ ứng với các ngữ cảnh ω với $y = y(x,\omega) \in \mathbb{R}^p_+$ được gọi là biến **recourse** cho biến quyết định x và ngữ cảnh ω. **Recourse action** được biểu diễn thông qua **giá trị trung bình**. Cách tìm chính là sử dụng phương pháp **Scenarios analysis**. Phương pháp này dựa trên một vectơ ngẫu nhiên α với số lượng hữu hạn các ngữ cảnh.

Giá trị kì vọng $Q(x)$ hiển nhiên là một phân phối rời rạc của ω. Ta lấy $\Omega = \{\omega_k\}$ là một tập hữu hạn có S phần tử $\omega_1, .., \omega_S$ ứng với S ngữ cảnh, với xác suất p_k tương ứng. Vì $y = y(x,\omega)$nên giá trị kì vọng của $v(y) = v(x,\omega) := q.y$ (một giá trị q ứng với toàn bộ y_k) là

$$Q(x) = \mathbf{E}_\omega[v(x,\omega)] = \sum_{k=1}^{S} p_k\ q\ y_k = \sum_{k=1}^{S} p_k\ v(x,\omega_k)$$

với

- P_k là xác suất của ngữ cảnh ω_k, q là **single unit penalty cost**

- $q\ y_k = v(x,\omega_k)$ là **penalty cost** khi dùng y_k đơn vị ở giai đoạn **recourse**, phụ thuộc vào biến quyết định x ở **first-stage** và ngữ cảnh ω_k

2.4.2 Two-stage SLP Recourse model - (dạng chuẩn)

Định nghĩa 5. **The canonical 2 - stage stochastic linear program with Recourse** có dạng như sau:

$$2-SLP: \quad \min_x \ g(x) \quad with$$

$$g(x) := c^T.x + v(y)$$

. subject to $Ax = b$ where $x \in \mathbb{X} \subset \mathbb{R}^n, x \geq 0$

. $v(z) := \min_{y \in \mathbb{R}^p_+} \mathbf{q}\ \mathbf{y}$ subject to $W\,.\,y = h(\omega) - T(\omega)\,.\,x =: z$

với $v(y) := v(x, \omega)$ là hàm giá trị ở second-stage, và $y = y(x, \omega) \in \mathbb{R}^p_+$ là **recourse** cho biến quyết định x và ngữ cảnh ω. 1. Chi phí recourse kì vọng của biến quyết định x là $Q(x) := \mathbf{E}_\omega[v(x, \omega)]$. Vì vậy ta minimize chi phí kì vọng tổng là

. $min_{x \in \mathbb{R}^n, y \in \mathbb{R}^p_+}$ $c^T\,.\,x + Q(x)$

2. Ta dùng biến quyết định thứ hai $y(\omega)$ để ta có thể điều chỉnh theo một cách tối ưu ứng với các ràng buộc của bài toán. Ta gọi là **recourse action**

$$x \quad - - - - - - - - - - - - - \quad T, h, \omega \quad - - - - - - - - - - - - - > y$$

3. Giá trị tối ưu của 2nd-stage LP là $v_* = v(y^*)$, với $y^* = y^*(x, \omega)$ là nghiệm tối ưu, ở đây $y^* \in \mathbb{R}^p$. Giá trị tối ưu chính là $c^T\,.\,x^* + v(y^*)$

3 Vấn đề 1 [Công nghiệp - Sản xuất]

3.1 Tóm tắt bài toán

Một nhà máy muốn sản xuất 8 sản phẩm và cần mua 5 phụ kiện để sản xuất. Mỗi đơn vị sản phẩm i cần a_{ij} đơn vị phụ kiện j, với $i = 1, .., 8$ và $j = 1, .., 5$ (a_{ij} có thể bằng 0). Số lượng sản phẩm theo nhu cầu được biểu diễn qua vectơ ngẫu nhiên $D = (D_1, ..., D_8)$ với D_i là số nguyên từ 0 đến 10. Trước khi biết các nhu cầu, nhà máy muốn đặt mua các phụ kiện với giá b_j mỗi đơn vị phụ kiện j. Sau khi biết nhu cầu D, nhà máy muốn sản xuất số sản phẩm không vượt quá nhu cầu. Biết mỗi đơn vị sản phẩm i tốn chi phí l_i để sản xuất và giá bán là q_i; mỗi đơn vị phụ kiện j dư thừa được bán lại với giá s_j. Số ngữ cảnh xảy ra là S=2, với mật độ $p_s = 1/2$. Vậy nhà máy cần mua phụ kiện và sản xuất như thế nào để tối ưu lợi nhuận thu được.

3.2 Mô hình hóa bài toán

Giả sử số phụ kiện mua trước là $x_j, j = 1, .., 5$. Sau khi đã biết nhu cầu D, ta cần quyết định số sản phẩm cần sản xuất. Gọi số sản phẩm được sản xuất là $z_i, i = 1, .., 8$ và số phụ kiện dư thừa sau khi sản xuất là $y_j, j = 1, .., 5$. Với mỗi giá trị thực $d = (d_1, .., d_8)$ của vectơ ngẫu nhiên nhu cầu $D = (D_1, .., D_8)$, ta tìm chiến lược sản xuất tốt nhất bằng cách giải bài toán quy hoạch tuyến tính sau:

$$\min_{z,y} \sum_{i=1}^{8} (l_i - q_i) z_i - \sum_{j=1}^{8} s_j y_j$$

điều kiện:

$$y_j = x_j - \sum_{i=1}^{8} a_{ij} z_i, \quad j = 1, ..., 5, \quad 0 \leq z_i \leq d_i, \quad i = 1, ..., 8, \quad y_i \geq 0.$$

- Xét ma trận A với các phần tử a_{ij}, $i = 1, ..., 8$ và $j = 1, ..., 5$ ta viết lại bài toán gọn hơn như sau:

$$\min_{z,y} (l - q)^T z - s^T y \quad (1)$$

điều kiện:

$$y = x - A^T z, \quad 0 \leq z \leq d, \quad y \geq 0$$

- Ta thấy rằng nghiệm của bài toán (1), vectơ y và z, dựa trên x và giá trị thực d của vectơ nhu cầu D. Gọi Q(x,d) là giá trị cực tiểu của bài toán trên. Số lượng phụ kiện cần mua x_j có thể tìm ra từ bài toán sau:

$$\underset{x \geq 0}{Min} \quad b^T x + \mathbb{E}[Q(x, D)] \qquad (2)$$

với giá trị kì vọng được tính dựa trên vectơ nhu cầu D. Số hạng đầu của hàm mục tiêu biểu diễn chi phí mua phụ kiện, số hạng sau biểu diễn chi phí kì vọng của kế hoạch sản xuất tối ưu, nếu đã biết vectơ x.
- Bài toán (1) và (2) là ví dụ của two-stage stochastic programming problem, với bài toán (1) được gọi là second-stage problem và bài toán (2) là first-stage problem. Vì second-stage problem chứa dữ liệu ngẫu nhiên (nhu cầu ngẫu nhiên D), giá trị tối ưu Q(x,D) là một biến ngẫu nhiên. Phân phối của biến ngẫu nhiên này phụ thuộc vào x, do đó first-stage problem không thể giải nếu không biết các biến của second-stage problem.
- Có 2 ngữ cảnh d_1, d_2 với xác suất xảy ra $p_1 = p_2 = 1/2$ nên bài toán (1) - (2) có thể gộp lại thành 1 bài toán quy hoạch tuyến tính sau:

$$Min \quad b^T x + \sum_{k=1}^{K} p_k [(l - q)^T z^k - s^T y^k] \qquad (3)$$

điều kiện:

$$y^k = x - A^T z^k, \quad k = 1, ..., K,$$
$$0 \leq z^k \leq d^k, \quad y^k \geq 0, x \geq 0$$

với các biến của bài toán là vectơ x, y_1, y_2, z_1, z_2. Ta gộp second-stage problem (1) vào bài toán, nhưng mỗi nghiệm (y_k, z_k) vẫn phụ thuộc vào ngữ cảnh k, vì vectơ d_k là khác nhau với mỗi ngữ cảnh.
- Nghiệm của bài toán (3) chính là nghiệm cần tìm của bài toán phát biểu ở phần 1.

3.3 Bài toán cụ thể

Với số sản phẩm $n = 8$ và số linh kiện $m = 5$. Lấy ngẫu nhiên các ma trận sau:

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 6 & 4 & 4 \\ 2 & 2 & 1 & 2 & 3 \\ 5 & 1 & 6 & 5 & 2 \\ 4 & 5 & 6 & 1 & 1 \\ 3 & 1 & 2 & 5 & 2 \\ 1 & 1 & 5 & 4 & 2 \\ 3 & 1 & 2 & 1 & 2 \\ 6 & 2 & 5 & 1 & 8 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 4 \\ 7 \\ 3 \\ 5 \\ 5 \end{bmatrix} \quad \mathbf{s} = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 2 \\ 1 \end{bmatrix} \quad \mathbf{l} = \begin{bmatrix} 10 \\ 20 \\ 10 \\ 30 \\ 20 \\ 10 \\ 40 \\ 20 \end{bmatrix} \quad \mathbf{q} = \begin{bmatrix} 50 \\ 60 \\ 115 \\ 75 \\ 125 \\ 65 \\ 50 \\ 60 \end{bmatrix}$$

Chú thích

- A : sản phẩm thứ i cần j linh kiện (Ví dụ $a_{13} = 6$ tức là sản phẩm 1 cần 6 linh kiện 3 để sản xuất)
- b : giá tiền đặt trước của linh kiện b_j $(j = 1, 2, .., m)$ trước khi biết nhu cầu (demand)
- s : giá tiền của linh kiện còn sót lại $(s_j \leq b_j \ (j = 1, 2, .., m))$
- l : chi phí để sản xuất sản phẩm i $(i = 1, 2, .., n)$
- q : giá bán của từng sản phẩm i $(i = 1, 2, .., n)$

Để giải quyết bài toán thuận tiện, ta gộp 2 ma trận l và q lại bằng cách sử dụng phép trừ ma trận $l - q$. Ta gọi đó là ma trận lq

$$\mathbf{lq} = \begin{bmatrix} -40 \\ -40 \\ -105 \\ -45 \\ -105 \\ -55 \\ -10 \\ -40 \end{bmatrix}$$

Vì đây là bài toán lập trình ứng dụng ngẫu nhiên, nên ta chưa biết được nhu cầu từng sản phẩm là bao nhiêu, và có bao nhiêu tình huống (scenerio) có thể xảy ra. Với số scenerio $S = 2$ và xác suất xảy ra cho mỗi scenerio là $p = 1/2$. Ta đưa bài toán về dạng 2-SLPWR trong đó tuân theo luật **production** $\leq$ **demand**

The first - stage problem:

Ta bắt đầu tìm giá trị tối ưu của first - stage, đối với bài toán này, biến quyết định là biến x với giá trị preorder cost là ma trận $b = (b_1, b_2, .., b_m)$ đã được đề cập ở trên. Số lượng x_j được xác định vào biểu thức sau:

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

Để giải được biểu thức trên, ta tiếp tục xem xét về stage 2 của bài toán

The second - stage problem:

Ở giai đoạn 2, ta có 2 biến quyết định là y và z tương ứng với 2 scenerio. Mỗi scenerio có xác suất là 50%. Đề bài mô tả $d = D = (D_1, D_2, .., D_n)$ trong đó d_i ứng với xác suất p_i tuân theo quy tắc **Binomial Distribution BIN(10, 1/2)**.

Chạy file code đầu vào ta được ma trận d_1 và d_2 ứng với nhu cầu (demand) của từng scenerio.

	Products	value
0	Product 1	6.0
1	Product 2	4.0
2	Product 3	6.0
3	Product 4	4.0
4	Product 5	6.0
5	Product 6	6.0
6	Product 7	4.0
7	Product 8	7.0

	Products	value
0	Product 1	5.0
1	Product 2	8.0
2	Product 3	3.0
3	Product 4	3.0
4	Product 5	6.0
5	Product 6	3.0
6	Product 7	4.0
7	Product 8	4.0

$$d_1 = \begin{bmatrix} 6 \\ 4 \\ 6 \\ 4 \\ 6 \\ 6 \\ 4 \\ 7 \end{bmatrix} \qquad d_2 = \begin{bmatrix} 5 \\ 8 \\ 3 \\ 3 \\ 6 \\ 3 \\ 4 \\ 4 \end{bmatrix}$$

Chú thích

- d_1 : ma trận chứa các demand của scenerio 1
- d_2 : ma trận chứa các demand của scenerio 2

Ta mô hình bài toán như sau:

$$\begin{cases} min\ g(x,y,z) = b^T.x + lq^T.z - s^T.y \\ y = x - Az \\ 0 \le z \le d, y \ge 0 \end{cases}$$

với b, lq, A là các ma trận đã được đề cập ở phần trước

- x là ma trận (5x1): số linh kiện cần đặt trước sản xuất
- y là ma trận (5x1): số linh kiện còn dư trong kho

- y có 2 ma trận : y_1 và y_2 ứng với scenerio 1 và scenerio 2
- z là ma trận (8x1) : số sản phẩm cần đặt trước sản xuất (Điều kiện z $\leq$ demand)

Để tìm được ma trận chi phí tối ưu, ta tính toán biểu thức sau:

$$z = b^T.x + 0.5\,[lq^T.z_1 - s^T.y_1] + 0.5\,[lq^T.z_2 - s^T.y_2]$$

$$= [b_1 \quad b_2 \quad b_3 \quad b_4] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + 0.5\,([lq_1 \quad lq_2 \quad .. \quad lq_8] \begin{bmatrix} z1_{11} \\ z1_{21} \\ ... \\ z1_{81} \end{bmatrix} - [s_{11} \quad s_{21} \quad .. \quad s_{51}] \begin{bmatrix} y1_{11} \\ y1_{21} \\ ... \\ y1_{51} \end{bmatrix})$$

$$+ \quad 0.5\,([lq_1 \quad lq_2 \quad .. \quad lq_8] \begin{bmatrix} z2_{11} \\ z2_{21} \\ ... \\ z2_{81} \end{bmatrix} - [s_{11} \quad s_{21} \quad .. \quad s_{51}] \begin{bmatrix} y2_{11} \\ y2_{21} \\ ... \\ y2_{51} \end{bmatrix})$$

Ta sẽ sử dụng giải thuật Simplex và tìm ra được các nghiệm tối ưu cũng như các kết quả như sau:

```
Gia tri ham muc tieu
-345.0
```

Giá trị hàm mục tiêu

```
Bang linh kien con sot lai
     Sites  level  marginal  lower  upper  scale
0  Sites 1    0.0       2.0    0.0    inf    1.0
1  Sites 2    0.0       4.0    0.0    inf    1.0
2  Sites 3    0.0       2.0    0.0    inf    1.0
3  Sites 4    0.0       3.0    0.0    inf    1.0
4  Sites 5    0.0       4.0    0.0    inf    1.0
     Sites  level  marginal  lower  upper  scale
0  Sites 1    0.0       0.0    0.0    inf    1.0
1  Sites 2    0.0       0.0    0.0    inf    1.0
2  Sites 3    0.0       0.0    0.0    inf    1.0
3  Sites 4    0.0       0.0    0.0    inf    1.0
4  Sites 5    0.0       0.0    0.0    inf    1.0
```

Số linh kiện còn sót lại

Bang san pham can dat truoc san xuat

	Products	level	marginal	lower	upper	scale
0	Product 1	0.0	47.0	0.0	inf	1.0
1	Product 2	0.0	10.0	0.0	inf	1.0
2	Product 3	3.0	0.0	0.0	inf	1.0
3	Product 4	0.0	34.0	0.0	inf	1.0
4	Product 5	6.0	0.0	0.0	inf	1.0
5	Product 6	0.0	1.0	0.0	inf	1.0
6	Product 7	0.0	30.0	0.0	inf	1.0
7	Product 8	0.0	58.0	0.0	inf	1.0

	Products	level	marginal	lower	upper	scale
0	Product 1	0.0	47.0	0.0	inf	1.0
1	Product 2	0.0	10.0	0.0	inf	1.0
2	Product 3	3.0	0.0	0.0	inf	1.0
3	Product 4	0.0	34.0	0.0	inf	1.0
4	Product 5	6.0	0.0	0.0	inf	1.0
5	Product 6	0.0	1.0	0.0	inf	1.0
6	Product 7	0.0	30.0	0.0	inf	1.0
7	Product 8	0.0	58.0	0.0	inf	1.0

Số sản phẩm cần đặt trước sản xuất

Bang linh kien dat truoc

	Sites	level	marginal	lower	upper	scale
0	Sites 1	33.0	0.0	0.0	inf	1.0
1	Sites 2	9.0	0.0	0.0	inf	1.0
2	Sites 3	30.0	0.0	0.0	inf	1.0
3	Sites 4	45.0	0.0	0.0	inf	1.0
4	Sites 5	18.0	0.0	0.0	inf	1.0

Số linh kiện cần đặt trước

4 Ứng dụng Stochastic Linear Program cho việc lập kế hoạch sơ tán khi ứng phó thiên tai

4.1 Bối cảnh

Các thảm họa cực đoan, ví như sóng thần, động đất, lốc xoáy, chiến tranh ... có ảnh hưởng rất lớn tới con người, chúng thường xảy ra đột ngột khiến ta không kịp phản ứng, gây nên tổn thất nặng nề về con người. Vì thế nên các nhà nghiên cứu vẫn luôn tìm các để tối ưu hóa bài toán sơ tán trong các thảm họa bất ngờ, đảm bảo cứu được nhiều mạng người nhất.

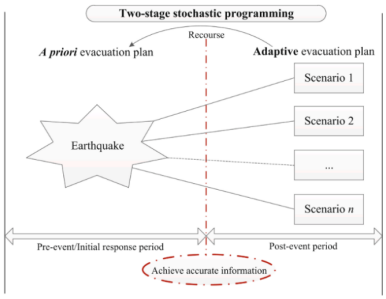
Two-stage stochastic programming là một phương pháp hữu hiệu cho bài toán trên. **Firststage decicsion** phải là một kế hoạch sơ tán vững chắc và đáng

tin cậy trong mọi viễn cảnh có thể xảy ra. **Second-stage decision** là kế hoạch sơ tán cho những người bị ảnh hưởng trong một viễn cảnh cụ thể.

4.2 Phát biểu bài toán

4.2.1 Bài toán sơ tán dưới dạng two-stage stochastic programming

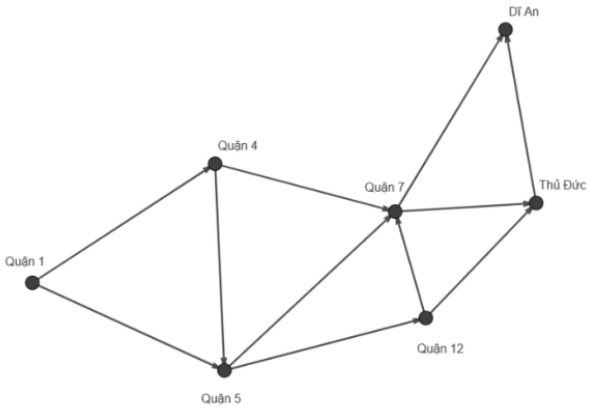
Các cơ quan phải lập ra một kế hoạch sơ tán tối ưu cho người dân. Tuy nhiên tại thời điểm ban đầu, các thống kê về thiệt hại đường xá, lưu lượng giao thông chưa được ghi nhận. Vì vậy, ta phải lên kế hoạch sơ tán trong điều kiện một số thông tin bị thiếu sót và chỉ có thể ước tính xác xuất xảy ra.
Bài toán two-stage stochastic programming cho kế hoạch sơ tán được phát biểu như sau:



- Giai đoạn đầu tiên, trước ngưỡng thời gian $\tilde{T}$: Động đất xảy ra, người dân nhận được cảnh báo. Chính phủ lên kế hoạch sơ tán tối ưu trong điều kiện thiếu sót thông tin và gửi tới người dân.

- Giai đoạn hai, sau ngưỡng thời gian $\tilde{T}$: Sau khi nhận được thông tin về tình hình đường xá, lưu lượng. Chính phủ sẽ lên một kế hoạch thích ứng với bối cảnh xảy ra và gửi đến người dân sơ tán.

Giả sử ở thành phố A, động đất xảy ra ở **quận 1**. Khi đó, người dân cần sơ tán tới địa điểm được đảm bảo an toàn (không bị động đất lan tới) đó là thành phố Dĩ An. Các tuyến đường nối giữa các địa điểm được thể hiện như hình dưới, động đất xả ra đã làm thiệt hại một vài tuyến đường yếu kém. Trước đó, các tuyến đường (Q.5 → Q.12) và (Q.12 → Thủ đức) có khả năng cao bị nứt gãy. Sau khi tính toán, kế hoạch tiên quyết (priori plan) sẽ là người dân phường a, b đi theo tuyến đường từ Q.1 → Q.4 → Q.7 → Dĩ An, còn người dân quận c sẽ đi theo tuyến đường Q.1 → Q.5 → Q.7 → Dĩ An. Sau ngưỡng thời gian, chính phủ nhận được thông tin tuyến đường Q.7 → Dĩ An bị ảnh hưởng làm giảm lưu lượng tối đa và thời gian di chuyển trên tuyến đường, và tuyến Q. 12 → Thủ Đức bị phá hủy hoàn toàn. Do đó kế hoạch thích ứng trong viễn cảnh này sẽ là người dân quận c thay đổi đi từ Q.7 → Thủ Đức → Dĩ An.



Hình 2. Sơ đồ các đường liên kết các địa điểm trong thành phố

4.2.2 Mô tả bài toán min-cost flow trong tối ưu việc sơ tán

Ta sẽ diễn tả việc lên kế hoạch cụ thể để sơ cứu người dân khỏi vùng nguy hiểm như là bài toán min-cost flow, với thời gian di chuyển và sức chứa trên các đường đi. Mục tiêu là di chuyển người dân tới nơi an toàn trong lượng thời gian tối ưu trên mạng **G(V, A, C, U, D)**. Trong đó:

- **V** là tập các đỉnh, hay là các địa điểm trên thực tế
- **A** là tập các cạnh nối các đỉnh
- $C(i,j)$, kí hiệu c_{ij}, là thời gian di chuyển trên đoạn đường (i,j) thuộc **A**.
- $U(i,j)$, kí hiệu u_{ij}, là sức chứa trên đoạn (i,j)
- $D(i)$, kí hiệu d_i, thể hiện luồng ở đỉnh i.

Với lượng người phù hợp với sức chứa của đoạn (i,j), thời gian đi trên đoạn này không thay đổi. Sẽ có đỉnh source, tức là nguồn cần sơ tán người dân khỏi và đỉnh sink, nơi an toàn để sơ tán người dân tới. Với mô hình two-stage stochastic programming, ở giai đoạn đầu tiên ta chỉ biết thời gian di chuyển và sức chứa dưới một xác suất cụ thể cho từng viễn cảnh. Nhưng người dân sẽ cần được sơ tán ngay lập tức dù chưa có đủ thông tin. Ở giai đoạn hai, kế hoạch sơ tán được xác định cụ thể khi đã biết $U(i,j)$ và $C(i,j)$ với mọi (i,j) thuộc A. Như vậy, hàm mục tiêu của ta sẽ là gồm giai đoạn đầu với chi phí (thời gian) phạt (tăng thêm) và giá trị kỳ vọng của chi phí giai đoạn hai.

4.3 Công thức, ký hiệu mô hình

4.3.1 Ký hiệu

Ta có bảng ký hiệu để mô tả công thức bài toán, các ràng buộc và hàm mục tiêu.

Ký hiệu	Định nghĩa
V	Tập hợp các đỉnh
i, j	Chỉ số của đỉnh i, j thuộc V
(i, j)	Cạnh có hướng từ i đến j
s	Chỉ số của viễn cảnh
S	Số viễn cảnh tối đa
v	Giá trị cung cấp từ đỉnh phát (source node)
$\tilde{T}$	Ngưỡng thời gian
T	Tổng đoạn thời gian
u_{ij}	Sức chứa của đoạn (i, j)
$u_{ij}^s(t)$	Sức chứa của đoạn (i, j) trong viễn cảnh s ở thời gian t
$c_{ij}^s(t)$	Thời gian di chuyển trên đoạn (i, j) trong viễn cảnh s ở thời gian t
μ_s	Xác suất viễn cảnh s

4.3.2 Biến quyết định

Có hai kiểu biến quyết định là:

- x_{ij} - luồng trên đoạn (i, j)
- $y_{ij}^s(t)$ - luồng trên đoạn (i, j) ở viễn cảnh s tại thời gian t

4.3.3 Các ràng buộc của bài toán

* Giai đoạn một:
Vì chúng ta sẽ thực hiện sơ tán người dân từ đỉnh source (s) đến sink (t). Trong quá trình đó, ta sẽ đi qua một số node trung gian, vì những node trung gian, trên thực tế là những địa điểm không được đảm bảo an toàn, nên người dân đi qua đó nhưng không lưu lại ở đó. Vậy nên luồng trên các cạnh phải thỏa mãn điều kiện như sau:

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

Trong đó d_i được tính theo định nghĩa:

$$d_i = \begin{cases} v, & i = source \\ -v, & i = sink \\ 0, & otherwise \end{cases}$$

(Tổng luồng đi vào phải bằng tổng luồng đi ra tại các node trung gian)

Ta lưu ý rằng công thức trên hoàn toàn không tương đương với

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

Đồng thời, mỗi đoạn đường sẽ có một sức chứa nhất định, nên: $0 \leq x_{ij} \leq u_{ij}, \ \forall (i,j) \in A$ (2)
Ở giai đoạn đầu tiên này, với hai điều kiện trên, ta có thể sinh ra một chu trình lặp đi lặp lại và không tới được đích. Để tránh điều này, ta có giá trị link penalty $p_{ij}, (i,j) \in A$, khi đi một chu trình thì giá trị này sẽ khiến giá trị hàm thời gian phạt tăng lên. Hàm phạt được định nghĩa là:

$$f(\mathbf{X}) = \sum_{(i,j)\in A} p_{ij}x_{ij} \quad (3)$$

mà:

$$\mathbf{X} := \{x_{ij}\}_{(i,j)\in A}$$

Ta sẽ tối ưu hàm phạt ứng với giai đoạn đầu trong hàm mục tiêu.

* Giai đoạn hai:
Giai đoạn thứ hai là giai đoạn sau khi ta đã biết được viễn cảnh cụ thể sẽ xảy ra, đánh giá từ giai đoạn thứ nhất để đạt được kế hoạch sơ tán hiệu quả nhất. Người dân sẽ được một chỉ dẫn cụ thể với thời gian sơ tán thấp nhất. Kế hoạch sơ tán trước ngưỡng thời gian là như nhau với mọi viễn cảnh có thể xảy ra trong tương lai. Trong khoảng thời gian ban đầu đến ngưỡng $\tilde{T}$, tổng số lượng người tán trên đường (i,j) được định nghĩa là x_{ij}.

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

Về cơ bản, trong mọi viễn cảnh thì khoảng thời gian từ 0 đến $\tilde{T}$, số lượng người được sơ cứu là như nhau theo kế hoạch tiền sơ cứu được đề ra ngay từ đầu.

Mô hình tối ưu trong giai đoạn hai với mục đích tối ưu tổng thời gian sơ cứu của những người bị ảnh hưởng từ nơi nguy hiểm đến nơi an toàn như sau:

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

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

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

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

Hàm mục tiêu (5) là tổng thời gian sơ tán của tất cả lưu lượng giao thông trong viễn cảnh s. Ràng buộc (6), (7) là cân bằng luồng tại các đỉnh trung gian và đảm bảo sức chứa của mỗi đoạn.
Ràng buộc (6) hiểu đơn giản là tại hai khoảng thời gian t', t liên tiếp, tổng luồng đi từ đỉnh i đến mọi đỉnh j, $(i,j) \in A$, tại thời điểm t trừ tổng luồng đi từ mọi j vào i, $(i,j) \in A$ tại thời điểm t' bằng 0 tại các đỉnh trung gian – không người dân nào bị bỏ lại đến cuối cùng.
Ràng buộc (8) là ràng buộc đảm bảo là kế hoạch tiền sơ tán được thực hiện trước khi biết được viễn cảnh sẽ xảy ra, nói cách khác, dù viễn cảnh nào xảy ra thì trong giai đoạn đầu tiên, ta đều thực hiện theo kế hoạch này

4.3.4 Mô hình hóa bài toán sơ tán với two-stage stochastic programming

Vấn đề sơ tán được quan tâm là có được một kế hoạch vững chắc trong giai đoạn đầu tiên bằng cách đánh giá các kế hoạch sơ tán ứng với từng viễn cảnh ở giai đoạn thứ hai. Để có thể làm được, ta sẽ đánh giá kế hoạch sơ tán trong giai đoạn đầu tiên với giá trị kỳ vọng của tổng thời gian sơ tán trong từng viễn cảnh s với xác suất của mỗi viễn cảnh là $u_s, s = 1, 2, .., S$. Và ta sẽ tối ưu thời gian tốn kém cho kế hoạch tiền sơ tán và giá trị kỳ vọng của tổng thời gian sơ cứu từng viễn cảnh theo như mô hình 2-SLP như sau:

$$
\begin{cases}
min \sum_{(i,j)\in A} p_{ij}.x_{ij} \\
s.t. \\
\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i, \forall i \in V \\
\\
0 \le x_{ij} \le u_{ij}, \forall (i,j) \in A \\
\\
\text{in which,} \\
\\
Q(Y,s) = min \sum_{(i,j)\in A_s} c^s_{ij}(t).y^s_{ij}(t) \\
s.t. \\
\sum_{(i_t,j'_t)\in A_s} y^s_{ij}(t) - \sum_{(j'_t,i_t)\in A_s} y^s_{ji}(t') = d^s_i(t), \forall i \in V, t \in \{1,2,..,T\}, s = 1,2,..,S \\
\\
0 \le y^s_{ij}(t) \le u^s_{ij}(t), \forall (i,j) \in A, t \in \{1,2,..,T\}, s = 1,2,..,S \\
\\
y^s_{ij}(t) = x_{ij}, \quad (i,j) \in A, s = 1,2,..,S, t \le \tilde{T} \qquad (9)
\end{cases}
$$

Mô hình này là mô hình phụ thuộc thời gian và two-stage stochastic. Vì số lượng viễn cảnh đề ra là hữu hạn nên mô hình trên có thể đưa về **single stage** như sau:

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

Ở đây, ta đưa việc tối lưu từng viễn cảnh lên thành một hàm mục tiêu để tối ưu với tập các biến quyết định là tập $\{x_{ij}, (i,j) \in A\}$, và $\{y_{ij}^s(t), (i,j) \in A, t = 1,2,..,T, s = 1,2,..,S\}$. Tập các biến ngẫu nhiên là thời gian di chuyển và sức chứa của từng đoạn.

4.4 Giải thuật

Mô hình (10) là mô hình quy hoạch tuyến tính nguyên, với các biến quyết định như đã mô tả bên trên. Trong mô hình này, ràng buộc cuối cùng là một ràng buộc phức tạp, vì ràng buộc các biến x_{ij} và $y_{ij}^s(t)$. Giải thuật sẽ trở nên chậm và tiêu tốn bộ nhớ. Do đó, ta sẽ áp dụng phương pháp Lagrangian relaxation để đơn giản hóa bài toán.

4.4.1 Lagrangian relaxation

Lagrangian relaxation là một phương pháp ước tính xấp sỉ nghiệm của một bài toán tối ưu với ràng buộc phức tạp, đưa nó về bài toán đơn giản hơn. Phương pháp này sẽ tính toán chi phí của việc vi phạm đẳng thức/ bất đẳng thức trong ràng buộc gốc, và sẽ cố gắng tối ưu hóa hàm chi phí này cùng với hàm mục tiêu ban đầu.
Trong bài toán quy hoạch tuyến tính, Lagrangian relaxation có thể được áp dụng như sau:
Cho bài toán quy hoạch tuyến tính:

$$min \quad \mathbf{c}^T\mathbf{x}$$

s.t.

$$\mathbf{Ax} = \mathbf{b}$$
$$\mathbf{x} \in X$$

Khi đó, phương pháp này nới lỏng ràng buộc bằng cách tính toán chi phí (hay 'giá trị phạt') cho việc 'sai lệch' với ràng buộc gốc, đưa hàm chi phí vào hàm mục tiêu để tối ưu. Cụ thể như sau:

$$min \quad \mathbf{c}^T\mathbf{x} + \mu^T(\mathbf{Ax - b})$$
$$\text{s.t.}$$
$$\mathbf{x} \in X$$

Hàm Lagrangian được định nghĩa là: $L(\mu) = min\{\mathbf{c}^T\mathbf{x} + \mu^T(\mathbf{Ax - b}) \mid \mathbf{x} \in X\}$, Lúc này, tùy thuộc vào giá trị μ^T mà ta sẽ tính toán được xấp xỉ giá trị tối ưu trong bài toán gốc, thuật toán cập nhật μ^T qua từng bước để đạt kết quả tốt nhất cho xấp xỉ giá trị tối ưu trong bài toán gốc sẽ được trình bày sau.

4.4.2 Phân tách mô hình bài toán

Quay lại mô hình (10), ta sẽ áp dụng phương pháp Lagrangian relaxation để nới lỏng ràng buộc cuối. Nhân tử Lagrange tương ứng sẽ là $\alpha_{ij}^s, (i,j) \in A, s = 1,2,..,S$. Khi đó, ta sẽ cộng thêm hàm mục tiêu như sau:

$$\sum_{s=1}^{S} \sum_{(i,j)\in A} \sum_{t\leq \tilde{T}} \alpha_{ij}^s (y_{ij}^s(t) - x_{ij}^s(t)) \qquad (12)$$

Vậy, sau khi nới lỏng điều kiện để được công thức (12), mô hình (10) có thể được nới lỏng điều kiện như sau:

$$min \quad \sum_{(i,j)\in A} p_{ij}.x_{ij} + \sum_{s=1}^{S} \left(\mu_s . \sum_{(i,j)\in A_s} c_{ij}^s(t).y_{ij}^s(t) \right) + \sum_{s=1}^{S} \sum_{(i,j)\in A} \sum_{t\leq \tilde{T}} \alpha_{ij}^s (y_{ij}^s(t) - x_{ij}^s(t))$$

$$\begin{cases} s.t. \\ \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i, \quad \forall i \in V \\ 0 \leq x_{ij} \leq u_{ij}, \quad \forall (i,j) \in A \\ \sum_{(i_t,j_t')\in A_s} y_{ij}^s(t) - \sum_{(j_t',i_t)\in A_s} y_{ij}^s(t') = d_i^s(t), \quad \forall i \in V, \\ t \in \{0,1,..,T\}, \quad s = 1,2,..,S \\ 0 \leq y_{ij}^s(t) \leq u_{ij}^s(t), \quad \forall (i,j) \in A, \quad t \in \{0,1,..,T\}, \quad s = 1,2,..,S \end{cases} \qquad (13)$$

Nhờ vào việc nới lỏng điều kiện cuối, các biến **X** và **Y** có thể tách biệt với nhau. Do đó, bài toán tối ưu ở mô hình trên có thể tách thành hai bài toán phụ như sau:

4.4.2.a Bài toán phụ 1: Bài toán min-cost flow

Bài toán phụ 1 được mô hình như sau:

$$\begin{cases} min\ \ SP1(\alpha) = \sum_{(i,j)\in A}\left(p_{ij} - \sum_{s=1}^{S}\sum_{t\le\tilde{T}}\alpha_{ij}^{s}(t)\right).x_{ij} = \sum_{(i,j)\in A} g_{ij}.x_{ij} \\ s.t \\ \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i, \forall i \in V \\ 0 \le x_{ij} \le u_{ij}, \forall (i,j) \in A \end{cases} \quad (14)$$

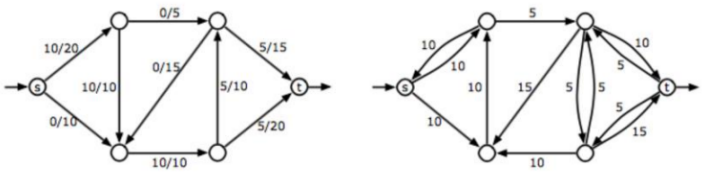
Khi đặt $g_{ij} = p_{ij} - \sum_{s=1}^{S}\sum_{t\le\tilde{T}}\alpha_{ij}^{s}(t)$, ta coi đó như là chi phí trên một đoạn. Vì vậy bài toán phụ 1 có thể được coi là bài toán **min-cost flow**.

* Thuật toán 1: Successive shortest path algorithm
Bước 1:Giả sử $G(V, A, U, C)$ là một mạng và x là luồng khả thi từ s tới t với chi phí tối thiểu trên mạng.
Khi đó, mạng thặng dư $G(V, A(x), U(x), C(x))$ được định nghĩa như sau:

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

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

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



Hình 3. Chuyển từ mạng ban đầu với luồng x sang mạng thặng dư

Đường tăng luồng được định nghĩa là đường đi đơn từ s tới t mà trong mạng thặng dư mà kênh trên đường đi chưa bị bão hòa, tức là sức chứa còn lại trên các đoạn có thể đi qua trên mạng thặng dư vẫn lớn hơn 0.

Bước 2: Chương trình sẽ dừng nếu giá trị luồng của x bằng giá trị K (luồng cần gửi từ s tới t) hoặc không còn đường tăng luồng. Nếu không ta sẽ đi tìm đường tăng luồng có chi phí nhỏ nhất.
Để tìm được đường tăng luồng có chi phí nhỏ nhất, ta có thể coi mạng thặng dư là một đồ thị có hướng, cạnh trọng số là giá trị u_{ij}. Sau đó tìm đường đi ngắn nhất (tức là chi phí thấp nhất) từ s tới t, có thể áp dụng các thuật toán quen thuộc như bellman-ford, floyd-warshall,...

Bước 3: Thực hiện tăng luồng trên đường tăng luồng vừa tìm được và cập nhật mạng thặng dư. Chú ý trường hợp vượt quá giá trị K cho trước.
Nếu ta gán K bằng vô hạn thì thuật toán trên sẽ cho ta luồng cực đại với chi phí tối thiểu.

Bây giờ, áp dụng thuật toán 1, ta có thể giải quyết được bài toán phụ 1 với chi phí tối thiểu. Ta cũng tìm được giá trị của X, qua đó tính toán được kế hoạch sơ tán ban đầu ứng với α.

4.4.2.b Bài toán phụ 2: Bài toán min-cost flow phụ thuộc thời gian

Bài toán phụ 2 được mô hình như sau:

$$\begin{cases} min\ \ SP2(\alpha) = \sum_{s=1}^{S} \sum_{(i,j)\in A} \left(\sum_{t\in\{0,1,..,T\}} \mu_s.c_{ij}^s(t) + \sum_{t\leq\tilde{T}} \alpha_{ij}^s(t) \right) y_{ij}^s(t) \\ s.t. \\ \sum_{(i_t,j'_t)\in A_s} y_{ij}^s(t) - \sum_{(j'_t,i_t)\in A_s} y_{ij}^s(t') = d_i^s(t),\ \ \forall i \in V, \\ \\ t \in \{0,1,..,T\},\ \ s = 1,2,..,S \\ \\ 0 \leq y_{ij}^s(t) \leq u_{ij}^s(t),\ \ \forall (i,j) \in A,\ \ t \in \{0,1,..,T\},\ \ s = 1,2,..,S \end{cases} \quad (15)$$

Bài toán phụ 2 có thể phân tác thành S bài toán phụ khác, mỗi bài toán phụ ứng với một viễn cảnh có thể xảy ra.

$$\begin{cases} min\ \ SP2(\alpha, s) = \sum_{(i,j)\in A} \left(\sum_{t\in\{0,1,..,T\}} \mu_s.c_{ij}^s(t) + \sum_{t\leq\tilde{T}} \alpha_{ij}^s(t) \right) y_{ij}^s(t) \\ s.t. \\ \sum_{(i_t,j'_t)\in A_s} y_{ij}^s(t) - \sum_{(j'_t,i_t)\in A_s} y_{ij}^s(t') = d_i^s(t),\ \ \forall i \in V,\ \ t \in \{0,1,..,T\} \\ \\ 0 \leq y_{ij}^s(t) \leq u_{ij}^s(t),\ \ \forall (i,j) \in A,\ \ t \in \{0,1,..,T\},\ \ s = 1,2,..,S \end{cases} \quad (16)$$

Ta có thể đặt : $g_{ij}^s(t) = \begin{cases} \mu_s.c_{ij}^s(t) + \alpha_{ij}^s(t), t \leq \tilde{T} \\ \mu_s.c_{ij}^s(t), \qquad \tilde{T} \leq t \leq T \end{cases}$

Khi đó, bài toán (16) trở thành bài toán min-cost flow phụ thuộc thời gian. Ta sẽ áp dụng thuật toán 1 có sửa đổi. Mạng thặng dư $G(V, A(y(t)), C(y(t)), U(y(t)))$ được định nghĩa lại như sau:
$A_s(y(t)) = \{(i_t, j'_t) \mid (i_t, j'_t) \in A_s, y_{ij}^s < u_{ij}^s\} \cup \{(j'_t, i_t) \mid (j'_t, i_t) \in A_s, y_{ij}^s > 0\}, \ s = 1, 2, .., S$

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

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

Ta lưu ý rằng mỗi đoạn (i, j) có thời gian di chuyển là $c_{ij}(t)$, ta coi như thời gian này là cố định trong suốt quá trình người dân đi từ $i \to j$. Nói cách khác, nếu một người bắt đầu từ đỉnh i tại thời điểm $t[k]$ đến j, thì người này sẽ đến j tại thời điểm $t[k] + c_{ij}(t[k])$. Nếu muốn bài toán chính xác hơn, ta có thể tự thêm những địa điểm trên đường nối các đỉnh trong đồ thị gốc và chia đều thời gian di chuyển ra.
Vì c_{ij} có thể mang giá trị âm trên mạng thặng dư, nên thuật toán dijkstra cho time-dependent không đảm bảo chi phí thấp nhất (ở bài toán sơ tán này, chi phí là thời gian). Tuy nhiên, ta có thể áp dụng thuật toán Bellman-Ford cho bài toán shortest path time-dependent trong trường hợp này như sau:
Đầu tiên, ta khởi tạo mảng L gồm $|V|$ phần tử, $L[i]$ thể hiện thời gian trên đường đi ngắn nhất từ điểm đầu (s) tới i. Khởi tạo mảng p để truy vết đường đi đã tính toán.

Gán $L[s] = t[0]$ và $L[i] = \infty$ với mọi i khác r.

Lặp lại $N - 1$ lần: Với mỗi đỉnh i từ 1 tới V, nếu (i, j) thuộc A và $L[j] \ != \ \infty$:

Nếu $L[i] + C[i][j][L[i]] < L[j]$ thì $L[j] = L[i] + C[i][j][L[i]]$ và $q[j] = i$.

Chi tiết hơn về Bellman – Ford, các edge case, có thể tham khảo thêm tại Ref 3. Ở đây, chính yếu chỉ cần thay đổi cách cập nhật giá trị để phù hợp với bài toán time-dependent.

Sau khi tìm được đường đi chi phí thấp nhất, ta tăng luồng và cập nhật mạng thặng dư như ở thuật toán 1

Giải được bài toán phụ 1 và 2, ta tìm được nghiệm tối ưu $Z^*_{LR}(\alpha)$ của mô hình (13):

$$Z^*_{LR}(\alpha) = Z^*_{SP1}(\alpha) + Z^*_{SP2}(\alpha)$$

Giá trị này chính là một cận dưới của giá trị tối ưu trong mô hình (10) ban đầu. Cận dưới này càng lớn thì càng tiến gần tới giá trị tối ưu, tức là ta sẽ sử dụng cận dưới lớn nhất để ước tính giá trị tối ưu trong mô hình (10):

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

4.4.3 Xác định cận trên

Nếu lời giải nới lỏng từ các bài toán phụ (14) và (15) thỏa mãn ràng buộc (feasible) ở mô hình gốc, thì có thể coi đó là một giá trị tiềm năng, gọi là v. Vì giá trị tối ưu cho mô hình ban đầu là giá trị nhỏ nhất Z của hàm tối ưu, nên nếu v thỏa mãn các ràng buộc của mô hình thì rõ ràng $v \geq Z$. Vậy nên v là một cận trên của giá trị tối ưu ban đầu.

Lấy giá trị nhỏ nhất trong các cận trên tìm được, đó là cận trên gần Z nhất.

4.4.4 Giải thuật giải bài toán 2-SLP trong sơ tán người dân áp dụng Lagrangian relaxation và subgradient optimization.

*** Thuật toán 2: Subgradient method**

Phát biểu toán học:
Cho $f : \mathbb{R}^n \to \mathbb{R}$ là một hàm lõm. Phương phát subgradient cổ điển thực hiện lặp m lần và cập nhật biến x để hàm f đạt cực tiểu. Cụ thể tại lần lặp thứ k:

. $$x^{(k+1)} = s^{(k)} - \alpha_k g^{(k)}$$

Trong đó $g^{(k)}$ là một subgradient của f tại $x^{(k)}$. Nếu f có thể lấy đạo hàm thì $g^{(k)}$ là vector gradient của f tại $x^{(k)}$. Trong một số mô hình học sâu, ta có thể áp dụng thuật toán này để cập nhật các giá trị trọng số quan trọng. Tương tự, ở đây, ta sẽ áp dụng nó để cập nhật các nhân tử Lagrange trong mô hình nới lỏng của ta.
Cụ thể, tổng quát các làm như sau:

Bước 1: Gán $k = 1$, gán giá trị dương ngẫu nhiên cho α^s_{ij}, và $UB = \infty$ (Upper Bound).

Bước 2:

- Giải bài toán phụ (14) sử dụng thuật toán 1. Tìm được kế hoạch sơ tán **X**.
- Giải bài toán phụ (14) sử dụng thuật toán 1 có chỉnh sửa. Tìm được kế hoạch sơ tán **Y**.
- Cập nhật Lower bound, Upper Bound và tính Relative Gap.

Bước 3: Cập nhật nhân tử Lagrange α_{ij}^s bằng subgradient: $\alpha_{ij}^s(t) := \alpha_{ij}^s(t) + \theta^{(k)} \times (y_{ij}^s(t) - xij)$. Với $\theta^{(k)}$ là bước nhảy tại lần lặp k.

Bước 4: Dừng chương trình nếu đã đủ số vòng lặp hoặc đạt được relative gap đủ tốt, nếu không quay lại bước 2.

4.5 Hiện thực mô hình bài toán bằng mô phỏng trên máy tính.

Xây dựng giải thuật min-cost flow:
Ta xây dựng hàm tìm min-cost path trên mạng như sau:

Tạo mảng $dis[V]$ thể hiện khoảng cách từ source tới các đỉnh: $dis[source] = 0$; $dis[i] = \infty$, $\forall\ i\ != source$;
Lặp $n - 1$ lần:
Xét tất cả cạnh (i, j) trong đồ thị:
If flow == capacity: continue
If $dis[j] > dis[i] + weight(i, j)$: $dis[j] = dis[i] + weight(i, j)$; $parent[j] = i$;

If $dis[sink] = \infty$ return false;
Return true;

Và hàm tìm min-cost path được xây dựng như sau:

While (find_path(source, sink)):
Xét tất cả cạnh trên đường tăng luồng: addflow = $min(capacity(i, j) - flow(i, j), cost+ = weight(i, j)$
Nếu total_flow + addflow > K: flow = K – total_flow
Cost = cost * addflow;
Trên đường tăng luồng,cập nhập flow cạnh xuôi: +addflow, cạnh ngược-addflow trên mạng thặng dư.
Cập nhật total_cost, total_flow;

Code C++:

```
bool find_path(int from, int to, vector<Edge*>& output)
{
    fill(distances, distances+nodecount, 1e12);
    fill(parents, parents+nodecount, (Edge*)0);
```

```
        distances[from] = 0;

        bool updated = true;
        int nt = nodecount;
        while (updated && nt>0)
        {
            nt--;
            updated = false;
            for (int j = 0; j < nodecount; ++j)
                for (int k = 0; k < (int)adj[j].size(); ++k) {
                    Edge* e = adj[j][k];
                    if (e → f ≥ e → c) continue;
                    if (distances[e → b] > distances[e → a] + e → w)
                    {
                        distances[e → b] = distances[e → a] + e → w;
                        parents[e → b] = e;
                        updated = true;
                    }
                }
        }

        output.clear();
        if (distances[to] == (ll)1e12) return false;
        int cur = to;

        while (parents[cur])
        {
            output.push_back(parents[cur]);
            cur = parents[cur] → a;
        }
        return true;
}

float min_cost_max_flow(int source, int sink, int K)
{
        ll total_cost = 0;
        int tt_flow = 0;
        vector < Edge* > p;
        while (find_path(source, sink, p))
        {
            int flow = INT_MAX;
            for (int i = 0; i < p.size(); ++i)
```

```
            if (p[i] → c − p[i] → f < flow) flow = p[i] → c − p[i] → f;
        if (tt_flow + flow > K)  flow = K − tt_flow;
        ll cost = 0;
        for (int i = 0; i < p.size(); ++i) {
            cost + = p[i] → w;
            p[i] → f+ = flow;
            p[i] → r → f− = flow;
        }
        cost = cost * flow; // cost per flow
        total_cost + = cost;
        tt_flow + = flow;
        if (tt_flow ≥ K) break;
    }
    return (float)total_cost/1000.0;
}
```

Tương tự với bài toán min-cost flow time dependent, ta thêm yếu tố thời gian vào mạng thặng dư và tìm min-cost path như trên.

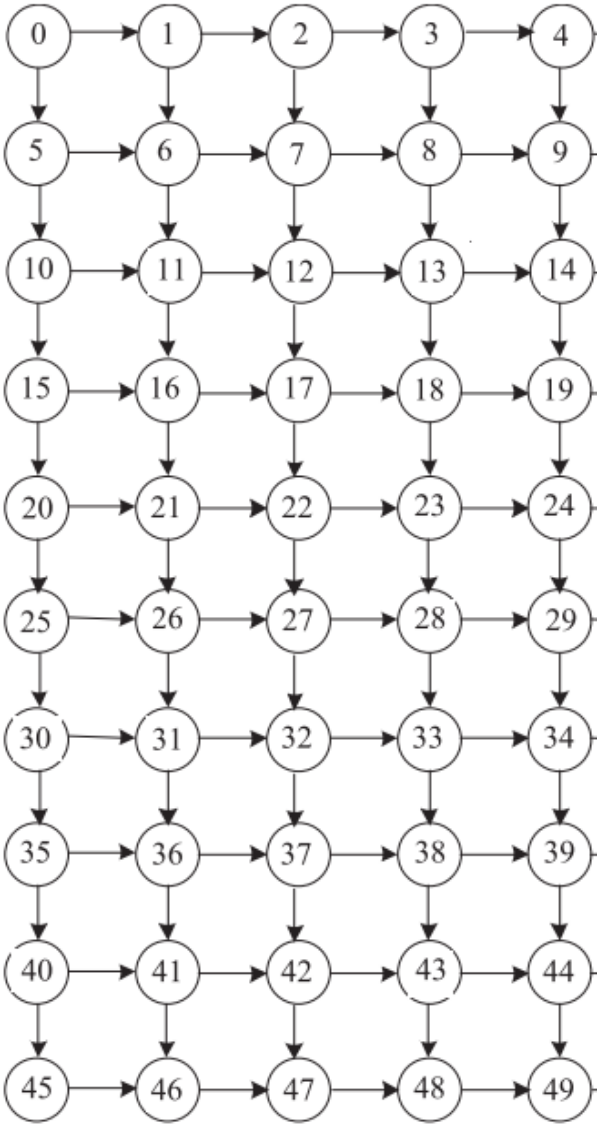
Bây giờ, ta sẽ thực hiện mô phỏng để tính toán giá trị tối ưu.

Dữ liệu của từng viễn cảnh được random ngẫu nhiên, cụ thể travel time trong đoạn [1,4], và capacity trong đoạn [80,100].

Bây giờ ta sẽ tính toán giá trị tối ưu của bài toán sơ tán người dân trên một mạng gồm 50 đỉnh và 85 cạnh như hình bên. Giả sử rằng có 160 xe cần sơ tán tới vùng an toàn, thời gian là 120 phút. Khoảng thời gian đơn vị là phút

Các cặp (vị trí cần sơ tán, vị trí an toàn) được nhập vào.

Số viễn cảnh có thể xảy ra là 5, xác xuất từng viễn cảnh và ngưỡng T được nhập từ người thực hiện, penalty là 3 phút. Thực hiện lặp để cập nhật nhân tử lagrange và tính giá trị lower bound, upper bound với từng ngưỡng T nhất định, ta được bảng kết quả như sau:



Di tản từ **0** → **49**:

Time Thresholds: 2
Lower Bound: 9064.05
Upper Bound: 9093.4

Time Thresholds: 3
Lower Bound: 9454.87
Upper Bound: 9652.9

Time Thresholds: 4

Lower Bound: 9816.95
Upper Bound: 9893.4

Di tản từ **5** $\rightarrow$ **39**:

Time Thresholds: 2
Lower Bound: 6622.91
Upper Bound: 7059.6

Time Thresholds: 3
Lower Bound: 6621.41
Upper Bound: 7264.6

Time Thresholds: 4
Lower Bound: 7055.13
Upper Bound: 7346.6

Sau khi tính toán được giới hạn giá trị tối ưu, ta tìm được prior plan trước ngưỡng thời gian T như sau (di tản tờ 0 -> 49, Time thresholds = 3, S = 5) :

0 -> 1 : 74
0 -> 5 : 86
1 -> 2 : 67
1 -> 6 : 7
2 -> 3 : 0
2 -> 7 : 67
3 -> 4 : 0
3 -> 8 : 0
4 -> 9 : 0
5 -> 6 : 0
5 -> 10 : 86
6 -> 7 : 0
6 -> 11 : 7
7 -> 8 : 0
7 -> 12 : 67
8 -> 9 : 0
8 -> 13 : 0
9 -> 14 : 0
10 -> 11 : 84
10 -> 15 : 2
11 -> 12 : 0
11 -> 16 : 91
12 -> 13 : 67
12 -> 17 : 0
13 -> 14 : 67
13 -> 18 : 0
14 -> 19 : 67
15 -> 16 : 2
15 -> 20 : 0
16 -> 17 : 0
16 -> 21 : 93
17 -> 18 : 0
17 -> 22 : 0
18 -> 19 : 0
18 -> 23 : 0
19 -> 24 : 67
20 -> 21 : 0
20 -> 25 : 0
21 -> 22 : 0
21 -> 26 : 93
22 -> 23 : 0
22 -> 27 : 0
23 -> 24 : 0
23 -> 28 : 0
24 -> 29 : 67
25 -> 26 : 0
25 -> 30 : 0
26 -> 27 : 2
26 -> 31 : 91
27 -> 28 : 2
27 -> 32 : 0
28 -> 29 : 0
28 -> 33 : 2
29 -> 34 : 67
30 -> 31 : 0
30 -> 35 : 0
31 -> 32 : 16
31 -> 36 : 75
32 -> 33 : 16
32 -> 37 : 0
33 -> 34 : 18
33 -> 38 : 0
34 -> 39 : 85
35 -> 36 : 0
35 -> 40 : 0
36 -> 37 : 75
36 -> 41 : 0
37 -> 38 : 0
37 -> 42 : 75
38 -> 39 : 0
38 -> 43 : 0
39 -> 44 : 85
40 -> 41 : 0
40 -> 45 : 0
41 -> 42 : 0
41 -> 46 : 0
42 -> 43 : 75
42 -> 47 : 0
43 -> 44 : 5
43 -> 48 : 70
44 -> 49 : 90
45 -> 46 : 0
46 -> 47 : 0
47 -> 48 : 0
48 -> 49 : 70

5 Tiến độ hoàn thành

Tuần	Nhiệm vụ	Tiến độ
1	- Nghiên cứu Python và GAMSPy	30%
	- Tìm hiểu và phân tích cơ sở lý thuyết Stochastic Programming	20%
2	- Nghiên cứu Python và GAMSPy	60%
	- Tìm hiểu và phân tích cơ sở lý thuyết Stochastic Programming	40%
	- Tìm hiểu và phân tích bài toán 1	30%
	- Tìm hiểu và phân tích bài toán 2	20%
3	- Nghiên cứu Python và GAMSPy	100%
	- Tìm hiểu và phân tích cơ sở lý thuyết Stochastic Programming	70%
	- Tìm hiểu và phân tích bài toán 1	50%
	- Coding bài toán 1	20%
	- Tìm hiểu và phân tích bài toán 2	40%
4	- Tìm hiểu và phân tích cơ sở lý thuyết Stochastic Programming	100%
	- Tìm hiểu và phân tích bài toán 1	80%
	- Coding bài toán 1	50%
	- Tìm hiểu và phân tích bài toán 2	50%
	- Viết báo cáo	10%
5	- Coding bài toán 1	80%
	- Tìm hiểu và phân tích bài toán 2	70%
	- Coding bài toán 2	50%
	- Viết báo cáo	70%
6	- Tìm hiểu và phân tích bài toán 2	100%
	- Coding bài toán 1	100%
	- Coding bài toán 2	100%
	- Viết báo cáo	100%

References

[1] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on stochastic programming: modeling and theory*. SIAM, 2021.

[2] S. W. Wallace and W. T. Ziemba, *Applications of stochastic programming*. SIAM, 2005.

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

[4] Bolin Ding, Jeffrey Xu Yu and Dorothea Wagner, *Finding Time-Dependent Shortest Paths over Large Graphs*, 2008

[5] Marshall L.Fisher, "The Lagrangian Relaxation Method for Solving Integer Programming Problems", *Management Science*, Vol.50, 2004.

[6] Stephen Boyd, Lin Xiao, and Almir Mutapcic, "Subgradient Methods", *Notes for EE392o, Stanford University*, 2003