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

MÔ HÌNH HÓA TOÁN HỌC (CO2011)

Báo cáo bài tập lớn

Lập trình ngẫu nhiên và Ứng dụng

GVHD: ThS. Mai Xuân Toàn
Sinh viên: Hồ Nguyễn Phi Hùng - 2211327.
Hồ Tuấn Linh - 2211848.
Trần Đăng Khoa - 2211648.
Dương Hoàng Long - 2211873.
Đặng Hữu Nhật Hoàng - 2211069.

TP.Hồ Chí Minh, Tháng 12 2023

Mục lục

1 Danh sách thành viên & Công việc

No.	Họ và tên	MSSV	Công việc	Phần trăm đóng góp
1	Hồ Nguyễn Phi Hùng	2211327	- Tìm hiểu vấn đề 1 và viết chương trình. - Soạn thảo Latex.	20%
2	Hồ Tuấn Linh	2211848	- Tìm hiểu vấn đề 2 và giải quyết. - Soạn thảo Latex.	20%
3	Trần Đăng Khoa	2211648	- Tổng hợp kiến thức nền. - Soạn thảo Latex.	20%
4	Dương Hoàng Long	2211873	- Tìm hiểu vấn đề 1 và viết chương trình. - Soạn thảo Latex.	20%
5	Đặng Hữu Nhật Hoàng	2211069	- Tìm hiểu vấn đề 2 và giải quyết. - Soạn thảo Latex.	20%

2 Kiến thức nền

2.1 Lập trình ngẫu nhiên - Khái niệm và mục tiêu

2.1.1 Lập trình? Lập trình ngẫu nhiên là gì? Bất định là gì?

Lập trình toán học là một tập hợp các phương pháp tính toán và toán học được sử dụng bởi các kỹ sư, nhà nghiên cứu, quản lý, nhà khoa học để giải quyết và đưa ra quyết định về các bài toán tối ưu hóa.

Tối ưu hóa toán học liên quan đến việc đưa ra quyết định thông qua các phương pháp toán học.

Lập trình ngẫu nhiên (SP) là một phần của lập trình toán học, nơi mà quyết định được đưa ra trong điều kiện bất định. Nói cách khác, SP có thể được coi là "Lập trình toán học (Tối ưu hóa) với các tham số ngẫu nhiên".

Trong **lập trình tuyến tính ngẫu nhiên**, một số dữ liệu của bài toán có thể được coi là bất định. Điều này có nghĩa là hàm mục tiêu là tuyến tính, nhưng một số dữ liệu của bài toán không cố định.

Sự ngẫu nhiên là không thể tránh khỏi trong việc xây dựng mô hình, từ dữ liệu quan sát được đến mô hình đề xuất. Sự ngẫu nhiên hoặc bất định có thể xuất phát từ nhiều nguồn khác nhau, chẳng hạn như thời tiết hoặc bất định tài chính. Thông thường, sự ngẫu nhiên được bỏ qua hoặc được xử lý bằng phân tích độ nhạy. Tuy nhiên, đối với các bài toán quy mô lớn, phân tích độ nhạy là vô ích. SP là cách để xử lý sự ngẫu nhiên, cả trong hàm mục tiêu và các ràng buộc.

Các bài toán thực tế trong Khoa học, Kỹ thuật và Công nghệ không được mô hình hóa như là ngẫu nhiên hay xác định. Các kỹ sư và nhà khoa học quyết định xem có nên mô hình hóa hiện tượng là ngẫu nhiên hay xác định dựa trên bài toán cần giải quyết. Trong các mô hình xác định, đầu ra của mô hình hoàn toàn được xác định bởi các giá trị tham số và các điều kiện ban đầu. Ngược lại, một mô hình ngẫu nhiên là một công cụ cho phép sự biến động ngẫu nhiên của một hoặc nhiều đầu vào theo thời gian.

Cuối cùng, **các bài toán có phản hồi (Stochastic)** là những bài toán trong đó một số quyết định hoặc hành động phản hồi (chỉnh sửa, sửa đổi) có thể được thực hiện sau khi bất định được tiết lộ.

2.1.2 Khái niệm cơ bản, giả định - Động lực

Định nghĩa: Chương trình tuyến tính (LP) với tham số ngẫu nhiên - SLP)
Một chương trình tuyến tính ngẫu nhiên (SLP) được định nghĩa như sau:

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

Trong đó, $x = (x_1, x_2, \ldots, x_n)$ là các biến quyết định, ma trận thực A và vector b là các ràng buộc xác định, và tham số ngẫu nhiên T, h trong $Tx \geq h$ xác định ràng buộc ngẫu nhiên hoặc xác suất.

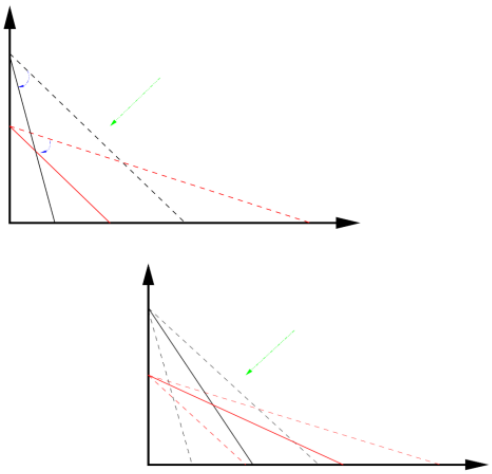
VÍ DỤ 1 (Động lực đơn giản đầu tiên)
Chúng ta xem xét việc tối ưu hóa sau:

$$\text{Minimize } z = x_1 + x_2, \text{ với điều kiện } x_1 \geq 0,\ x_2 \geq 0.$$

$$\begin{cases} \omega_1 x_1 + x_2 \geq 7; \\ \omega_2 x_1 + x_2 \geq 4. \end{cases}$$

Ở đây, tham số ω_1, ω_2 là các biến ngẫu nhiên tuân theo phân phối Uniform(a, b), cụ thể là $\omega_1 \sim \text{Uniform}(1, 4)$, $\omega_2 \sim \text{Uniform}(1/3, 1)$.

Khi cả $\omega_1 = \omega_2 = 1$ thì hai điều kiện trở thành $x_1 + x_2 = 4$ tạo ra đường đỏ, và $x_1 + x_2 = 7$ tạo ra đường nét đứt màu xanh lam, bạn rõ ràng nhận được vùng khả thi chứa đầy mũi tên màu xanh lá (Hình 1).



Hình 1: SP đơn giản với hai biến quyết định

Để giải quyết vấn đề khi ω_1, ω_2 thực sự là biến ngẫu nhiên, chúng ta có thể áp dụng hai phương pháp chính:

1. Phương pháp chờ và xem: Trong phương pháp này, chúng ta giả định rằng có thể quyết định về các biến quyết định $x = [x_1, x_2]$ sau khi quan sát vector ngẫu nhiên $\omega = [\omega_1, \omega_2]$. Điều này cho phép chúng ta giải quyết vấn đề mà không cần chờ đợi.

2. Phương pháp không chờ đợi: Trong phương pháp này, chúng ta giải quyết vấn đề mà không cần chờ đợi, tức là chúng ta cần quyết định về $x = [x_1, x_2]$ trước khi biết giá trị của $\omega = [\omega_1, \omega_2]$. Điều này đòi hỏi chúng ta phải quyết định phải làm gì với việc không biết ω.

Chúng tôi đề xuất hai cách tiếp cận để giải quyết sự không chắc chắn:

(a) Đoán về sự không chắc chắn: Chúng tôi sẽ đoán một số giá trị hợp lý cho ω. Ba đề xuất hợp lý – mỗi đề xuất cho chúng ta biết một điều gì đó về mức độ 'rủi ro' của chúng ta:

- Không thiên vị: Chọn giá trị trung bình cho mỗi tham số ngẫu nhiên ω
- Bi quan: Chọn giá trị tồi tệ nhất cho ω
- Lạc quan: Chọn giá trị tốt nhất cho ω.

Ví dụ, sử dụng phương pháp không thiên vị, phân phối đồng nhất Uniform(a, b) rõ ràng có trung bình $(a + b)/2$, vì vậy $\hat{\omega} = (5/2, 2/3)$. Chương trình của chúng tôi

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

có giá trị tối ưu $z_1 = 50/11$ tại điểm $\hat{x} = (\hat{x}_1, \hat{x}_2) = 18/11, 32/11$

nếu tuân theo

$$\begin{cases} 5/2x_1 + x_2 \geq 7; \\ 2/3x_1 + x_2 \geq 4; \\ x_1 \geq 0, x_2 \geq 0. \end{cases}$$

Các giải pháp Bi quan của chương trình Tối thiểu hóa $z = x_1 + x_2$ là:

Bi quan với $\hat{\omega} = [1, 1/3]$

$$\begin{cases} 1x_1 + x_2 \geq 7; \ 1/3x_1 + x_2 \geq 4 \\ x_1 \geq 0, \ x_2 \geq 0 \end{cases}$$

$\Rightarrow z_2 = 7$ tại điểm $\hat{x} = (\hat{x}_1, \hat{x}_2) = (0, 7)$

2.2 Bài toán tuyến tính ngẫn nhiên một giai đoạn - Không có hành động bù đắp

Chúng ta bắt đầu với bài toán tuyến tính ngẫu nhiên một giai đoạn, hay 1-SLP, nghĩa là bài toán tuyến tính ngẫu nhiên không có hành động bù đắp hoặc cụ thể là không có chi phí cho các hành động sửa chữa.

Định nghĩa: Bài toán tuyến tính ngẫu nhiên một giai đoạn - Không có hành động bù đắp: 1-SLP
Xét bài toán sau đây $LP(\alpha)$ có tham số là vector ngẫu nhiên α:

$$\text{Minimize } Z = g(x) = f(x) = c^T \cdot x = \sum_{j=1}^{n} c_j x_j$$

với ràng buộc

$$Ax = b, \text{ (ràng buộc xác định)}$$
$$\text{và } Tx \geq h \text{ (ràng buộc ngẫu nhiên)}$$

với các giả định sau:

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

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

2. Giá trị (T, h) không biết trước: chúng không được biết trước khi một phiên bản của mô hình xảy ra, $h(\alpha)$ chỉ phụ thuộc vào các tham số ngẫu nhiên α_j.

3. **Sự không chắc chắn** được biểu diễn bằng phân phối xác suất của các tham số ngẫu nhiên $(\alpha_j) = \alpha$ nên bài toán tuyến tính xác định là trường hợp đặc biệt của bài toán tuyến tính ngẫu nhiên khi α_j là hằng số.

Chúng ta phải giải quyết các bài toán quyết định khi vector $x = (x_1, x_2, \ldots, x_n) \in \mathcal{X}$ của các biến quyết định phải được đưa ra trước khi thực hiện của vector ngẫu nhiên $\alpha \in \Omega$ được biết. Thường chúng ta đặt các giới hạn dưới và trên cho x qua miền $\mathcal{X} = \{x \in \mathbb{R}^n : l \leq x \leq u\}$.

CÁCH TIẾP CẬN: Trong bài toán tuyến tính ngẫu nhiên, chúng ta sử dụng một số giả định và sự thật.

- **Giả định cơ bản** - Chúng ta biết một (phân phối xác suất chung) của dữ liệu. Do đó, cách tiếp cận đầu tiên là sử dụng ràng buộc xác suất (Chance constraint) LP.
- **Phân tích kịch bản**- không hoàn hảo, nhưng hữu ích, là cách tiếp cận thứ hai. Phân tích kịch bản giả định rằng có một số hữu hạn các quyết định mà tự nhiên có thể đưa ra (kết quả của sự ngẫu nhiên). Mỗi quyết định có thể này được gọi là một kịch bản.

2.2.1 PHƯƠNG PHÁP 1: Sử dụng ràng buộc cơ hội và rủi ro chấp nhận được

Chúng ta có thể thay thế $Tx \geq h$ bằng ràng buộc xác suất $P[Tx \geq h] \geq p$ cho một mức độ tin cậy được quy định trước $p \in (0.5, 1)$, (phải được xác định bởi người sở hữu bài toán.) Chương trình LP trong Định nghĩa ở trên với các tham số ngẫu nhiên $\alpha = [\alpha_1, \alpha_2, \ldots]$ thì được gọi là **Chương trình LP Ràng buộc Xác suất, hoặc chỉ gọi tắt là 1-SLP.**

Rủi ro thì được xử lý một cách rõ ràng, nếu định nghĩa một

$$\textit{rủi ro chấp nhận được } r_x := P[\text{Not}(Tx \geq h)] = P[Tx \leq h] \leq 1 - p$$

thì $(1 - p)$ là rủi ro tối đa có thể chấp nhận được.

Ràng buộc cơ hội $Tx \leq h$ ngụ ý rằng rủi ro chấp nhận được r_x nhỏ hơn một giá trị tối đa được xác định trước $1 - p \in (0, 1)$.

Định nghĩa: Bài toán tuyến tính ngẫu nhiên hoặc 1-SLP với Ràng buộc Xác suất được xác định bởi hệ số ngẫu nhiên $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ trong ràng buộc cơ hội, và mục tiêu tuyến tính $f(x)$:

$$\text{SP} : \min_x Z,\ Z = f(x) = c^T \cdot x = \sum_{j=1}^{n} c_j x_j,\ c_j \in \mathbb{R}$$

s.t

$$Ax = b \text{ (x = (x1, x2, ..., xn) tạo ra các biến quyết định)}$$

$$P[Tx = \alpha \cdot x \leq h] \leq (1 - p)\ (0 < \text{p} < 1).$$

LƯU Ý: chúng ta sử dụng vector tham số $\alpha = [\alpha_1, \alpha_2, \ldots]$ nói chung, và ký hiệu $\omega = [\omega_1, \omega_2, \ldots, \omega_S]$ cụ thể cho các trạng thái s được gọi là kịch bản. Chúng ta xử lý từng kịch bản $\omega \in \Omega$ có thể bằng cách kết hợp nhiều tham số ngẫu nhiên α_i cùng một lúc trong một SP.

2.2.2 PHƯƠNG PHÁP 2: Đối với các ràng buộc ngẫu nhiên $T(\alpha)x \leq h(\alpha)$

Đối với các ràng buộc ngẫu nhiên $T(\alpha)x \leq h(\alpha)$ là sử dụng Phân tích Kịch bản. Đối với mỗi kịch bản $(T_s; h_s)$, $s = 1, \ldots, S$, ta giải quyết bài toán:

$$\text{Minimize } \{f(x) = c^T \cdot x; \quad \text{với điều kiện } Ax = b,\ T^s x \leq h^s\}$$

Loại chương trình này nhắm đến một mục tiêu tuyến tính cụ thể trong khi tính đến một hàm xác suất liên quan đến các kịch bản khác nhau. Do đó, chúng ta tìm ra một giải pháp tổng thể bằng cách xem xét các giải pháp kịch bản x^s $(s = 1, \ldots, S)$.

Ưu điểm của phương pháp này là mỗi vấn đề kịch bản là một LP. Tuy nhiên, nhược điểm là phân phối rời rạc dẫn đến mô hình LP số nguyên hỗn hợp (nói chung, có thể là mô hình không lồi).

VÍ DỤ 2: Xem xét LP sau đây với $n = 4$ biến quyết định $x = (x_1, x_2, x_3, x_4)'$

$$P_0 : \min Z = g(x) = -x_2$$

và các ràng buộc ngẫu nhiên dưới dạng $T(\alpha)x \leq h(\alpha) \Leftrightarrow T(\alpha)x + W = h(\alpha)$.

Chúng ta có thể bắt đầu với phân phối rời rạc của các tham số $\alpha = (\alpha_1, \alpha_2)$, thoả mãn $\alpha_1 = 1$, $\alpha_2 = 3/4$ với mật độ 0.25; và $\alpha_1 = -3$, $\alpha_2 = 5/4$ với mật độ 0.75.

Ở đây với α ngẫu nhiên $= (\alpha_1, \alpha_2)$ chúng ta chỉ có $S = 2$ kịch bản $\omega_1 = (\alpha_1 = 1, \alpha_2 = 3/4)$ với mật độ 0.25; $\omega_2 = (\alpha_1 = -3, \alpha_2 = 5/4)$ với mật độ 0.75.

Đặt $T(\alpha) = [\alpha_1, \alpha_2, 0, 1]$, $h(\alpha) = 2 + \alpha_1$, và để $x_1 \geq 2$, và $x_j \geq 0$ cho, $j \in \{1, 2, 3, 4\}$ chương trình cụ thể của chúng ta là

$$\begin{cases} x_1 + x_2 + x_3 = 3 \text{ (như các ràng buộc chắc chắn)} \\ \alpha_1 x_1 + \alpha_2 x_2 + x_4 = 2 + \alpha_1 \text{ (ràng buộc ngẫu nhiên)} \end{cases}$$

Điều này có nghĩa là có hai phương trình có thể (hoặc các kịch bản ω_1, ω_2): hoặc ràng buộc thứ hai là $x_1 + 3/4x_2 + x_4 = 3$ (kịch bản xảy ra với xác suất 0.25), hoặc nó là $-3x_1 + 5/4x_2 + x_4 = -1$ (kịch bản xảy ra với xác suất 0.75).

2.3 Giới thiệu về Vấn đề Xác suất Tổng quát (GSP) với Hành động Bù đắp.

GSP là một tập hợp các quyết định được thực hiện mà không có đầy đủ thông tin về một số sự kiện ngẫu nhiên. Những quyết định này được gọi là quyết định giai đoạn đầu và thường được biểu diễn bằng một vector $\mathbf{x}$. Sau khi nhận được thông tin đầy đủ về sự thực hiện của một số vector ngẫu nhiên α, các hành động sửa chữa hoặc bù đắp $\mathbf{y}$ được thực hiện. Chúng tôi biểu diễn các quyết định giai đoạn thứ hai bằng vector $\mathbf{y}$.

Chúng tôi viết vấn đề 2-SP cho chương trình xác suất hai giai đoạn cho phép không thỏa mãn với các ràng buộc ngẫu nhiên $\mathbf{T} \cdot \mathbf{x} \leq \mathbf{h}$, $\mathbf{T} \cdot \mathbf{x} + \mathbf{W} \cdot \mathbf{y} = \mathbf{h}$ hoặc tương đương $\mathbf{W} \cdot \mathbf{y} = \mathbf{h} - \mathbf{T} \cdot \mathbf{x}$. Chúng tôi sửa chữa những bất thỏa mãn sau này và phải trả tiền cho các hành động sửa chữa qua $\mathbf{y}$. Cả α và $\mathbf{y}$ có thể sử dụng các hàm số, như $\alpha(\omega)$, $\mathbf{y}(\omega)$ hoặc $\mathbf{y}(s)$, để thể hiện sự phụ thuộc rõ ràng vào một kịch bản ω hoặc một kết quả s của thí nghiệm ngẫu nhiên e kết hợp với việc mô hình hóa 2-SP.

Định nghĩa: Chương trình xác suất hai giai đoạn (vấn đề 2-SP tổng quát) Chương trình xác suất hai giai đoạn (2-SP) mở rộng từ Định nghĩa trên có dạng:

$$2 \text{ - } SP : \min_{\mathbf{x}} g(\mathbf{x}) \text{với } g(\mathbf{x}) = f(\mathbf{x}) + E_\omega[v(\mathbf{x}, \omega)]$$

trong đó $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ là các biến quyết định giai đoạn đầu, $f(\mathbf{x})$ có thể là tuyến tính hoặc không, một phần của hàm mục tiêu tổng quát $g(\mathbf{x})$.

Giá trị trung bình $Q(\mathbf{x}) := E_\omega[v(\mathbf{x}, \omega)]$ của một hàm:

$$v : \mathbb{R}^n \times \mathbb{R}^S \to \mathbb{R}$$

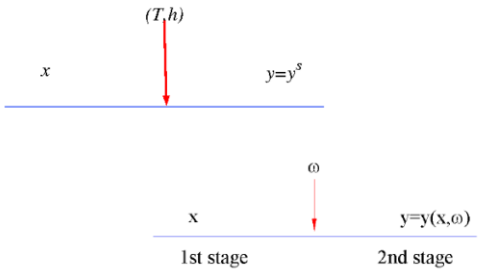
dựa trên ảnh hưởng của các kịch bản $\omega . Q(\mathbf{x})$ là giá trị tối ưu của một vấn đề giai đoạn thứ hai nhất định

$$\min_{\mathbf{y} \in \mathbb{R}^p} q\mathbf{y} \mid \text{phụ thuộc vào } \mathbf{T} \cdot \mathbf{x} + \mathbf{W} \cdot \mathbf{y} = \mathbf{h}.$$

Các vector $\alpha = \alpha(\omega)$ và $\mathbf{y} = \mathbf{y}(\omega)$ được gọi là các biến quyết định sửa chữa, điều chỉnh hoặc bù đắp, chỉ được biết sau khi thực hiện thí nghiệm ngẫu nhiên e.

Chúng tôi tối thiểu hóa tổng chi phí kỳ vọng $g(\mathbf{x}) = f(\mathbf{x}) + Q(\mathbf{x})$ trong khi thỏa mãn

$$\mathbf{W} \cdot \mathbf{y}(\omega) = \mathbf{h}(\omega) - \mathbf{T}(\omega) \cdot \mathbf{x}.$$



Hình 2: Chế độ xem tiêu chuẩn của chương trình ngẫu nhiên hai giai đoạn Được phép của Maarten van der Vlerk, Univ. của Groningen, NL

Ở đây $\mathbf{W}$ được gọi là ma trận bù đắp $m \times p$, và chúng tôi bắt đầu với trường hợp đơn giản của $m = 1$, $\mathbf{q}$ là vector chi phí bù đắp đơn vị, có cùng kích thước với $\mathbf{y}$, và $\mathbf{y} = \mathbf{y}(\omega) \in \mathbb{R}^p$.

2.4 Lập trình tuyến tính ngẫu nhiên 2 giai đoạn

Bài toán 2-stage stochastic linear programming là một mô hình quyết định tối ưu đối với các vấn đề quyết định trong môi trường không chắc chắn. Trong thực tế, có nhiều quyết định phải được đưa ra dựa trên thông tin không chắc chắn, và mô hình này là một công cụ quan trọng để giải quyết những thách thức này.

Ở giai đoạn đầu tiên, bài toán yêu cầu quyết định về việc phân bổ tài nguyên hoặc lập kế hoạch sản xuất dựa trên thông tin có sẵn tại thời điểm đó. Tuy nhiên, thông tin này thường không chắc chắn và có thể biến đổi tại giai đoạn thứ hai. Ở giai đoạn này, khi thông tin thêm vào trở nên rõ ràng hơn, quyết định cần được điều chỉnh để đảm bảo tối ưu trong bối cảnh mới.

Bài toán này thường được mô tả bằng các biểu diễn toán học, sử dụng các hàm mục tiêu và ràng buộc tương ứng. Sự kết hợp giữa quyết định ở giai đoạn đầu và giai đoạn sau cùng với sự không chắc chắn trong thông tin tạo nên một mô hình phức tạp và đòi hỏi sự linh hoạt trong quyết định.

Bài toán 2-stage stochastic linear programming đưa ra một cách tiếp cận toán học để giải quyết vấn đề quyết định trong bối cảnh không chắc chắn, giúp tối ưu hóa hiệu suất và đồng thời duy trì sự linh hoạt cần thiết khi đối mặt với sự biến động của thông tin thực tế. Dưới đây là một ví dụ đơn giản về 2-stage stochastic linear programing

Hàm mục tiêu:

$$\min_{x \in X} c^T \cdot x + \min_{y(\omega) \in Y} \mathbf{E}_\omega [q \cdot y]$$

subject to:

$$Ax = b \qquad \text{First Stage Constraints}$$

$$T(\omega) \cdot x + W \cdot y(\omega) = h(\omega) \quad \text{Second Stage Constraints}$$

$$\text{or Shortly} \qquad W \cdot y = h(\omega) - T(\omega) \cdot x$$

Chương trình SLP này chỉ định 2-SP(2) ở trên cho mục tiêu - một mục tiêu (hàm) ngẫu nhiên lớn cụ thể g(x) có :

(1) f(x) - tất định là hàm tuyến tính, trong khi tính toán .

(2) đối với hàm xác xuất v(X,ω) liên quan đến các kịch bản khác nhau ω .

y = y(X,ω) $\in$ Rp+ được đặt tên là biến hành động truy đòi đối với quyết định x và thực hiện ω . Các hành động truy đòi được xem là hành động khắc phục bị phạt trong SLP.

Hiệu chỉnh Phạt được thể hiện thông qua giá trị trung bình $Q(X) = E\omega[v(x,\omega)]$. Làm thế nào để tìm hiểu được nó các phương pháp tiếp cận chính - PHƯƠNG PHÁP 2 : phân tích lại kịch bản.

Để giải bài toán 2-stage stochastic linear programming, các phương pháp tiếp cận dựa trên một vectơ ngẫu nhiên a có số lượng hữu hạn các khả năng thực hiện , được gọi là kịch bản.

Giá trị kì vọng Q(x) hiển nhiên đối với phân bố rời rạc của ω! .

Vì vậy, chúng ta lấy $\Omega = \omega k$ là một tập hữu hạn có kích thước S (có một số lượng hữu hạn kịch bản $\omega 1$, ... , $\omega S \in \Omega$, với khối lượng xác xuất tương ứng p_k).

Vì $y = y(x,\omega)$ nên kì vọng của $v(y) = v(x,\omega) := q \cdot y$ (một chi phí q cho tất cả p_k) là :

$$Q(x) = E_\omega[v(X,\omega)] = \sum_{k=1}^{S} p_k \cdot q \cdot y_k = \sum_{k=1}^{S} p_k \cdot v(X,\omega_k)$$

Với

p_k là xác xuất của scenario ω_k , q là giá cho mỗi đơn vị.

và q $y_k = v(X,\omega_k)$ - chi phí tăng thêm bởi việc sử dụng y_k trong giai đoạn hiệu chỉnh, phụ thuộc vào cả quyết định x ở giai đoạn đầu và các kịch bản ngẫu nhiên ω_k.

3 Vấn đề 1

3.1 Giới thiệu vấn đề

Xét một hãng công nghiệp F sản xuất n sản phẩm. Nhãn hàng có M bộ phận khác nhau cấu thành sản phẩm được cung cấp từ bên thứ 3. Một đơn vị sản phẩm thứ i yêu cầu $a_{ij} \geq 0$ với $i = 1, \ldots, n$ và $j = 1, \ldots, m$. Nhu cầu về sản phẩm được mô hình hóa dưới dạng vector ngẫu nhiên $\omega = D = (D_1, D_2, \ldots D_n)$.

The second-stage problem:

Đối với một giá trị khả thi ($d = (d_1, d_2, \ldots, d_m)$ của vector nhu cầu ngẫu nhiên D ở trên, chúng ta có thể tìm ra được phương án sản xuất tốt nhất cho hãng công nghiệp bằng cách giải phương trình stochastic linear program(SLP) dưới đây :

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

Trong đó ta có :

$z = (z_1, z_2, \ldots, z_n)$ là số lượng sản phẩm được sản xuất

$y = (y_1, y_2, \ldots, y_m)$ là số lượng linh kiện của từng bộ phận

$s_j < b_j$(được định nghĩa là chi phí đặt hàng trước trên mỗi đơn vị của phần j) và x_j với $j = 1, \ldots, m$ là số lượng chi tiết cần đặt trước khi sản xuất

phụ thuộc vào

$$\begin{cases} y_j & = x_j - \sum_{i=1}^{n} a_{ij} z_i, \quad j = 1, \ldots, m \\ 0 & \leq z_i \leq d_i, \quad i = 1, \ldots, n; \quad y_j \geq 0, \quad j = 1, \ldots, m. \end{cases}$$

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

MODEL:

$$\begin{cases} \min_{z,y} & Z = c^T z - s^T y \\ \text{với} & c = (c_i := l_i - q_i) \text{ là hệ số chi phí} \\ & y = x - A^T z, \text{ trong đó } A = [a_{ij}] \text{ là ma trận kích thước } n \times m, \\ & 0 \leq z \leq d, \quad y \geq 0. \end{cases} \tag{1}$$

Ta thấy rằng lời giải của bài toán này, tức là các vector z,y phụ thuộc vào việc thực hiện d của nhu cầu ngẫu nhiên $\omega = D$ cũng như là phụ thuộc vào bước 1 với $x = (x_1.x_2, \ldots, x_m)$

The first-stage problem:

Toàn bộ mô hình này dựa trên một nguyên tắc phổ biến đó là **nhu cầu ≥ sản xuất**.

Ta đặt $Q(x) := E[Z(x, y)] = E_\omega[x, \omega]$ biểu thị giá trị tối ưu của bài toán. Vậy ta cần chứng tỏ rằng $b = (b_1, b_2, \ldots, b_m)$ được xây dựng dựa theo chi phí đặt hàng trước khi sản xuất trên mỗi đơn vị của phần thứ j(đã nêu ở trên). Đại lượng x_j được xác định từ bài toán tối ưu hóa sau.

$$min_{g(x,y,z)} = b^T \cdot x + Q(x) = b^T \cdot x + E[Z(z)] \tag{2}$$

Trong đó $Q(x) = E_\omega[Z] = \sum_{j=1}^{n} (p_j \cdot c_j \cdot z_j)$ được phép lấy w.r.t với phân bố của xác xuất $\omega = D$.

Phần đầu tiên của hàm mục tiêu biểu thị chi phí đặt hàng trước X và ngược lại thứ hai thể hiện chi phí dự kiến của kế hoạch tối ưu(model) được đưa ra bởi cách cập nhật số lượng khách hàng đã đặt hàng trước z với việc đã sử dụng nhu cầu ngẫu nhiên $D = d$ với mật độ của chúng.

3.2 Yêu cầu bài toán

VẤN ĐỀ: [sản xuất n sản phẩm thỏa mãn sản xuất ≤ nhu cầu]

Sử dụng mô hình 2-SLPWR được đưa ra trong Công thức (1) và (2) khi $n = 8$ sản phẩm, số lượng kịch bản $S = 2$ với mật độ $p_s = 1/2$, số lượng phần cần đặt trước khi sản xuất $m = 5$, chúng ta mô phỏng ngẫu nhiên dữ liệu vector **b, l, q, s** và ma trận A có kích thước $n \times m$. Chúng ta cũng giả định rằng vector nhu cầu ngẫu nhiên $\omega = D = (D1, D2, , Dn)$ nơi mỗi ω_i với mật độ p_i tuân theo phân phối nhị thức $Bin(10, 1/2)$.

YÊU CẦU: xây dựng các mô hình số học của Công thức (1) và (2) với dữ liệu được mô phỏng. Tìm nghiệm tối ưu $x, y \in \mathrm{R}^m$, và $z \in \mathrm{R}^n$ bằng phần mềm phù hợp (như GAMSPy).

3.3 Giải quyết bài toán

3.3.1 Xây dựng mô hình biểu thức số học

Bài toán trên đang mô tả một kịch bản nhu cầu hữu hạn d_1, d_2 xảy ra với xác suất dương p_1, p_2, trong đó $p_1 + p_2 = 1$. Do đó, vấn đề hai giai đoạn (1)–(2) có thể được viết lại dưới dạng một vấn đề lập kế hoạch tuyến tính quy mô lớn như sau:

Chúng ta muốn tối thiểu hóa hàm mục tiêu:

$$min_{x,y,z} = b^T \cdot x + \sum_{k=1}^{2} p_k[(l-q)^T \cdot z_k - s^T \cdot y_k] \tag{3}$$

Với các ràng buộc như sau:

$$y_k = x - A^T \cdot z_k, \quad k = 1, 2,$$
$$0 \le z_k \le d_k, \quad y_k \ge 0, \quad k = 1, 2,$$
$$x \ge 0$$

Trong đó, việc tối thiểu hóa được thực hiện qua các biến vector x và z_k, y_k, $k = 1, 2$. Chúng ta đã tích hợp vấn đề giai đoạn thứ hai (1) vào công thức này, nhưng chúng ta đã phải cho phép giải pháp (z_k, y_k) của nó phụ thuộc vào kịch bản k, vì thực hiện đơn vị ngẫu nhiên d_k khác nhau trong mỗi kịch bản. Do đó, vấn đề (3) có số lượng biến và ràng buộc độc lập với số lượng kịch bản K.

3.3.2 Thiết kế chương trình giải nghiệm tối ưu

Khởi tạo các tham số ban đầu: Chúng ta cần có các tham số b, l, q, s, D và ma trận A để cho chương trình có thể giải được các nghiệm tối ưu. Vì trong bài này các tham số này là ngẫu nhiên nên chúng ta sẽ sử dụng thư viện **numpy** trong Python để tạo các tham số này một các ngẫu nhiên. Chúng ta cũng sẽ khai báo các tham số n, m, S, p_s có săn trong phần này

Vì để bài toán có tính thực tế nên chúng ta sẽ thêm 1 số điều kiện cho các tham số (chúng ta có thể bỏ qua những điều kiện này nếu không cần thiết).

```
import numpy as np

# Initialize parameters
n = 8 # Number of products
S = 2 # Number of scenarios
```

```
p_s = 1/2 # Density
m = 5 # Number of parts to be ordered before production

# Create matrix A with random positive integer values from 1 to 10
A = np.random.randint(1, 11, size=(n, m))
# Create random demand vector D following Binomial distribution Bin(10, 1/2)
D = np.random.binomial(10, 1/2, (S, n))
# Create vector b with random values in the range from 0 to 50
b = np.random.uniform(0, 50, size=m)
# Create vector l with random values in the range from 60 to 1200
l = np.random.uniform(100, 2400, size=n)
# Create vector q with random values in the range from 1000 to 10000
q = np.random.uniform(1000, 10000, size=n)
# Create vector s with random values less than b
s = np.array([np.random.uniform(0, b_j, size=1)[0] for b_j in b])
```

Xây dựng mô hình: Để tìm nghiệm tối ưu cho bài toán này ta có thể sử dụng các công cụ như GAMS, Gamspy, ... Trong bài này chúng ta sẽ sử dụng thư viện **cvxpy** trong Python để giải nghiệm bài toán. Đây là đoạn code xây dựng mô hình tối ưu:

```
import cvxpy as cp
# Initialize variables
x = cp.Variable(m, integer=True)
y = cp.Variable((S, m), integer=True)
z = cp.Variable((S, n), integer=True)

# Calculate c
c = l - q

# Construct the objective function
objective = cp.Minimize(b.T @ x + p_s * (cp.sum(c.T @ z[1, :]) - cp.sum(s.T @ y[1, :]))
    + p_s * (cp.sum(c.T @ z[0, :]) - cp.sum(s.T @ y[0, :])))

# Construct constraints
constraints = [y[s, :] == x - A.T @ z[s, :] for s in range(S)]
constraints += [z[s, :] >= 0 for s in range(S)]
constraints += [z[s, :] <= D[s, :] for s in range(S)]
constraints += [y[s, :] >= 0 for s in range(S)]

# Solve the optimization problem
prob = cp.Problem(objective, constraints)
prob.solve(solver=cp.CBC)
```

3.3.3 Một số ví dụ về giải nghiệm tối ưu bằng chương trình

Dưới đây là một số kết quả nhận được sau khi chạy chương trình tìm nghiệm tối ưu được thiết kế ở mục 3.3.2

```
b =  [37.54941482 16.47775494 10.39734888 13.91386655 23.15948897]
l =  [ 741.68129288 1831.80861714 2145.61588134 1121.76450949 1342.54030574
  362.12289754 2387.417785   2253.20930606]
q =  [3851.32611854 3348.86900857 3998.97758746 1236.12997797 4398.42756949
 5628.9523247  3628.30668492 6362.87578693]
s =  [31.42990563  7.72725274  8.92044397  5.13799098  0.29710861]
D =
 [[6 5 5 7 8 5 3 3]
 [3 7 5 5 4 4 1 3]]
A =
 [[ 3 10  2 10  5]
 [ 7  4  6  1  5]
 [ 2  6  9  5  3]
 [ 8  8  5  5  1]
 [ 8  6  3  8  7]
 [ 3  4  4  8  5]
 [ 4  4  3  5  1]
 [ 8 10  5  5  1]]

Giá trị tối ưu: -71635.1148760033
Giải pháp x: [178. 220. 155. 224. 157.]
Giải pháp y:
 [[-0. -0. -0. -0. -0.]
 [38. 58. 16. 78. 40.]]
Giải pháp z:
 [[ 6.  5.  5. -0.  8.  5.  3.  3.]
 [ 3.  7.  5. -0.  4.  4.  1.  3.]]
```

Hình 3: Kết quả 1

```
b =  [48.50965947 19.72286351 48.04499035 28.97584463 12.31647138]
l =  [1667.89621332  151.77756507 2199.24163215 1188.66588296 1101.69316715
 1244.41373139 2001.08643496 1743.12368178]
q =  [7957.55558081 1993.83700796 8280.87346447 6800.03770654 5613.20499735
 9979.58494883 1983.4438089  7216.77959227]
s =  [13.34464528  0.37048382 39.66720789 19.46057292  5.29102597]
D =
 [[ 5  2  7  5 10  5  3  2]
 [ 2  7  3  4  4  4  7  3]]
A =
 [[10  5  7  1  7]
 [ 1  6  8  4 10]
 [10  3 10 10  4]
 [ 8  6  7  7  4]
 [ 4  9  5  3  9]
 [ 5 10  3  7  2]
 [ 2  7  9  7  8]
 [ 5 10  3  2  6]]

Giá trị tối ưu: -137606.89849647693
Giải pháp x: [237. 248. 227. 187. 215.]
Giải pháp y:
 [[-0. -0. -0. -0. -0.]
 [97. 57. 58. 53. 41.]]
Giải pháp z:
 [[ 5.  2.  7.  5. 10.  5. -0.  2.]
 [ 2.  7.  3.  4.  4.  4. -0.  3.]]
```

Hình 4: Kết quả 2

4 Vấn đề 2

4.1 Giới thiệu vấn đề

Đề tài đề cập đến giải pháp tối ưu để sơ tán người dân bị ảnh hưởng bởi thảm họa một cách sớm nhất (động đất, sóng thần, khủng bố...) nhằm giảm thiểu thiệt hại và thương vong. Mục đích của bài báo này nhằm giải quyết vấn đề lập kế hoạch sơ tán cho người dân bằng cách xây dựng cho một khung mô hình chung khi xảy ra thảm họa. Đề xuất mô hình lập trình ngẫu nhiên hai giai đoạn, xét cả 2 lựa chọn: đường đi ưu tiên(trước thảm họa) và đường đi thích ứng(sau thảm họa) để đưa ra kế hoạch sơ tán cho những người bị ảnh hưởng bởi thảm họa từ khu vực nguy hiểm đến khu vực an toàn. Nhưng không giống như các mô hình xác định, nó liên quan đến thời gian di chuyển và dung lượng của liên kết ngẫu nhiên. Cụ thể như, một phần đường giao thông có thể bị phá hủy trong thời gian xay ra thảm họa, gây ra tình trạn ùn tắc.

=> Do đó, bài báo này đề cập đến việc lập kế hoạch sơ tán ưu tiến cho những người bị ảnh hưởng với các yếu tố không chắc chắn được đề cập ở trên, và:

- Để xây dựng một quy trình sơ tán rõ ràng của những người dân bị ảnh hưởng bởi thảm họa, bài báo này đã đề xuất mô hình luồng chi phí tối thiểu (min-cost flow) dựa trên lập trình ngẫu nhiên hai giai đoạn. Hơn nữa, mô hình này được phân tách thành hai băi toán con bằng phương pháp nhân tử Lagrangin (Lagrangin relaxtion). Hai bài toán con tương ứng là luồng chi phí tối thiểu và trường hợp phụ thuộc vào thời gian của nó. Do đó bài toán đề xuất được chia thành hai bài toán con và có thể xử lý được.
- Để có được giải pháp tối ưu gần đúng, khung thuật toán tối ưu hóa Subgradient được áp dụng để giảm dần khoảng cách giữa giới hạn trên và giới hạn dưới, tức là các giá trị mục tiêu của mô hình ban đầu và mô hình nhân tử; và hai bài toán con được giải bằng thuật toán đường đi ngắn nhất liên tiếp. Kết quả bằng số cho thấy mô hình được đề xuất trong bài viết này vượt trội hơn cả mô hình xác định và mô hình wait and see. Chứng minh mô hình mô hình sơ tán ngẫu nhiên hai giai đoạn cung cấp một công cụ ra quyết định thực tế để sơ tán người dân bị ảnh hưởng trong khu vực thảm họa.

4.2 Mô hình sơ tán ngẫu nhiên 2 giai đoạn

Cho đồ thị có hướng (mạng) **G** với **m** đỉnh và **n** cạnh. Mỗi cạnh được cho bởi dung lượng u_{ij} biểu thị lượng tối đa có thể thông qua (số nguyên không âm) và chi phí hay thời gian di chuyển trên cung (i,j) c_{ij}.

Với giá trị **v** cho trước, tiến hành tìm tất cả các luồng thỏa giá trị **v**, trong số chúng chọn ra luồng có chi phí thấp nhất. Bài toán này được gọi là **luồng chi phí tối thiểu**. Nhưng trong **Algorithm 1** có một số khác biệt nhỏ: chúng ta muốn tìm ra đường đi ngắn với luồng cực đại (để có thể sơ tán nhiều người nhất có thể) nhưng cũng cần phải đảm bảo chi phí phải thấp nhất, được gọi là **luồng cực đại với chi phí tối thiểu**.

Để tìm được đường đi ngắn nhất ta có nhiều thuật toán với độ phức tạp khác nhau, dựa theo Algorithm 1 ta sẽ chọn thuật toán **Bellman-Ford** để xác định luồng có chi phí tối thiểu.

Vấn đề sơ tán được quan tâm là có được kế hoạch chắc chắn trong giai đoạn đầu bằng đánh giá của kế hoạch thích ứng trong giai đoạn hai. Để đạt được mục đích này, bài viết xây dựng biểu tối ưu: mức phạt đối với kế hoạch sơ tán ưu tiên và thời gian sơ tán tổng thể dự kiến của kế hoạch sơ tán thích ứng theo từng kịch bản, và xác xuất xảy ra của từng kịch bản được giả định là:

$$\mu_s,\ s = 1,\ 2,\ \cdots, S.$$

Xây dựng một kế hoạch sơ tán chi tiết cho người dân từ khu vực nguy hiểm đến nơi an toàn như một min cost flow problem với thời gian và dung lượng di chuyển của các liên kết là ngẫu nhiên. Mục tiêu bài toán là thời gian di tản tối thiểu trong một mạng lưới chi phí công suất:

$$\mathbf{G(V,A,C,U,D)}$$

Với:

- V là tập hợp các nút
- A là tập hợp các liên kết với thời gian di chuyển và dung lượng di chuyển ngẫu nhiên
- c(i,j) là thười gian di chuyển trên liên kết(mắt xích) $(i,j) \in A$
- u(i,j) là dung lượng liên kết (i,j)
- d(i) là luồng tồn tại nút i thuộc V

First-stage: Thời gian và dung lượng di chuyển của các liên kết được biết một cách xác suất nhưng những người di tản phải được sơ tán từ source đến các nút khác trước khi có thông tin về thời gian di chuyển và dung lượng ở second-stage.

Second-stage: Kế hoạch sẽ được xác định cùng với thời gian di chuyển và dung lượng thực tế. Lưu ý rằng, kế hoạch trong first-stage có thể không khả thi trong thực tế, vậy nên hàm mục tiêu sẽ bao gồm các chi phí phát sinh (phạt khi đi sai) ở first-stage và giá trị kỳ vọng của chi phí ở second-stage.

4.2.1 Mức phạt của kế hoạch ban đầu và thời gian sơ tán kì vòng của từng kịch bản phải được tối ưu:

$$min \sum_{(i,j)\in A} p_{ij}x_{ij} + \sum_{s=1}^{S} \mu_s . Q(Y,s)$$

Do giai đoạn 2 có số lượng kịch bản có hạn nên mô hình trên tương đương với:

$$min \sum_{(i,j)\in A} p_{ij}x_{ij} + \sum_{s=1}^{S} (\mu_s . \sum_{(i,j)\in A_s} c_{ij}^s(t).y_{ij}^s(t)$$

→ Ta có mô hình:

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

Kí hiệu:

- p_s : Xác suất kịch bản s
- $c_{ij}^s(t)$: Thời gian di chuyển của liên kết (i,j) trong kịch bản s ở thời gian t.
- T: Toongt số khoảng thời gian
- u_{ij} : Dung lượng ban đầu của liên kết vật lí (i,j)
- $u_{ij}^s(t)$: Dung lượng của liên kết (i,j) trong kịch bản s ở thời gian t.

Biến quyết định

- x_{ij} : Luồng liên kết (i,j)
- $y_{ij}^s(t)$: Luồng liên kết (i,j) trong kịch bản s ở thời gian t.

4.3 Thuật toán giải quyết vấn đề:

Trong mô hình này, ràng buộc liên kết là một ràng buộc phức tạp, dẫn đến mô hình không thể giải được trong thời gian đa thức. Vì vậy, phần này nhằm mục đích giãn rang buộc liên kết vào hàm mục tiêu bằng phương pháp nhân tử Lagrange. Dựa trên phương pháp này sẽ phân tách bài toán ban đầu thanh 2 bài toán con có giá trị mục tiêu là giới hạn dưới của giá trị tối ưu của mô hình. Cụ thể, mô hình ban đầu được phân tách thành bài toán luồng chi phí tối thiểu tiêu chuẩn và bài toán phụ thuộc vào thời gian.

Ràng buộc này đặc trưng cho mối quan hệ giữa việc lựa chọn 1 liên kết vật lí và các cung phụ thuộc vào thời gian dựa trên kịch bản tương ứng. Do đó, ta có nhân tử Lagrange $\alpha_{ij}^s, (i,j) \in A, s = 1,2,...,S, t \leq T$ cho ràng buộc liên kết này và sau đó ràng buộc này có thể được nới lỏng thành hàm mục tiêu:

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

Sau khi nơi lỏng ta có mô hình sau:

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

Bài toán con 1: Bài toán luồng chi phí tối thiểu

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

Hàm mục tiêu của bài toán con 1 có thể định nghĩa là $g_{ij} := p_{ij} - \sum_{s=1}^{s}\sum_{t\le T}\alpha_{ij}^{s}(t)$ để thể hiện cho phí chung của từng liên kết.

Bài toán con 2: Bài toán luồng chi phí tối thiểu phụ thuộc thời gian

$$\begin{cases} \min SP2(\alpha) = \sum\limits_{s=1}^{S}\sum\limits_{(i,j)\in A}\left(\sum\limits_{t\in\{0,1,\dots,T\}}\mu_s.c_{ij}^{s}(t) + \sum\limits_{t\le T}\alpha_{ij}^{s}(t)\right)y_{ij}^{s}(t) \\ \sum\limits_{(i_t,j_t)\in A_S} y_{ij}^{s}(t) - \sum\limits_{(j_{t'},i_t)\in A_S} y_{ij}^{s}(t') = d_i^{s}(t), \forall\, i \in V, \\ t \in \{0,1,\dots,T\}, s = 1,2,\dots,S \\ 0 \le y_{ij}^{s}(t) \le u_{ij}^{s}(t), \forall\,(i,j) \in A, t \in \{0,1,\dots,T\}, s = 1,2,..,S \end{cases}$$

Algoruthm 1: Thuật toán đường đi liên tục ngắn nhất cho bài toán luồng chi phí tối thiểu (Sucessive Shortest Path Algorithm)

Buoc 1: Lấy biến x như một luồng khả thi giữa bất kì nút thu nào và nó có mức chi phí tối thiểu phân phối trong các luồng khả thi có cùng giá trị luồng.

Buoc 2: Thuật toán sẽ kết thúc nếu giá trị luồng của x đạt đến v hoặc không có đường dẫn chi phí tối thiểu nào khác trong mạng lưới (V,A(x),C(x),U(x),D). Nếu không, đường ngắn nhất với luồng tối đa được tính bằng giải thuật label-correcting. Các hàm A(x),C(x),U(x) trong phần còn lại của mạng lưới được định nghĩa là:

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

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

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

Buoc 3: Tăng giá trị luồng dọc theo đường có chi phí tối thiểu. Nếu giá trị luồng sau khi tăng lên không vượt quá v, chuyển qua bước 2.

Vì khoảng thời gian được chia thành 2 giai đoạn thời gian nên chi phí tổng quát được xác định là:

$$g_{ij}^{s}(t) = \begin{cases} \mu_s . c_{ij}^{s}(t) + \alpha_{ij}^{s}(t), t \le \bar{T} \\ \mu_s . c_{ij}^{s}(t), \qquad \bar{T} < t \le T \end{cases}$$

Bằng cách giải 2 bài toán con, giá trị mục tiêu tối ưu Z_{LR}^{*} cho mô hình nới lỏng với nhân tử Lagrangian α có thể được biểu diễn dưới dạng:

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

Giá trị mục tiêu tối ưu của mô hình nới lỏng là giới hạn dưới của giá trị mục tiêu tối ưu của mô hình ban đầu. Để có được giải pháp chất lượng cao, cần đạt được giới hạn dưới lớn nhất có thể với giá trị mục tiêu tối ưu của mô hình ban đầu.

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

Thuật toán Bellman - Ford

Thuật toán Bellman–Ford là một thuật toán tính các đường đi ngắn nhất nguồn đơn trong một đồ thị có hướng có trọng số (trong đó một số cung có thể có trọng số âm).

Ý tưởng của thuật toán như sau:

- Ta thực hiện duyệt n lần, với n là số đỉnh của đồ thị.
- Với mỗi lần duyệt, ta tìm tất cả các cạnh mà đường đi qua cạnh đó sẽ rút ngắn đường đi ngắn nhất từ đỉnh gốc tới một đỉnh khác.
- Ở lần duyệt thứ n, nếu còn bất kỳ cạnh nào có thể rút ngắn đường đi, điều đó chứng tỏ đồ thị có chu trình âm, và ta kết thúc thuật toán.

Cách thức hoạt động của thuật toán Bellman-Ford như sau:

- Bước 1 : Khởi tạo khoảng cách từ điểm bắt đầu đến tất cả các điểm trong đồ thị là vô cùng trừ điểm bắt đầu là 0
- Bước 2: Duyệt qua từng cạnh trong đồ thị và cập nhật khoảng cách của các đỉnh. Cụ thể, với mỗi cạnh (u, v) có trọng số w, nếu khoảng cách hiện tại từ điểm bắt đầu đến u cộng với trọng số w nhỏ hơn khoảng cách hiện tại từ điểm bắt đầu đến v thì cập nhật khoảng cách của v bằng khoảng cách từ điểm bắt đầu đến u cộng với trọng số
- Bước 3: Kiểm tra xem có chu trình âm trong đồ thị hay không. Nếu có, thuật toán sẽ thông báo là không thể tìm đường đi ngắn nhất vì các cạnh có trọng số âm sẽ khiến cho đường đi ngắn nhất không tồn tại.

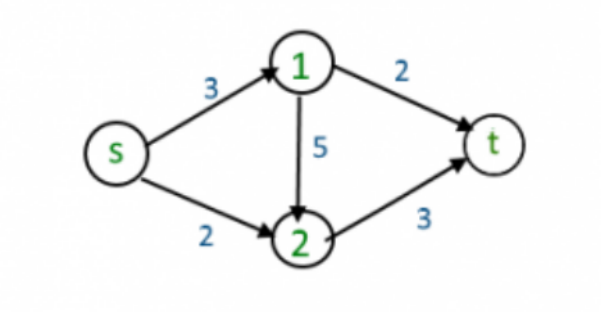
Nếu không có chu trình âm trong đồ thị, sau khi thực hiện bước 2, thuật toán Bellman-Ford sẽ trả về khoảng cách ngắn nhất từ điểm bắt đầu đến tất cả các điểm trong đồ thị

Thuật toán chạy trong thời gian $O(V \cdot E)$, trong đó V là số đỉnh và E là số cạnh của đồ thị (đây không phải là thuật toán hiệu quả nhất để tìm đường đi ngắn nhất trong đồ thị có hướng với trọng số cạnh âm).

Luồng cực đại: Từ nguồn s (có thể là siêu nguồn sau khi tổng hợp từ nhiều nguồn của đồ thị gốc) đến đích s(tương tự với nguồn), tìm ra luồng cực đại.

Trước tiên nhóm đưa ra một cách tiếp cận của thuật toán tham lam, đó là khởi đầu các luồng đều bằng 0, và có giá trị ngày càng cao lên sau đó.

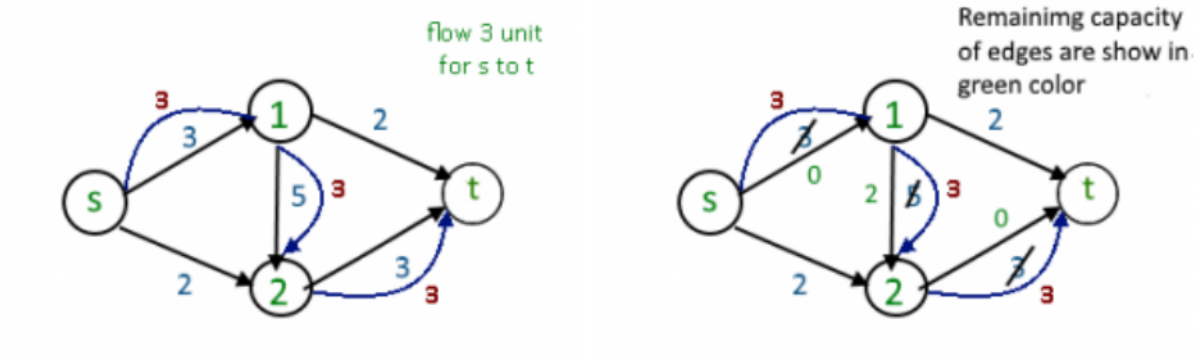
Ta xét 1 đồ thị cụ thể sau:



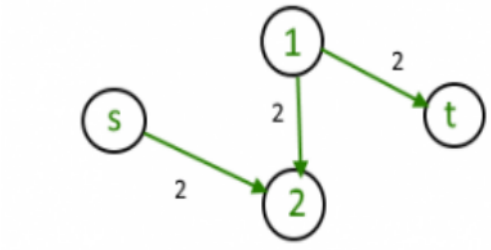
Gọi f(e) là luồng trên cạnh e, C(e) là dung lượng của cạnh đó

Việc tìm kiếm đường đi cần đảm bảo rằng f(e) < C(e)

Xem xét một ví dụ nhỏ trong hình, bằng giải thuật **Bellman-Ford** ta dễ dạng nhận thấy đường dẫn [s -> 1 -> 2 -> t] với luồng cực đại 3 đơn vị, tức là mang 3 đơn vị từ s -> t và đi qua các nút trung gian:



Sau đó tiến hành cập nhật lại mạng và bỏ đi các cạnh không cần thiết (có dung lượng = 0)



Từ đồ thị trên ta có thể kết luận rằng max flow của đồ thị = 3 vì không còn đường đi nào từ s -> t

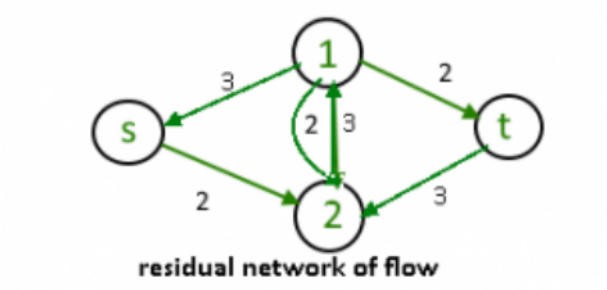
Tuy nhiên max flow của đồ thị ban đầu nên là 5. Để giải quyết vấn đề này chúng ta sử dụng biểu đồ dư

Ý tưởng là mở rộng thuật toán tham lam trên bằng cách thêm vào thao tác hoàn tác

Từ đồ thị G và một luồng f trong nó, chúng ta xây dựng một mạng luồng Gf có cùng tập đỉnh và đích với G và có hai cạnh ứng với mỗi cạnh trong G.

- Cạnh lùi: f(e) (lưu ý cạnh lùi có chi phí = -(chi phí của cạnh tiến))
- Cạnh tiến: C(e)-f(e)
- Các đường dẫn từ nguồn đến đích với f(e) < C(e) cho tất cả các cạnh.

Từ đó ta có thể cập nhật lại đồ thị dựa trên bổ sung trên:



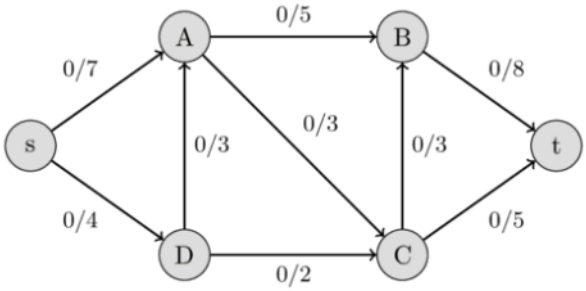
Tiếp tục tìm ra đường dẫn [s -> 2 -> 1 -> t] với luồng cực đại 2 đơn vị sau đó cập nhật lại đồ thị, lúc này đã không còn đường đi từ s->t kết luận **max flow = 5**

Thuật toán Ford-Fulkerson (Nhóm tự tìm hiểu để so sánh với giải thuật 1)

Phương pháp Ford-Fulkerson hoạt động như sau. Đầu tiên, chúng ta đặt luồng của mỗi cạnh về 0. Sau đó, chúng tôi tìm kiếm một đường dẫn tăng cường từ s đến t. Đường dẫn tăng cường là một đường dẫn đơn giản trong biểu đồ dư, tức là dọc theo các cạnh có dung lượng dư là dương. Nếu tìm thấy đường dẫn như vậy thì chúng ta có thể tăng luồng dọc theo các cạnh này. Tiếp tục tìm kiếm các đường dẫn tăng cường và tăng luồng. Khi đường tăng cường không còn tồn tại nữa thì luồng sẽ đạt cực đại.

Nói rõ hơn về tăng luồng, gọi C là công suất dư nhỏ nhất của cạnh trên đường dẫn. Chúng ta tăng luồng bằng cách cập nhật f(i,j)+=c và f(j,i)-=c với f(i,j) là luồng từ i đến j

Dưới đây là một ví dụ để chứng minh phương pháp. Giá trị đầu tiên của mỗi cạnh biểu thị luồng, ban đầu là 0 và giá trị thứ hai biểu thị dung lượng.

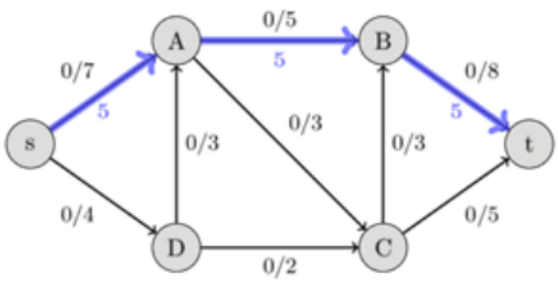


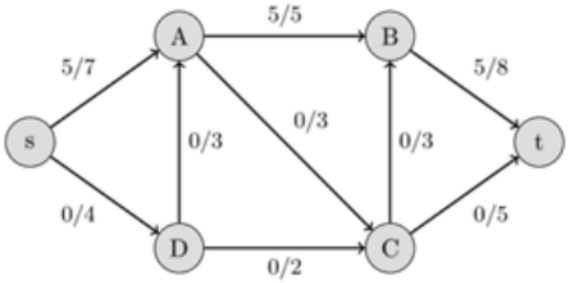
Tìm luồng cực đại là luồng có giá trị lớn nhất có thể qua, sau đó tăng luồng trên các cạnh

Khởi đầu, luồng bằng 0, tiếp

Chúng ta có thể tìm thấy con đường s - A - B - t với dung lượng dư 7, 5 và 8. Mức tối thiểu của chúng là 5, do đó chúng ta có thể tăng luồng dọc theo đường dẫn này lên 5.

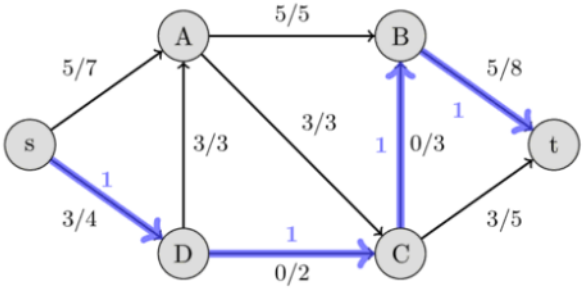
Max flow = 5

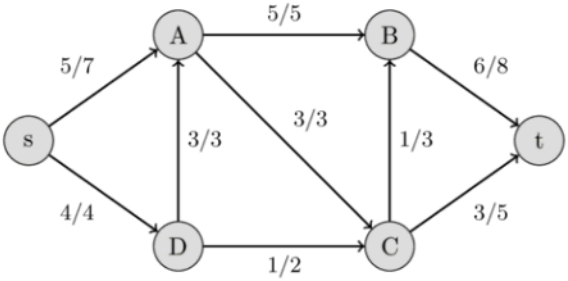




Một lần nữa chúng ta tìm kiếm một đường dẫn tăng cường, lần này chúng ta tìm thấy đường dẫn s - D - A - C - t với dung lượng còn lại là 4, 3, 3 và 5. Do đó, chúng ta có thể tăng lưu lượng lên 3

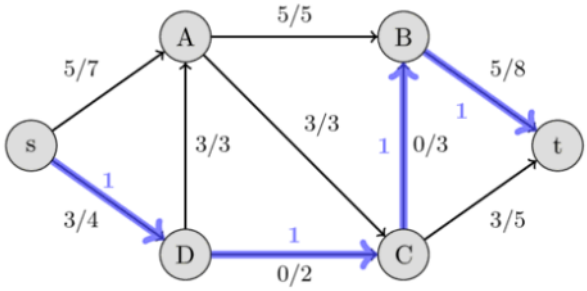
Max flow = 5 + 3

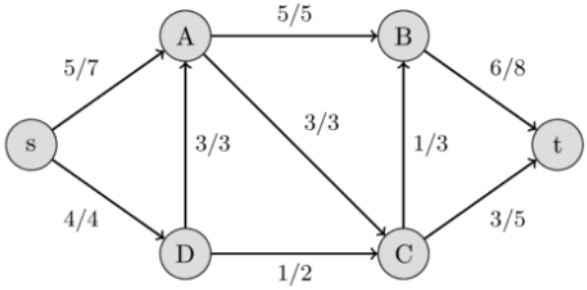




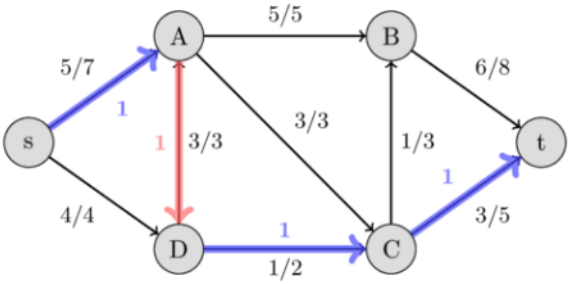
Lần nữa, ta tìm thấy đường s - D - C - B - t với công suất dư 1, 2, 3 và 3, do đó, chúng ta tăng lưu lượng lên 1.

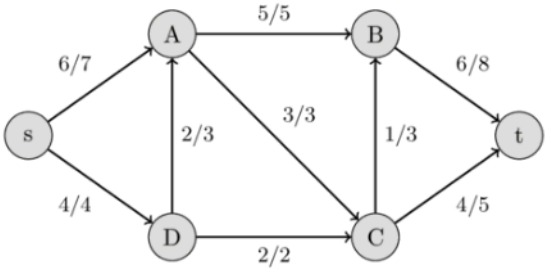
Max flow = 5 + 3 + 1 = 9





Có vẻ như không thể tăng luồng lên nữa tại bước này. Tuy nhiên nếu xem xét kĩ ta nhận thấy rằng, cạnh (A,D). thay vì truyền đi 3 đơn vị từ A đến D thì chỉ truyền đi 2 và bù đắp điều này bằng cách tăng thêm luồng 1 từ s đến A, điều này cho phép ta thêm luồng 1 dọc theo đường dẫn D –C – t. Từ đó hình thành nên đường tổng quát từ đích đến nguồn là s- A- D- C- t, mặc dù như ta đã thấy không có bất kỳ đường dẫn nào từ A đến D.





y giờ, không thể tìm thấy một con đường tăng cường giữa, do đó **Max flow = 10**

4.3.1 Thuật toán Cyclle Canceling

- là một phương pháp được sử dụng để giải quyết vấn đề dòng chảy có chi phí tối thiểu trong một mạng lưới
- Thuật toán Cycle Canceling duy trì một giải pháp khả thi trong mạng lưới và tiếp tục bằng cách tăng dòng chảy dọc theo các chu trình có chi phí âm trong mạng lưới dư, do đó hủy chúng. Thuật toán này là một trong những thuật toán đầu tiên để giải quyết vấn đề dòng chảy có chi phí tối thiểu.
- Đối với vấn đề dòng chảy có chi phí tối thiểu với dữ liệu nguyên, phiên bản tổng quát của thuật toán hủy chu trình chạy trong thời gian giả đa thức, nhưng có thể có được một số cài đặt cụ thể trong thời gian đa thức bằng cách chỉ định các lựa chọn của các chu trình để hủy.

Mô tả về thuật toán:

1. Tạo một đồ thị với các nút và cạnh, nơi mỗi cạnh có một công suất và một chi phí.
2. Chạy thuật toán, duy trì một giải pháp khả thi trong mạng lưới.
3. Thuật toán tiếp tục bằng cách xác định các chu trình có chi phí âm trong mạng lưới dư.
4. Sau đó, nó tăng dòng chảy tối đa có thể trong các chu trình này, do đó hủy chúng.
5. Thuật toán kết thúc khi mạng lưới dư không chứa chu trình có chi phí âm nào.

So sánh

Ford-Fulkerson, Cancel-and-Tighten (còn được gọi là Canceling Cycle), và Successive Shortest Path đều là các thuật toán được sử dụng để giải quyết các vấn đề về dòng chảy trong đồ thị. so sánh giữa các thuật toán:

- Thuật toán Ford-Fulkerson: Thuật toán này được sử dụng để tìm dòng chảy tối đa trong một mạng. Nó bắt đầu với một dòng chảy ban đầu là 0 và trong khi có một đường đi từ nguồn đến đích trong đồ thị dư, nó tìm một đường đi mở rộng và tăng dòng chảy dọc theo đường đi này bằng công suất dư tối thiểu của các cạnh dọc theo đường đi. Độ phức tạp thời gian của thuật toán Ford-Fulkerson là O(max flow * E) nơi E là số lượng cạnh. Độ phức tạp không gian là O(E + V) nơi V là số lượng đỉnh.
- Thuật toán Cancel-and-Tighten:. Thuật toán này thường được sử dụng giải quyết các vấn đề về dòng chảy với chi phí tối thiểu, nơi nó lặp lại hủy các chu trình âm trong đồ thị dư cho đến khi không còn chu trình nào tồn tại.
- Thuật toán Successive Shortest Path: Thuật toán này được sử dụng để giải quyết vấn đề về dòng chảy với chi phí tối thiểu. Nó tính toán các đường đi ngắn nhất giữa các nút nguồn và đích với chi phí cung (thông qua thuật toán Dijkstra chẳng hạn) trong khi tất cả dòng chảy chưa được gửi. Độ phức tạp thời gian và không gian của thuật toán này có thể thay đổi tùy thuộc vào cách triển khai cụ thể và các thuộc tính của đồ thị đầu vào

Lưu ý rằng độ phức tạp thời gian và không gian của các thuật toán này có thể thay đổi tùy thuộc vào các đặc điểm cụ thể của đồ thị và cách triển khai chính xác của thuật toán. Ngoài ra, các độ phức tạp này là giới hạn trên lý thuyết và thời gian thực tế và sử dụng không gian có thể ít hơn trong thực tế. Cũng quan trọng là lưu ý rằng các thuật toán này được thiết kế để giải quyết các loại vấn đề khác nhau (dòng chảy tối đa so với dòng chảy với chi phí tối thiểu), vì vậy chúng có thể không được so sánh trực tiếp trong tất cả các trường hợp.

Tài liệu tham khảo

[1] Alexander Shapiro, Darinka Dentcheva, Andrzej Ruszczynski - Lectures on Stochastic Programming Modeling and Theory-SIAM-Society for Industrial and Applied Mathematic

[2] MM Assignment Stochastic Programming and Applications 2023

[3] Li Wang - A two-stage stochastic programming framework for evacuation planning in disaster responses