Vietnam National University, Ho Chi Minh City
University of Technology
Faculty of Computer Science and Engineering

MATHEMATICAL MODELING (CO2011)

Assignment (Semester: 231, Duration: 06 weeks) "Stochastic Programming and Applications" (Version 0.1, in Preparation)

Instructor(s):

Nguyễn Tiến Thịnh, CSE-HCMUT

Hồ Chí Minh City, September 2023

Vietnam National University, Ho Chi Minh City
University of Technology
Faculty of Computer Science and Engineering

MATHEMATICAL MODELING
(CO2011)

Member list & Wordload

ID	*Member*	*Contribute*	*Point*
2212345	*Hồ Thanh Nhã*	20%	
2212338	*Vũ Trần Duy Nguyên*	20%	
2211093	*Nguyễn Huy Hoàng*	20%	
2211091	*Nguyễn Huy Hoàng*	20%	
2212971	*Huỳnh Đức Tài*	20%	

Hồ Chí Minh City, September 2023

Mục lục

1 Kiến thức nền

1.1 Giới thiệu về Stochastic Programming và Optimization

1.1.1 Programing ? What is Stochastic Programming ?

Định nghĩa 1. ***Mô hình Stochastic Programming*** *liên quan đến hai loại biến: biến quyết định (decision variables) và biến ngẫu nhiên (random variables). Biến quyết định đại diện cho các lựa chọn mà người quyết định có thể kiểm soát: như mức sản xuất (production levels), phân bổ đầu tư (investment allocations), hoặc mức tồn kho (inventory levels). Biến ngẫu nhiên đại diện cho các yếu tố không chắc chắn ảnh hưởng đến kết quả của vấn đề, như nhu cầu, giá cả, hoặc thời tiết.*

Mục đích sử dụng mô hình:
Tìm ra các giá trị tối ưu cho các biến quyết định nhằm giảm thiểu hoặc tối đa hóa một tiêu chí cụ thể, như chi phí, lợi nhuận, hoặc rủi ro, đồng thời xem xét các kịch bản có thể xảy ra của các biến ngẫu nhiên.Stochastic Programming đã được áp dụng rộng rãi trong nhiều lĩnh vực, từ tài chính đến vận tải và tối ưu hóa năng lượng

1.1.2 Stochastic Linear Program (SLP)

Định nghĩa 2. ***Stochastic linear programming*** *là một bài toán tối ưu mà trong đó*

- *Hàm mục tiêu và các ràng buộc phải ở dạng tuyến tính*
- *Một vài hoặc tất cả các tham số là yếu tố không chắc chắn nhưng biết được phân phối xác suất của chúng*

Stochastic linear program (SLP) là sự tối ưu mà:
Minimize $Z = g(\mathbf{x}) = f(\mathbf{x}) = \mathbf{c}^T \cdot \mathbf{x}$, mà trong đó $A\mathbf{x} = \mathbf{b}$,và $T\mathbf{x} \geq h$
với $\mathbf{x} = (x1, x2, \cdots, x_n)$ (biến quyết định),
một ma trận xác định A và vector $\mathbf{b}$ (nhằm ràng buộc xác định - deterministic constraints), với các biến ngẫu nhiên T, h mà $T\mathbf{x} \geq h$ xác định ràng buộc khả năng hay xác suất (chance or probabilistic constraints).

1.2 One-Stage Stochastic linear Programming - No recourse(1-SLP)

Định nghĩa 3. *Xét sự tối ưu bên dưới LP(α) mà đã được tham số hóa hóa bởi vector ngẫu nghiên α*

$$\begin{aligned} &\textit{Minimize } Z = g(\boldsymbol{x}) = f(\boldsymbol{x}) = \boldsymbol{c}^T \cdot \boldsymbol{x} = \textstyle\sum_{j=1}^{n} c_j x_j \\ &\textit{mà trong đó } A\boldsymbol{x} = \boldsymbol{b} \textit{ (certain constraints)} \\ &\qquad\qquad T\boldsymbol{x} \geq h \textit{ (stochastic constraints)} \end{aligned}$$

với giả định rằng:

1. *ma trận $T = T(\alpha)$ và vector $h = h(\alpha)$ thể hiện sự không chắc chắn thông qua stochastic constraints*

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

2. *giá trị (T, h) là không xác định: chúng không xác định trước khi một thức thể (instance) của mô hình xuất hiện, $h(\alpha)$ phụ thuộc vào duy nhất biến ngẫu nhiên $alpha_j$;*

3. *sự không chắc chắn được biểu hiện bởi sự phân phối xác suất (probability distribution) của các tham số ngẫu nhiên $(\alpha_j) = \alpha$ vì thê deterministic LP là một trường hợp suy biến của Stochastic LP khi α_j là một hằng số,*

   - *Chúng ta giải quyết vấn đề quyết định (decision problems) mà ở đó vector $x = (x_1, x_2, \cdots, x_n) \in \chi$ của biến quyết định phải được khởi tạo trước khi hiện thực hóa vector tham số $\alpha \in \Omega$ xác định được.*

   - *Thông thường ta đặt giới hạn trên và giới hạn dưới cho x thông qua một miền $\chi = x \in \mathbb{R}^n : l \leq x \leq u$.*

APPROACHES: Trong Stochastic Programming ta tối ưu một số giả định và sự thật.

Fundamental assumption- Chúng ta biết một phân phối xác suất (chung) của dữ liệu. Vì vậy cách tiếp cận đầu tiên tạo ra ProProbabilistic (Chance) Constraint LP.

The Scenario Analysis- không hoàn hảo, nhưng hữu dụng, với cách tiếp cận thứ hai. The Scenario Analysis giả định rằng có hữu hạn sự quyết định mà tự nhiên có thể tạo (kết quả của sự ngẫu nhiên). Mỗi sự quyết định khả thi được gọi là **scenario**.

APPROACH 1 : use Chance constraint and Acceptable risk

- Chúng ta có thể thay thế $Tx \geq h$ với probabilistic constraints $\mathbb{P}[Tx \geq h] \geq p^2$ với mức độ đáng tin cậy $p \in (.5, 1)$, (được đưa ra bởi người đưa ra vấn đề.)
  LP trong Định nghĩa (??) ở trên với các tham số ngẫu nhiên $\alpha = [\alpha_1, \alpha_2, \cdots]$ nên được gọi là **Probabilistic Constraint LP** hay **1-SLP.**

- Rủi ro sau đó sẽ được quan sát một cách kĩ lưỡng nếu định nghĩa được:

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

  khi đó $(1 - p)$ là rủi ro tối đa chấp nhận được (**maximal acceptable risk.**)
  Ràng buộc khả năng (chance constraint) $Tx \leq h$ ngụ ý rằng
  rủi ro chấp nhận được r_x bé hơn giá trị tối đa $1 - p \in (0, 1)$.

Định nghĩa 4. *Stochastic LP hoặc 1-SLP với Probabilistic Constraints được định nghĩa bởi các hệ số ngẫu nhiên $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ với các ràng buộc khả năng,*
và một mục tiêu tuyến tính (linear objective) $f(x)$:

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

$$\textit{sao cho} \begin{cases} Ax = b(x = (x_1, x_2, \cdots, x_n) \textit{ tạo ra các biến quyết định}) \\ \mathbb{P}[Tx = \alpha \cdot x \leq h] \leq (1-p)(0 < p < 1). \end{cases}$$

NOTE: ta sử dụng tham số vector $\alpha = [\alpha_1, \alpha_2, \cdots]$ nói chung và kí hiệu $\omega = [\omega_1, \omega_2, \cdots, \omega_S]$ cho mỗi trạng thái s được gọi là scenarios. Chúng ta có thể xử lý scenario $\omega \in \omega$ bằng cách kết hợp nhiều tham số ngẫu nhiên α_i cùng một lúc trong một SP.

APPROACH 2 : dùng cho stochastic constraints $T(\alpha)x \leq h(\alpha)$

Sử dụng phận tích Scenario của $T(\alpha)x \leq h(\alpha)$
Với mỗi scenario $(T^s; h^s), s = 1, ..., S$, giải

$$\text{Minimize } f(x) = c^T \cdot x; \textit{sao cho } Ax = b, T^s x \leq h^s$$

Loại chương trình này nhắm đến một mục tiêu tuyến tính (linear objective) cụ thể trong khi tính đến một hàm xác suất (probability function) liên quan đến các scenario khác nhau. Do đó, chúng ta tìm ra một giải pháp tổng thể bằng cách xem xét các giải pháp cho từng scenario $x^s (s = 1, ..., S)$.
Ưu điểm: Mỗi vấn đề scenario là một LP.
Nhược điểm: phân bố rời rạc $\longrightarrow$ mô hình mixed-integer LP (mô hình có thể lồi).

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

Chương trình tuyến tính ngẫu nhiên hai giai đoạn với Recourse (2-SLPWR), hoặc chính xác hơn là với biện pháp penalize corrective, thường được mô tả như sau

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

hay tổng quan hơn

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

với $v(x, \omega) := q \cdot y$
sao cho

$$\begin{aligned} Ax = b \text{ First Stage Constraints,} \\ T(\omega) \cdot x + W \cdot y(\omega) = h(\omega) \text{ Second Stage Constraints} \\ \text{hay ngắn gọn hơn } W \cdot y = h(\omega) - T(\omega) \cdot x \end{aligned}$$

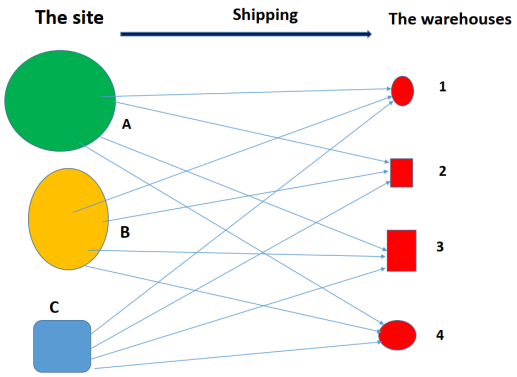
Two-Stage Stochastic linear programming (2-SLP) chỉ định (2-SP) mục tiêu - a specific random grand objective(function) g(x)

1. Xác định $f(x)$ là hàm tuyến tính trong khi tính toán
2. đối với hàm xác suất $v(x, \omega)$ liên quan đến các kịch bản khác nhau ω .

2 Question 1

2.1 Giới thiệu vấn đề

Xem xét một doanh nghiệp công nghiệp F nơi một nhà sản xuất sản xuất n sản phẩm. Có các bộ phận khác nhau (các bộ phận con) cần được đặt hàng từ tổng cộng m nhà cung cấp bên thứ ba (các địa điểm). Bức tranh này mô tả một kế hoạch vận chuyển của doanh nghiệp công nghiệp F với m = 3 từ các nhà cung cấp và n = 4 địa điểm sản xuất.



Một đơn vị sản phẩm i cần $a_{ij} \geq 0$ đơn vị của bộ phận j, trong đó $i = 1, \ldots, n$ và $j = 1, \ldots, m$.Nhu cầu cho những sản phẩm đó được mô hình hóa như một vector ngẫu nhiên $\omega = D = (D_1, D_2, \ldots, D_n)$.

The second-stage problem

Đối với một giá trị quan sát cụ thể của nhu cầu ngẫu nhiên $d = (d_1, d_2, \ldots, d_n)$, ta cần xác định kế hoạch sản xuất tối ưu bằng cách giải quyết bài toán tuyến tính ngẫu nhiên (SLP) với biến quyết định $z = (z_1, z_2, \ldots, z_n)$ - số lượng đơn vị sản phẩm được sản xuất và những biến quyết định khác $y = (y_1, y_2, \ldots, y_n$ - số lượng phần còn tồn tại trong kho

2.2 Giải thích sơ lược về bài toán

Bài toán trên mô tả vấn đề lập kế hoạch sản xuất hai giai đoạn cho một công ty công nghiệp (được ký hiệu là F) sản xuất nhiều sản phẩm sử dụng các bộ phận từ các nhà cung cấp bên thứ ba. Bài toán này liên quan đến các quyết định liên quan đến lập kế hoạch sản xuất và đặt hàng các bộ phận

1. Tổng quan:

- Sản phẩm và nhà cung cấp: Công ty công nghiệp sản xuất n sản phẩm, và có m nhà cung cấp bên thứ ba từ đó đặt mua các nguyên liệu khác nhau.
- Kế hoạch vận chuyển: Kế hoạch vận chuyển liên quan đến việc vận chuyển các nguyên liệu từ nhà cung cấp đến các địa điểm sản xuất

2. Kế hoạch sản xuất:

- Bài toán được chia thành hai giai đoạn: giai đoạn thứ nhất và giai đoạn thứ hai.

- Giai đoạn thứ nhất (first-stage) đưa ra quyết định về việc đặt hàng trước (x) trước khi biết đến nhu cầu thực tế.
- Giai đoạn thứ hai (second-stage) xảy ra sau khi nhu cầu thực tế được biết và đưa ra quyết định sản xuất cụ thể (z, y).

3. Hàm mục tiêu:

- Giai đoạn thứ nhất cố gắng tối thiểu hóa chi phí đặt hàng trước và chi phí kỳ vọng của quyết định sản xuất.
- Hàm mục tiêu của giai đoạn thứ nhất là tổng của chi phí đặt hàng trước và kỳ vọng chi phí sản xuất.

4. Ràng buộc:

- Các ràng buộc dựa trên mối quan hệ giữa các bộ phận và sản phẩm, giới hạn sản xuất của mỗi sản phẩm và chi phí đặt hàng trước.

5. Quyết định dựa trên nhu cầu thực tế;

- Sau khi nhu cầu thực tế được biết, giai đoạn thứ hai quyết định cụ thể về việc sản xuất bao nhiêu sản phẩm (z) và giữ lại bao nhiêu bộ phận trong kho (y).

6.Mục đích:

- Mô hình tối ưu hóa này giúp công ty đưa ra kế hoạch sản xuất và đặt hàng hiệu quả, giảm thiểu chi phí và tối ưu hóa sản xuất dựa trên xác suất của nhu cầu thực tế.

2.3 Phân tích đề

Bài toán trên mô tả vấn đề lập kế hoạch sản xuất hai giai đoạn cho một công ty công nghiệp (được ký hiệu là F) sản xuất nhiều sản phẩm sử dụng các bộ phận từ các nhà cung cấp bên thứ ba. Bài toán này liên quan đến các quyết định liên quan đến lập kế hoạch sản xuất và đặt hàng các bộ phận

Phương pháp liệt kê các ngữ cảnh(scenarious)

- Bước 1: Xác định hàm mục tiêu
- Bước 2: Xác đánh các ràng buộc
- Bước 3: Xác định các ngữ cảnh(scenarious) có thể xảy ra với từng ngữ cảnh , xác định các giá trị cho các tham số(yếu tố) không chắc chắn và xác suất xảy ra ngữ cảnh(scenarios) đó
- Bước 4: Giải bài toán tối ưu ứng với từng biến cố. Tìm ra nghiệm tối ưu cho từng bài toán đó.
- Bước 5: Tính giá trị cuối cùng của nghiệm tối ưu(kết quả của bài toán) là giá trị kì vọng của nghiệm tối ưu

Bài toán: Xác định số lượng sản xuất đáp ứng nhu cầu thị trường để tối ưu lợi nhuận thu được

Trong đó:

- m: Số đơn vị nguyên liệu cần đặt hàng trước khi sản xuất
- n: Số lượng loại sản phẩm
- $z \in \mathbb{R}^n$: Số lượng sản phẩm tạo ra cho từng loại
- $x \in \mathbb{R}^m$: Số lượng đơn vị nguyên liệu order về để sản xuất ra sản phẩm
- $A \in \mathbb{R}^{(n \times m)}$: Ma trận phụ thuộc giữa số sản phẩm cần sản suất và số nguyên liệu cần dùng a_{ij}: số nguyên liệu loại j $\rightarrow$ cần để sản xuất ra 1 đơn vị sản phẩm loại i
- $b \in \mathbb{R}^m$: Chi phí để mua 1 đơn vị nguyên liệu
- $l \in \mathbb{R}^n$: Chi phí sản xuất ra 1 sản phẩm
- $q \in \mathbb{R}^n$: Giá bán 1 sản phẩm ra thị trường
- $y \in \mathbb{R}^m$: Số đơn vị nguyên liệu còn lại (hàng tồn)
- $s \in \mathbb{R}^m$: Giá bán nguyên liệu

Giải thích

- $b^T x$: tổng chi phí đặt mua nguyên liệu.
- $l - q$: hệ số chi phí hay số tiền mất khi sản xuất 1 đơn vị sản phẩm, lời khi $(l - q)$ âm.
- $(l - q)^T z$: tổng số tiền mất khi sản xuất z sản phẩm.
- $s^T y$: tổng số tiền lời khi bán lại nguyên liệu thừa.

Theo đề: Sử dụng mô hình 2-SLPWR để giải quyết bài toán với n=8 và m=5 , scenarios S=2 với mật độ ps = 1/2 là số lượng đơn vị nguyên liệu cần đặt hàng trước khi sản xuất. Giả sử rằng nhu cầu của thị trường là $\omega = D = (D1, D2, \ldots, D8)$ trong đó mỗi ω_i với xác suất p_i tuân theo phân phối nhị thức Bin(10,1/2) nên giá trị $\omega_i = range[0, 10]$

Ta chọn:

$$\text{l} = (\text{l1},\text{l2},\ldots,\text{l8}) = (432,\ 458,\ 361,\ 142,\ 233,\ 484,\ 229,\ 199)$$
$$\text{q} = (\text{q1},\text{q2},\ldots \text{q8}) = (4834,\ 8360,\ 5823,\ 2438,\ 6711,\ 4257,\ 8837,\ 8754)$$
$$\Longrightarrow \text{c= l-q= } (\text{c1},\text{c2},\ldots,\text{c8}) = (-4402,\ -7902,\ -5462,\ -2296,\ -6478,\ -3773,\ -8614,\ -8555)$$
$$\text{s} = (\text{s1},\text{s2},\ldots,\text{s5}) = (65,\ 70,\ 17,\ 2,\ 70)$$
$$\text{b} = (\text{b1},\text{b2},\ldots,\text{b5}) = (231,\ 207,\ 302,\ 236,\ 427)$$

$$A_{8\times5} = A_{(8\times5)} = \begin{bmatrix} 2 & 1 & 5 & 8 & 7 \\ 8 & 7 & 8 & 1 & 3 \\ 5 & 1 & 1 & 8 & 4 \\ 2 & 4 & 7 & 9 & 5 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 9 & 1 & 9 & 9 \\ 4 & 9 & 8 & 5 & 4 \\ 3 & 8 & 4 & 9 & 4 \end{bmatrix}$$

Nhu cầu của thị trường ứng với scenarios S = 2 là:

$$\omega_{S1} = D = (D_1, D_2, ..., D_8) = (5, 2, 4, 4, 4, 6, 5, 5)$$
$$\omega_{S2} = D = (D_1, D_2, ..., D_8) = (5, 5, 7, 5, 6, 4, 4, 5)$$

Mô hình của giai đoạn thứ hai được biểu diễn như sau:

$$min_{z,y} Z = \sum_{i=1}^{n}(l_i - q_i)z_i - \sum_{j=1}^{m} s_j y_j$$
$$y_j = x_j - \sum_{i=1}^{n} a_{ij} z_i$$
$$0 \le z_i \le d_i \quad ; y_j > 0$$

Theo yêu cầu bài toán, ta cần tìm x, y, z sao cho $ming(x, y, z)$ được tối ưu hóa:

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

Sau khi chạy code nghiệm tối ưu của bài toán thu được là:

$$ming(x, y, z) = -29503$$

với

$$y_{S1} = (7, 0, 0, 1, 1)$$
$$y_{S2} = (0, 4, 1, 0, 0)$$
$$z_{S1} = (0, 2, 4, 0, 4, 0, 5, 5)$$
$$z_{S2} = (0, 3, 5, 0, 6, 0, 3, 5)$$

2.4 Code: Python + Gamspy

Để giải quyết 1 bài toán LP với lượng biến lớn và khó thì chúng ta có thể sử dụng gói Gamspy trong python kết hợp với numpy. Gamspy cung cấp các công cụ để mô hình hóa và giải quyết các bài toán tối ưu hóa tuyến tính bằng cách sử dụng giao diện lập trình Python thuận tiện.

Trong đó: các giá trị b, l, q, s, A cho sẵn và D ngẫu nhiên.

```
1 import numpy as np
```

```
from gamspy import Container, Set, Parameter, Variable, Equation,
    Model, Sum, Sense

model=Container()
n=8 # i type product
m=5 # j type material
scenario = 2 # 2 scenario

sc=Set(model, name ="sc", description="scenario", records= ["sc"+
    str(k) for k in range (1, scenario + 1)])
i=Set(model, name="i", description="productions", records=["i" + str(x)
    for x in range (1,n+1)])
j=Set(model, name="j", description="materials", records=["j" + str(x)
    for x in range (1,m+1)])

b=Parameter(
    model, name="b", description="pre-order_cost",
    domain=j,
    records= np.random.randint(100, 500, size=(1,m)),
)
s=Parameter(
    model, name="s", description="inventory_cost",
    domain=j,
    records= np.random.randint(1, 99, size=(1,m)),
)
l=Parameter(
    model, name="l", description="production_cost",
    domain=i,
    records= np.random.randint(100, 500, size=(1,n)),
)
q=Parameter(
    model, name="q", description="price product",
    domain=i,
    records= np.random.randint(1000, 10000, size=(1,n)),
)
c=Parameter(
    model, name="c", description="equation c= l-q",
    domain=i,
```

```
)
c[i] = l[i]-q[i]

ps=Parameter(
    model, name="ps", description="probability of scenario",
    domain=sc,
    records=np.array([0.5, 0.5]), # ps[sc]= 0.5
)
d=Parameter(
    model, name="d", description="demand",
    domain=[sc,i],
    records= np.random.binomial(10, 0.5, size=(scenario,n)),
)
a=Parameter(
    container=model, name="a",
    domain=[i,j],
    # matrix A(n x m)
    description="aij",
    records= np.random.randint(1,10, size=(8, 5)),
)

x=Variable(
    model, name="x",
    domain=j,
    type="integer", description="Sum pre-order_material",
)
y=Variable(
    model, name="y",
    domain=[sc,j],
    type="integer", description="inventory",
)
z=Variable(
    model, name="z",
    domain=[sc,i],
    type="integer", description="production",
)


```

```
74
75 supply1 = Equation(model, name="supply1", domain=j,)
76 supply1[j] = x[j] >= 0
77
78 supply2 = Equation(model,  name="supply2", domain=[sc, j],)
79 supply2[sc, j]= y[sc,j] == x[j] - Sum(i, a[i,j]*z[sc, i])
80
81 supply3 = Equation(model, name="supply3", domain=[sc,j],)
82 supply3[sc,j] = y[sc,j] >= 0
83
84 demand1 = Equation(model, name="demand1",  domain=[sc,i],)
85 demand1[sc,i] = z[sc,i] <= d[sc,i]
86
87 demand2 = Equation(model, name="demand2", domain=[sc,i],)
88 demand2[sc,i] = z[sc,i] >= 0
89
90 #! equation 7+8
91 equation78 = Model(
92     model, name="equation78",
93     equations=[supply1, supply2, supply3, demand1, demand2],
94     problem="MIP",
95     sense=Sense.MIN,
96     objective= Sum(j, b[j]*x[j]) + Sum(sc, ps[sc]*(Sum(i, c[i]*z[sc,i])
    - Sum(j, s[j]*y[sc,j]))),
97 )
98
99
100 equation78.solve()
101 print("\nThe solution X of the equation is: ")
102 print(x.records)
103 print("\nThe solution Y of the equation is: ")
104 print(y.records)
105 print("\nThe solution Z of the equation is: ")
106 print(z.records)
107 print(f'Optimal solution found = {equation78.objective_value}')
108 print(f'Solution = {-equation78.objective_value}')
109 print("\n----------------End program----------------")
```

3 Question 2

3.1 Đặt vấn đề

- HIện nay, thiên tai có thể xảy ra bất cứ lúc nào, kể cả động đất, sóng thần, hay bất cứ sự kiện nào cũng có thể gây ra thiệt hại về người và của. Mục tiêu cuối cùng của việc cứu hộ là cung cấp cho người bị ảnh hưởng những sự giúp đỡ cần thiết. Mục tiêu trong bài tập này, là lên kế hoạch cho việc sơ tán người bị ảnh hưởng từ nơi nguy hiểm đến nơi an toàn. Nhưng trên thực tế, có rất nhiều yếu tố tác động đến kế hoạch ban đầu, chẳng hạn như việc cầu đường bị hư hỏng dẫn đến không thể lưu thông, hay việc quá nhiều xe cộ đi cùng trên một cung đường sẽ dẫn đến ùn tắc. Vì thế, chúng ta có thể xem xét bài toán như là một bài toán tối ưu hóa ngẫu nhiên, hay trong nội dung bài tập này sẽ được mô tả ***là lập trình ngẫu nhiên với 2 giai đoạn dựa theo những viễn cảnh***(scenario-based two-stage stochastic programming)

3.2 Chiến thuật

1) Trong bài toán này, chúng ta sẽ xem xét lựa chọn đường đi trước khi thảm họa xảy ra và ngay sau đó, để có thể tạo ra một kế hoạch sơ tán ban đầu, mà ở đó, chúng ta có cân nhắc đến yếu tố ngẫu nhiên theo từng viễn cảnh, bao gồm thời gian di chuyển cũng như sức chứa của từng con đường sẽ có thể thay đổi ở trong giai đoạn 2.

2) Bài toán này đề xuất một mô hình với ***luồng di chuyển có chi phí thấp nhất*** (min – cost flow) dựa trên lập trình ngẫu nhiên với 2 giai đoạn. Thêm vào đó, để có thể dễ dàng giải được bài toán này, ta sẽ chia mô hình này thành 2 bài toán con theo phương pháp xấp xỉ Lagrangian, tương ứng với bài toán luồng di chuyển có chi phí thấp nhất và cũng chính nó trong trường hợp phụ thuộc vào thời gian. Và phương pháp để giải những bài toán này là sử dụng ***Thuật toán 1: Successive Shortest Path*** mà ta sẽ đề cập ở những phần sau.

3.3 Mô tả:

Bảng 1: Chỉ số và tham số sử dụng trong mô hình toán học

Ký hiệu	Định nghĩa
V	Tập hợp của các nút
A	Tập hợp của các liên kết
i , j	Chỉ số của các nút, i , j ∈ V
(i , j)	Chỉ số của các liên kết có hướng, (i , j)∈ A
s	Chỉ số của những viễn cảnh
S	Tổng số lượng những viễn cảnh
v	Giá trị cung cấp của nút ở nguồn
$\widetilde{T}$	Ngưỡng thời gian
T	Thời gian tổng cộng
u_{ij}	Sức chứa của liên kết (i , j)
$u_{ij}^{s}(t)$	Sức chứa của liên kết (i,j) ở viễn cảnh s vào thời điểm t
$c_{ij}^{s}(t)$	Thời gian di chuyển trên liên kết (i,j) ở viễn cảnh s vào thời điểm tĐịnh nghĩa
μ_s	Xác suất của viễn cảnh s

Bảng 2: Biến quyết định sử dụng trong mô hình toán học

Biến quyết định	Định nghĩa
x_{ij}	Luồng di chuyển trên liên kết (i,j)
$y_{ij}^{s}(t)$	Luồng di chuyển trên liên kết (i,j) trong viễn cảnh s tại thời điểm t

3.3.1 Những ràng buộc của bài toán

• **Giai đoạn 1:** Trong giai đoạn 1, một kế hoạch khả thi nên được lập ra để di chuyển từ "super – source" đến "super – sink" (đây là một node giả tập trung tất cả các node bắt đầu và các node kết thúc) . Ta có những ràng buộc sau cho giai đoạn 1:

• Cân bằng luồng (Flow balance):

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

• Đảm bảo về sức chứa:

$$0 \leqslant x_{ij} \leqslant u_{ij}, \forall (i,j) \in A$$

• Để chắc chắn trong lời giải của bài toán, không tồn tại những vòng lặp xảy ra khi bài toán cố gắng đảm bảo về cân bằng luồng, chúng ta tạo nên biến penalty $p_{ij}, (i,j) \in A$, và :

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

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

• **Giai đoạn 2:** Trong giai đoạn này, người sơ tán sẽ nhận được các thông tin về thảm họa dựa trên thời gian thực. Trước một thời điểm mốc (threshold) , người dân sẽ được sơ tán theo một con đường giống

kế hoạch ban đầu (priori path), sau đó, họ sẽ nhận được kế hoạch lúc sau (adaptive path). Theo lập luận trên, ta có ràng buộc giữa kế hoạch ban đầu và lúc sau:

$$\sum_{t\leqslant\widetilde{T}} y_{ij}^{s}(t) = x_{ij}, (i,j) \in A, s = 1,2,...,S(1)$$

Và hàm mục tiêu với những ràng buộc dựa trên từng scenario:

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

s.t.

$$\sum_{(i_t,j_{t'})\in A_s} y_{ij}^{s}(t) - \sum_{(j_{t'},i_t)\in A_s} y_{ji}^{s}(t^{'}) = d_i^{s}(t), \forall i \in V, t \in \{0,1,...,T\}$$

$$0 \leqslant y_{ij}^{s}(t) \leqslant u_{ij}^{s}(t), \forall (i,j) \in A, t \in \{0,1,...,T\}$$

$$\sum_{t\leqslant\widetilde{T}} y_{ij}^{s}(t) = x_{ij}, (i,j) \in A, \text{với } s = 1,2,...,S$$

3.3.2 Mô hình của bài toán

Mô hình này là một mô hình phụ thuộc vào thời gian và là mô hình sơ tán gồm 2 giai đoạn(stochastic two-stage evacuation planning model). Mục tiêu của mô hình này là để phát triển một kế hoạch ban đầu đủ chắc chắn để cung cấp cho người bị ảnh hưởng. Vì ta giả sử rằng có hữu hạn số lượng viễn cảnh có thể xảy ra, ta có một mô hình có duy nhất 1 giai đoạn như sau :

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

3.4 Hướng giải quyết

3.4.1 Giới thiệu phương pháp Lagrangian relaxation:

Theo ta đã thấy, đa số các bài toán thực tế về đồ thị là bài toán NP- complete và do đó rất khó để giải. Và phương pháp Lagrangian Relaxation ra đời như 1 cách để giải quyết, ý tưởng của phương pháp này là sử dụng các cấu trúc ngầm của những vấn đề phức tạp để áp dụng các thuật toán hiệu quả khác, hay nói cách khác, Lagrangian Relaxation là phương thức của sự phân rã (decompositon).

Trong bài toán tìm kiếm đường đi với chi phí thấp nhất mà ta đề cập ở trên, ta có thể thấy ràng buộc (1) là rất khó có thể giải quyết, và do đó ta có thể đưa nó vào hàm mục tiêu cùng với nhân tử Lagrangian :

$$\sum_{s=1}^{S}(\sum_{t\leqslant\widetilde{T}}(\sum_{(i,j)\in A}a_{ij}^{s}(t).(y_{ij}^{s}(t)-x_{ij}))) \quad (2)$$

Để giải quyết bài toán một cách hiệu quả hơn, mô hình gốc có thể được phân tách thành các bài toán nhỏ hơn, mà kết quả tối ưu của chúng sẽ là chặn dưới của kết quả tối ưu của mô hình (*). Theo đó, bài toán gốc sẽ được phân tách thành bài toán min-cost flow tiêu chuẩn và bài toán min-cost flow có dựa vào thời gian, và có thể được giải một cách hiệu quả.

Sau khi relax được ràng buộc khó và đưa chúng vào hàm mục tiêu, ta nhận thấy ta có thể tách 2 biến quyết định, X và Y, ra thành 2 phần khác nhau như sau:

SubProblem 1: Min- cost Flow Problem

$$\begin{cases} minSP1(\alpha)=\sum_{(i,j)\in A}(p_{ij}-\sum_{s=1}^{S}\sum_{t\leqslant\widetilde{T}}a_{ij}^{s}(t))x_{ij} \\ s.t. \\ \sum_{(i,j)\in A}x_{ij}-\sum_{(j,i)\in A}x_{ji}=d_i,\forall i\in V \\ 0\leq x_{ij}\leq u_{ij},\forall(i,j)\in A \end{cases}$$

Ta có thể biểu diễn chi phí tổng quát cho từng link là:

$$g_{ij}:=p_{ij}-\sum_{s=1}^{S}\sum_{t\leqslant\widetilde{T}}a_{ij}^{s}(t)$$

Từ đó, SubProblem 1 có thể được giải bằng thuật toán Successive Shortest Path (được thực hiện bằng Algorithm 1). Ta gọi kết quả tối ưu là $Z_{SP1}^{*}(\alpha)$.

SubProblem 2: Time-Dependent Min-cost Flow Problem·

$$\begin{cases} minSP2(\alpha)=\sum_{s=1}^{S}\sum_{(i,j)\in A}(\sum_{t\in\{0,1,...,T\}}\mu_s.c_{ij}^{s}(t)+\sum_{t\leqslant\widetilde{T}}a_{ij}^{s}(t))y_{ij}^{s}(t) \\ \sum_{(i_t,j_{t'})\in A_s}y_{ij}^{s}(t)-\sum_{(j_{t'},i_t)\in A_s}y_{ji}^{s}(t')=d_i^{s}(t),\forall i\in V, \\ t\in\{0,1,...,T\},s=1,2,...,S \\ 0\leqslant y_{ij}^{s}(t)\leqslant u_{ij}^{s}(t),\forall(i,j)\in A,t\in\{0,1,...,T\},s=1,2,...,S \end{cases}$$

Ta gọi kết quả tối ưu của SubProblem 2 là$Z_{SP1}^{*}(\alpha)$.

Và với 2 giai đoạn, cho từng viễn cảnh $s\in 1,2,\ldots,S$, chi phí tổng quát có thể được mô tả như sau:

$$g_{ij}^{s}=\mu_s.c_{ij}^{s}(t)+a_{ij}^{s}(t),t\leqslant\widetilde{T}$$
$$g_{ij}^{s}=\mu_s.c_{ij}^{s}(t),t\leqslant\widetilde{T}$$

Khi hoàn tất, ta có thể biểu diễn kết quả tối ưu với một bộ của vecto nhân tử Lagrangian α như sau:

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

Nhưng, khi ta muốn làm cho nghiệm bài toán rõ ràng hơn, vì kết quả tối ưu của mô hình đã được relax là chặn dưới của kết quả tối ưu cuối cùng, nên ta có thể tìm kết quả nào gần với kết quả tối ưu cuối cùng hơn bằng cách:

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

3.4.2 Phân tích Algorithm 1:

Algorithm 1 sử dụng giải thuật successive shortest path for min – cost flow problem được mô tả trong bài báo như sau:

Step 1: Take variable x as a feasible flow between any OD and it has the minimum delivery cost in the feasible flows with the same flow value.
Step 2: The algorithm will terminate if the flow value of x reaches v or there is no minimum cost path in the residual network (V, A(x), C(x), U(x), D); otherwise, the shortest path with the maximum flow is calculated by label-correcting algorithm, and then go to Step 3. The functions A(x), C(x), U(x) in the residual network can be defined as:

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

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

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

Step 3: Increase the flow along the minimum cost path. If the increased flow value does not exceed v, go to Step 2.

Tổng quan, chúng ta tìm hiểu về các tính chất của một bài toán minimum – cost – flows. Xem xét một đồ thị G = (V, E) với sức chứa, chặn dưới, chi phí và sự cân bằng. Một luồng di chuyển "giả" $\psi : E \to R$ là một luồng di chuyển có chi phí thấp nhất nếu nó thỏa mãn 3 điều kiện:
• **Khả thi (Feasible):** $l(e) \leqslant \psi(e) \leqslant u(e)$ với mọi cạnh e
• **Cân bằng (Balanced):** $\sum_u (\psi(u \to v) - \psi(v \to w)) = b(v))$ với mọi đỉnh v
• **Tính đúng đắn cục bộ (Locally Optimal):** residual graph G_ψ không chứa những vòng lặp có chi phí là âm

Một chiến thuật bổ sung được gọi là successive shortest path sẽ dần dần cải thiện sự khả thi và tính đúng đắn cục bộ của pseudo flow cho đến khi nó thực sự cân bằng.
Giải thuật bắt đầu bằng việc khởi tạo pseudo flow ψ , và cứ thế đẩy thêm "flow" qua một con đường ngắn nhất trong residual graph G_ψ. Và câu hỏi đặt ra là liệu giải thuật có thể trả về một luồng di chuyển khả thi, hay có thể biết được không tồn tại một luồng di chuyển nào như thế. Chúng ta cần xem xét một nhận định quan trọng: Con đường ngắn nhất (Shortest paths) chỉ được định nghĩa đầy đủ ở trong một đồ thị mà không có vòng lặp âm. Để có thể xem xét giải thuật Successive Shortest Path, chúng ta cần có bổ đề sau:
Sau mỗi vòng lặp của giải thuật Successive Shortest Path, một pseudo flow ψ khả thi thì cũng có tính đúng đắn cục bộ, và đó, residual graph G_ψ không chứa vòng lặp âm.

Trong mỗi vòng lặp của giải thuật Successive Shortest Path, chúng ta có thể tìm đường đi ngắn nhất bằng giải thuật label- correcting (chúng ta có thể dùng giải thuật Dijkstra để tìm ra đường đi ngắn nhất đầu tiên, nhưng khi chúng ta thực hiện việc tăng luồng qua một cạnh có chi phí là dương, thì sẽ sinh ra một "residual edge" với chi phí âm, nên chúng ta không thể dung giải thuật Dijkstra trong những vòng lặp sau).

• Tìm hiểu về phương pháp Label Correcting: Một phương pháp đơn giản nhất để hiện thực Label Correcting là sử dụng quy tắc FIFO để cập nhật hàng đợi lưu các đỉnh có khả năng xét tới. Hay chúng ta nói đây là phương pháp Bellman – Ford, độ phức tạp của thuật toán có thể dựa vào tính chất sau đây:

Với từng node i và số nguyên k $\geqslant$ 1,cho:

d_i^k=Khoảng cách ngắn nhất từ 1 đến i sử dụng con đường có k cạnh hay ít hơn

Nơi mà $d_i^k = \infty$ nếu như không tồn tại một con đường từ 1 đến i với k cạnh hay ít hơn. Khi đó nhãn d_i tại cuối chu kỳ thứ k của của mỗi vòng lặp bé hơn hoặc bằng d_i^k

Tính chất trên ngụ ý rằng phương pháp Label Correcting, sẽ tìm được tất cả khoảng cách ngắn nhất sau nhiều nhất V − 1 chu kỳ. Bởi vì mỗi chu kỳ trong vòng lặp cần tổng cộng O(E) phép tính (mỗi cạnh được xét duy nhất một lần trong mỗi chu kỳ), thời gian chạy của phương pháp này là O(VE).

Tổng quát thuật toán Successive Shortest Path, số lượng vòng lặp là O(B), độ phức tạp của thuật toán phụ thuộc vào việc tính toán đường đi ngắn nhất, độ phức tạp tương đối sẽ là O(VEB) , với B được hiểu là tổng lượng flow cần thiết được gửi hay được nhận xuyên suốt đồ thị.
Với bài toán thực tế được trình bày trong paper tham khảo, việc tìm ra được "residual graph" cũng phức tạp hơn một chút so với một bài toán bình thường, với các hàm tính "residual graph" của SubProblem 1 đã được trình bày trong Algorithm 1, các hàm tương tự của SubProblem 2 là:

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

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

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

Với phương pháp Label ˘ Correcting trong thực tế thì để áp dụng được cho các hệ thống thực hay có quy mô lớn, chúng ta phải điều chỉnh để cho phù hợp với bài toán yêu cầu hơn. Theo em tìm hiểu, với phương pháp Label – Correcting đã được điều chỉnh, độ hiệu quả có thể phụ thuộc vào nhiều yếu tố khác nhau trong việc chọn trạng thái ban đầu hay cách mà đồ thị sẽ thay đổi theo thời gian. Và trong những trường hợp tổng quát, phương pháp Label ˘ Correcting khi xem xét đến thời gian sẽ có độ phức tạp là O(V^3M^2), với M là một số nguyên lớn sao cho khoảng thời gian từ t_0 đến $t_0 + M\delta$ là giai đoạn bài toán được xét, và là khoảng thời gian nhỏ mà có sự thay đổi trong tình trạng đồ thị.

4 Tài liệu tham khảo

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

2. Dimitri P.Bertsekas, (2005). Dynamic Programming and Optimal Control, Volume I, Third Edition.

3. Dimitri P. Bertsekas,(1998). Network Optimization: Continuous and Discrete Models.

4. Jeff Erickson, (2019). Algorithms, First Edition Ziliaskopoulos, Mahmassani. Time-Dependent, Shortest-Path Algorithm for Real-Time Intelligent Vehicle Highway System Applications.

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

6. S. W. Wallace and W. T. Ziemba, Applications of stochastic programming. SIAM, 2005.