

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



MÔ HÌNH HOÁ TOÁN HỌC - CO2011

---

STOCHASTIC PROGRAMMING AND APPLICATIONS

---

LỚP L02 - HỌC KỲ 231

GVHD: TS. Nguyễn Tiến Thịnh  
SV: Nguyễn Hữu Chiến - 2110855 (Nhóm trưởng)  
Lê Hoàng Ngọc Hân - 2210935  
Nguyễn Thị Hiền Hạnh - 2210920  
Nguyễn Mạnh Tuấn - 2213786  
Đào Hồ Hương Vi - 2112647

TP. HỒ CHÍ MINH, THÁNG 12/2023

# Mục lục

<b>1</b>	<b>Vấn đề 1</b>	<b>2</b>
1.1	Tổng quan kiến thức . . . . .	2
1.2	Giải quyết vấn đề . . . . .	3
1.3	Hiện thực . . . . .	4
1.4	Nhận xét . . . . .	8
<b>2</b>	<b>Vấn đề 2</b>	<b>9</b>
2.1	Bài toán sơ tán hai giai đoạn của Li Wang (2020) . . . . .	9
2.1.1	Giới thiệu . . . . .	9
2.1.2	Giải quyết bài toán . . . . .	10
2.1.3	Xây dựng mô hình . . . . .	13
2.1.4	Thuật toán . . . . .	16
2.2	Thuật toán Successive Shortest Path . . . . .	19
2.3	Demo thuật toán . . . . .	21
2.3.1	Cách thức demo . . . . .	21
2.3.2	Kết quả demo . . . . .	25
2.3.3	Độ hiệu quả của giải thuật . . . . .	26
2.4	Kết luận . . . . .	29
<b>3</b>	<b>Tài liệu tham khảo</b>	<b>30</b>

# 1 Vấn đề 1

## 1.1 Tổng quan kiến thức

Mô hình truy đòi SLP 2 giai đoạn – dạng chính tắc.

Mô hình Stochastic Linear Program (SLP) là một dạng của lập kế hoạch tuyến tính mà có sự không chắc chắn (stochasticity) trong dữ liệu đầu vào.

Recourse action có thể là các hành động, chiến lược được thực hiện sau khi sự không chắc chắn đã được tiết lộ.

Mô hình Stochastic Linear Program With Recourse action (SLPWR) được thiết kế để xử lý không chắc chắn ở hai cấp độ: một là trong các tham số và ràng buộc của mô hình tuyến tính và hai là sau khi nhận thông tin mới (recourse action).

Chương trình tuyến tính ngẫu nhiên có truy đòi (2-SLPWR) được xây dựng dưới dạng:

$$2 - SLP : \min_x g(x) := c^T \cdot x + v(y)$$

mà  $v(y) := \min_{y \in R_+^p} q \cdot y$

Mô hình tuân theo quy tắc sau:

+ Những hạn chế ở giai đoạn đầu:  $g(x)$  tuân theo  $Ax = b$  với  $A$  là ma trận,  $x$  và  $b$  là vector.

mà  $x \in X \subset R^n, x \geq 0$

+ Ràng buộc ở giai đoạn 2:  $v(y)$  tuân theo  $W \cdot y = h(\omega) - T(\omega) \cdot x =: z$

Trong đó  $v(y) := v(x, \omega)$  là hàm giá trị giai đoạn 2 và  $y = y(x, \omega) \in R_+^p$  là thao tác truy đòi với quyết định  $x$  và hiện thực hóa  $\omega$ .

Chi phí truy đòi của quyết định  $x$  là  $Q(x) := E_\omega[v(x, \omega)]$  (chi phí dự kiến trước việc truy đòi  $y(\alpha)$  cho bất kỳ chính sách  $x \in R^n$ ). Do đó, tổng chi phí bé nhất dự kiến là

$$\min_{x \in R^n, y \in R_+^p} c^T \cdot x + Q(x).$$

Thiết kế các biến quyết định thứ 2  $y(\omega)$  để có thể sửa đổi, điều chỉnh đối với các ràng buộc ban đầu theo cách thông minh, đó là hành động truy đòi.

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

Giá trị tối ưu của LP giai đoạn 2 là  $v_* = v(y^*)$  với  $y^* = y^*(x, \omega)$  là giải pháp tối ưu tại  $y^* \in R_+^p$ . Tổng giá trị tối ưu là  $c^T \cdot x^* + v(y^*)$ .

Mục tiêu của mô hình SLPWR thường là tối ưu hóa giá trị kỳ vọng của một hàm mục tiêu trong bối cảnh của sự không chắc chắn và khả năng điều chỉnh sau khi có thêm thông tin.

## 1.2 Giải quyết vấn đề

Vấn đề cần giải quyết là tìm phương án sản xuất tốt nhất sản phẩm trong sản xuất công nghiệp với ác biến, vector, ma trận được ký hiệu như sau:

Ký hiệu	Định nghĩa
$n$	số sản phẩm
$m$	số linh kiện lắp ráp
$A$	ma trận kích thước $m \times n$ biểu thị một sản phẩm $i$ cần $j$ linh kiện, với $a_{ij}$ là số đơn vị linh kiện cụ thể ( $a_{ij} \geq 0$ ), $i = 1, \dots, n$ và $j = 1, \dots, m$
$D$	vector ngẫu nhiên có chiều dài $n$ biểu thị số sản phẩm mong muốn sản xuất
$s$	vector kích thước $m$ biểu thị giá sản phẩm tồn kho
$x$	vector kích thước $m$ là số linh kiện cần đặt hàng
$y$	vector có chiều dài $m$ biểu thị số linh kiện còn lại
$z$	vector kích thước $n$ biểu thị số lượng đơn vị được sản xuất
$l$	vector kích thước $n$ biểu thị chi phí bổ sung
$q$	vector kích thước $n$ biểu thị đơn giá sản phẩm
$c$	vector kích thước $n$ biểu thị hệ số chi phí với $c_i = l_i - q_i$
$b$	vector kích thước $n$ biểu thị chi phí đặt hàng trước
$p$	mật độ xác suất
$Q(x)$	hàm biểu thị giá trị tối ưu

Với 2 phương trình cần giải quyết:

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

Quan sát 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 quyết định ở giai đoạn 1  $x = (x_1, x_2, \dots, x_n)$ .

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

trong đó  $Q(x) = E_\omega[Z] = \sum_{i=1}^n p_i c_i z_i$  được lấy bằng phân bố xác suất của  $\omega = D$ .

Sự phân bố của biến ngẫu nhiên này phụ thuộc vào các quyết định  $x$  ở giai đoạn đầu, và do đó, bài toán ở giai đoạn đầu không thể giải được nếu không hiểu bài toán ở giai đoạn thứ hai.

Trong trường hợp đặc biệt có hữu hạn nhiều kích bản nhu cầu  $d^1, \dots, d^S$  xảy ra với xác suất dương  $p^1, \dots, p^S$ , với  $\sum_{k=1}^S p_k = 1$ , bài toán hai giai đoạn  $(*)-(**)$  có thể được viết dưới dạng một bài toán quy hoạch tuyến tính quy mô lớn:

$$\text{Mô hình} = \begin{cases} \min c^T x + \sum_{k=1}^S p_k [(l - q)^T z^k - s^T y^k] \\ \text{với } y^k = x - A^T z^k, k = 1, \dots, S. \\ 0 \leq z^k \leq d^k, y^k \geq 0, k = 1, \dots, S \\ x \geq 0 \end{cases} \quad (***)$$

Trong đó việc giảm thiểu được thực hiện trên các biến vector  $x$  và  $z^k, y^k$ ,  $k = 1, \dots, S$ . chúng tôi đã tích hợp bài toán (\*) ở giai đoạn thứ hai vào công thức này, nhưng chúng tôi phải cho phép nghiệm  $(z^k, y^k)$  của nó phụ thuộc vào kịch bản  $k$ , bởi vì việc hiện thực hóa nhu cầu  $d$  là khác nhau trong mỗi kịch bản. Vì vậy, bài toán (\*\*\*) có số lượng biến và số ràng buộc tỉ lệ thuận với số lượng kịch bản  $S$ .

Điều đáng chú ý sau đây. Ở đây có 3 loại biến quyết định: số lượng bộ phận được đặt hàng (vector  $x$ ), số lượng đơn vị được sản xuất (vector  $z$ ) và số lượng bộ phận còn lại trong kho (vector  $y$ ). Các biến quyết định này được phân loại một cách tự nhiên thành các biến quyết định giai đoạn đầu và giai đoạn hai. Nghĩa là, các quyết định  $x$  ở giai đoạn đầu phải được đưa ra trước khi hiện thực hóa dữ liệu ngẫu nhiên và do đó phải độc lập với dữ liệu ngẫu nhiên, trong khi các biến quyết định ở giai đoạn thứ hai  $z$  và  $y$  được đưa ra sau khi quan sát dữ liệu ngẫu nhiên và được các chức năng của dữ liệu các biến quyết định ở giai đoạn đầu thường được gọi là quyết định hiện tại (giải pháp) và các quyết định ở giai đoạn thứ hai được gọi là quyết định chờ xem (giải pháp). Cũng có thể nhận thấy rằng bài toán (\*) ở giai đoạn thứ hai là khả thi cho mọi khả năng thực hiện dữ liệu ngẫu nhiên; ví dụ: lấy  $z = 0$  và  $y = x$ . trong tình huống như vậy chúng ta nói rằng bài toán có cách truy đòi tương đối đầy đủ.

### 1.3 Hiện thực

- Tải thư viện

```
1 pip install pulp
2 pip install gamspy
3 pip install numpy
```

Program 1: Tải thư viện

- Khởi tạo biến và ma trận

```
1 # Tao data
2
3 import numpy as np
4
5 # Bien yeu cau
6 n = 8 # So san pham
7 m = 5 # So linh kien
8 S = 2 # So scenarios
9
10 # Set seed de kiem soat bo data tao ra
11 np.random.seed(0)
12
13 # Ma tran A
14 A = np.random.randint(1, 10, size=(n, m))
15
16 # b, l, q, s
17 b = np.random.uniform(1, 5, size=m) # gia dat hang truoc/unit linh kien
18 l = np.random.uniform(5, 10, size=n) # chi phi thoa man 1 unit demand cho moi san pham
19 q = np.random.uniform(1, 5, size=n) # gia ban/1 unit
20 s = np.random.uniform(0.5, 3, size=m) # gia tri thu hoi/unit linh kien
21
22 #in data
23 A, b, l, q, s
```

---

Program 2: Khởi tạo biến và ma trận

```
(array([[6, 1, 4, 4, 8],  
       [4, 6, 3, 5, 8],  
       [7, 9, 9, 2, 7],  
       [8, 8, 9, 2, 6],  
       [9, 5, 4, 1, 4],  
       [6, 1, 3, 4, 9],  
       [2, 4, 4, 4, 8],  
       [1, 2, 1, 5, 8]]),  
array([2.05822245, 4.09693476, 2.82460133, 3.2737358 , 1.0751592 ]),  
array([8.08817749, 8.06047861, 8.08466998, 9.71874039, 8.4091015 ,  
       6.7975395 , 7.18515977, 8.48815598]),  
array([1.24090189, 3.66706686, 3.68255148, 1.84153024, 1.51570519,  
       2.2617134 , 2.45484308, 3.28078708]),  
array([1.59650378, 2.9709346 , 0.75511203, 1.02219189, 0.90327379]))
```

Hình 1: Thiết lập biến và ma trận A

```
1 # tao data demand D cho tung san pham, theo Bin(10, 0,5)
2 D = [np.random.binomial(10, 0.5, S) for _ in range(n)]
3
4 # in data
5 D
```

Program 3: Tạo data cho demand

```
[array([6, 4]),
 array([5, 4]),
 array([3, 3]),
 array([6, 3]),
 array([4, 4]),
 array([6, 3]),
 array([7, 3]),
 array([8, 5])]
```

Hình 2: Thiết lập demain D

```

1 import pulp as lp
2
3 # tao model, Lp.LpProblem = linear programming problem, import pulp de giai, model nay xac
  dinh muc tieu la minimize
4 model = lp.LpProblem("2-SLPWR", lp.LpMinimize)
5
6 # decision variables
7 # xj: tao xj voi gia tri tu 0 -> m
8 x = lp.LpVariable.dicts("x", range(m), lowBound=0)
9
10 # y^kj: tuong tu
11 y = [[lp.LpVariable(f"y_{k}_{j}", lowBound=0) for j in range(m)] for k in range(S)]
12
13 # z^ki: tuong tu
14 z = [[lp.LpVariable(f"z_{k}_{i}", lowBound=0, upBound=10) for i in range(n)] for k in range(
  S)]
15
16 # ham muc tieu
17 # Giam thieu tong chi phi bao gom preorder cost va expected cost voi cac scenarios khac nhau
18 model += lp.lpSum(b[j] * x[j] for j in range(m)) + \
19         lp.lpSum(0.5 * ((lp.lpSum((l[i] - q[i]) * z[k][i] for i in range(n))) - \
20                        lp.lpSum(s[j] * y[k][j] for j in range(m))) for k in
21                        range(S))
22
23 # gioi han
24 for k in range(S):
25     for j in range(m):
26         model += y[k][j] == x[j] - lp.lpSum(A[i][j] * z[k][i] for i in range(n))
27
28     for i in range(n):
29         model += z[k][i] <= D[i][k]
30
31 #in model
32 model

```

Program 4: Set model

```

1 # Giai model
2 model.solve()
3
4 # tim ra gia tri x, y, z toi uu
5 x_solution = {j: x[j].varValue for j in range(m)}
6 y_solution = {(k, j): y[k][j].varValue for k in range(S) for j in range(m)}
7 z_solution = {(k, i): z[k][i].varValue for k in range(S) for i in range(n)}
8
9 # In ket qua
10 print("Optimal values x", x_solution)
11 print("Optimal values y", y_solution)
12 print("Optimal values z", z_solution)

```

Program 5: Giải model và tìm ra giá trị tối ưu

```

Optimal values x {0: -0.0, 1: -0.0, 2: -0.0, 3: -0.0, 4: -0.0}
Optimal values y {(0, 0): 0.0, (0, 1): 0.0, (0, 2): 0.0, (0, 3): 0.0, (0, 4): 0.0, (1, 0): 0.0, (1, 1): 0.0, (1, 2): 0.0, (1, 3): 0.0, (1, 4): 0.0}
Optimal values z {(0, 0): 0.0, (0, 1): 0.0, (0, 2): 0.0, (0, 3): 0.0, (0, 4): 0.0, (0, 5): 0.0, (0, 6): 0.0, (0, 7): 0.0, (1, 0): 0.0, (1, 1): 0.0,
(1, 2): 0.0, (1, 3): 0.0, (1, 4): 0.0, (1, 5): 0.0, (1, 6): 0.0, (1, 7): 0.0}

```

Hình 3: Kết quả cuối cùng



## 1.4 Nhận xét

- Mô hình 2-SLPWR dự đoán khả năng sản xuất sản phẩm cần thiết mà đáp ứng được các yêu cầu liên quan không chắc chắn tương đối tốt.
- Giải quyết được vấn đề trong sản xuất công nghiệp của các nhà máy.
- Ngoài ra mô hình này còn được ứng dụng ở nhiều lĩnh vực khác như Logistics, Tài chính, Quản lý dự án,...
- Tuy nhiên, hạn chế trong mô hình này đem lại là sự phức tạp về tính toán, khi nguồn dữ liệu lớn và các ràng buộc quá nhiều.
- Mô hình này còn đòi hỏi về phân phối xác suất để đưa ra giả định, từ đó giải quyết vấn đề. Nếu giả định đó không chính xác thì kết quả mô hình này đưa ra cũng không chính xác, vì vậy hạn chế của mô hình này cũng là phụ thuộc vào giả định phân phối xác suất.
- Đặc biệt, mô hình 2-SLPWR yêu cầu khả năng giải quyết tư duy, giải thuật phức tạp khi sử dụng mô hình này trong các dự án, kinh doanh, sản xuất.

## 2 Vấn đề 2

### 2.1 Bài toán sơ tán hai giai đoạn của Li Wang (2020)

#### 2.1.1 Giới thiệu

Thiên tai cực đoan (động đất, bão, hỏa hoạn, cuồng phong, v.v.) có thể tấn công một cộng đồng mà không hề có cảnh báo trước và để lại nhiều thiệt hại về tài sản và tính mạng. Mục tiêu chính của ứng phó khẩn cấp là cung cấp chỗ trú ẩn và hỗ trợ những người bị ảnh hưởng càng sớm càng tốt. Để đạt được mục tiêu này, một số nghiên cứu về ứng phó khẩn cấp tập trung vào việc tìm cách cung cấp nhu yếu phẩm cho những người ở khu vực an toàn và/hoặc xác định vị trí của các trung tâm cung cấp nhu yếu phẩm. Tuy nhiên, một số nhà nghiên cứu khác quan tâm đến việc lên kế hoạch sơ tán rõ ràng cho những người bị ảnh hưởng từ khu vực nguy hiểm đến khu vực an toàn. Nghiên cứu này nhằm mục đích giải quyết vấn đề lên kế hoạch sơ tán cho những người bị ảnh hưởng bằng cách xây dựng một lộ trình chung khi xảy ra thảm họa. Như mọi người đã biết, rất khó để ước tính tác động và thiệt hại trên đường đi do thảm họa gây ra (chẳng hạn như động đất, bão, v.v.) vì chúng ta gần như không biết trước đường cường độ của thiên tai. Vì vậy, vấn đề sơ tán này thường được coi như là một bài toán quy hoạch động ngẫu nhiên, trong đó tính ngẫu nhiên không chỉ phát sinh từ thời gian di chuyển mà còn phát sinh từ sức chứa của đường đi. Nói cách khác, thiệt hại xảy ra trên một đường đi nhất định dẫn đến sự ngẫu nhiên về thời gian di chuyển và sức chứa có thể ngăn người dân thoát khỏi vùng thiên tai.

Với sự phát triển của internet và công nghệ thông tin hiện đại, mọi người có thể nắm bắt được thông tin về mạng lưới đường bộ theo thời gian thực và về những sự kiện bất ngờ thông qua nhiều kênh truyền thông khác nhau. Vì vậy, làm thế nào để tìm con đường sơ tán tối ưu cho người dân bị ảnh hưởng trong điều kiện đã có sẵn thông tin theo thời gian thực là vấn đề được quan tâm. Để đạt được mục đích này, chúng ta sẽ tiếp tục xem xét tính sẵn có thông tin theo thời gian thực và do đó quá trình sơ tán có thể được chia thành hai giai đoạn. Trong giai đoạn đầu tiên, người ta giả định rằng những người bị ảnh hưởng không thể có được thông tin về mức độ thiên tai và mức độ thiệt hại của đường đi; và sau một khoảng thời gian nhất định, họ có thể nhận được thông tin mạng lưới đường bộ chính xác thông qua một số thiết bị giám sát theo thời gian thực. Theo chúng tôi được biết, quy hoạch ngẫu nhiên với truy đòi được sử dụng để tìm ra những quyết định không lường trước được phải được đưa ra tiên nghiệm để biết cách thực hiện của một số biến ngẫu nhiên sao cho tổng chi phí dự kiến của các hành động truy đòi có thể được giảm thiểu. Dựa trên phân tích trên, bài toán sơ tán được xây dựng dưới dạng mô hình quy hoạch ngẫu nhiên hai giai đoạn-dựa-trên-tình-huống để biểu diễn tính ngẫu nhiên phát sinh từ cường độ trận động đất và ảnh hưởng của nó. Mô hình này gồm hai giai đoạn. Các quyết định ở giai đoạn đầu là kế hoạch sơ tán đáng tin cậy cho tất cả các cấp độ thảm họa. Các quyết định ở giai đoạn thứ hai liên quan đến kế hoạch sơ tán cho những người bị ảnh hưởng để ứng phó với các điều kiện đường đi dựa trên tình huống cụ thể.

Những ý chính của bài báo này có thể được tóm tắt như sau:

(1) Bài báo này trước tiên đề xuất mô hình quy hoạch ngẫu nhiên hai giai đoạn xem xét cả đường đi tiên nghiệm (trước thảm họa) và đường đi thích ứng (sau thảm họa) để cung cấp kế hoạch sơ tán tiên nghiệm cho những người bị ảnh hưởng khỏi khu vực nguy hiểm tới khu vực an toàn. Không giống như các mô hình tất định (deterministic models), mô hình này liên quan đến sự sẵn có của thời gian và sức chứa ngẫu nhiên dựa trên tình huống. Vì vậy, bài viết này đề cập đến việc lập được kế hoạch sơ tán tiên nghiệm cho những người bị ảnh hưởng có tính đến tính không chắc chắn của thời gian di chuyển và sức chứa của đường đi trong giai đoạn thứ hai.

(2) Để xây dựng lộ trình di chuyển rõ ràng cho người dân bị ảnh hưởng khi thảm họa xảy ra, bài báo này đề xuất mô hình luồng chi phí tối thiểu (min-cost flow) dựa trên quy hoạch ngẫu nhiên hai giai đoạn. Hơn thế nữa, mô hình này được phân tách thành hai bài toán con bằng phương pháp Lagrangian relaxation. Hai bài toán con tương ứng là bài toán luồng chi phí tối thiểu (min-cost flow) và trường hợp phụ thuộc thời gian của nó. Do đó, mô hình quy hoạch ngẫu nhiên hai giai đoạn được phân thành hai bài toán con có thể giải quyết được.

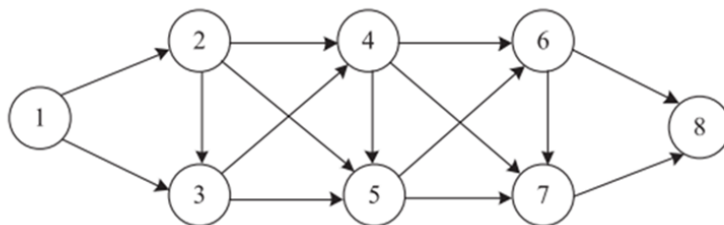
(3) Để thu được lời giải tối ưu gần chính xác, tối ưu hóa subgradient được áp dụng để giảm dần khoảng cách tương đối của giới hạn trên và giới hạn dưới, tức là các giá trị mục tiêu của mô hình ban đầu và mô hình phân rã; và hai bài toán con được giải quyết bằng thuật toán đường đi ngắn nhất liên tiếp (successive shortest path). Các kết quả tính toán cho biết rằng mô hình được đề xuất trong bài viết này vượt trội hơn cả mô hình tất định và mô hình wait-and-see, minh chứng rằng mô hình sơ tán ngẫu nhiên hai giai đoạn cung cấp công cụ đưa ra quyết định thực tế cho việc sơ tán người dân bị ảnh hưởng trong khu vực thảm họa.

### 2.1.2 Giải quyết bài toán

#### \* Biểu diễn bài toán sơ tán như một bài toán quy hoạch ngẫu nhiên hai giai đoạn:

Lấy động đất làm ví dụ để mô tả quá trình sơ tán người dân. Trong quá trình sơ tán, giả định rằng những người bị ảnh hưởng có thể nhận được thông tin cảnh báo sớm và họ thoát khỏi khu vực nguy hiểm bằng chính ô tô của mình. Trong giai đoạn sau-sự-kiện, sau khi nhận được cảnh báo nguy hiểm, thông tin chính xác về trận động đất sẽ được cung cấp bằng các công cụ liên lạc tiên tiến. Tuy nhiên, việc ứng phó khẩn cấp và sơ tán ban đầu được thực hiện mà không biết chính xác phạm vi và mức độ ảnh hưởng của trận động đất. Trong trường hợp này, phản ứng ban đầu sẽ phụ thuộc vào một số tình huống thiệt hại. Do đó, giai đoạn sơ tán nên được chia thành hai giai đoạn theo thời gian thu thập thông tin chính xác. Trong giai đoạn đầu tiên không lường trước được, quyết định sơ tán phải được đưa ra trước khi xảy ra những tình huống không chắc chắn trong tương lai và kế hoạch sơ tán ở giai đoạn thứ hai được xác định sau khi thông tin không chắc chắn được xác thực. Dựa trên phân tích trên, mục tiêu là lập kế hoạch sơ tán tối ưu trong giai đoạn đầu tiên trong điều kiện không chắc chắn sẽ phải đối mặt trong giai đoạn thứ hai.

Sau đây, khái niệm bài toán lập kế hoạch sơ tán ngẫu nhiên hai giai đoạn được minh họa cụ thể bằng một network đơn giản với 8 nút và 15 liên kết (xem [Hình 4](#)).



Hình 4: Minh họa network đường sơ tán.

Ở đây, nút 1 và 8 được giả định lần lượt là khu vực nguy hiểm và khu vực an toàn, và có bốn ô tô là **a**, **b**, **c**, và **d** để sơ tán đến khu vực an toàn. Trong giai đoạn trước và đầu sự kiện, bốn

ô tô này được giả định là được sơ tán theo kế hoạch tiên nghiệm, tức là ô tô **a** và **b** đi dọc theo các đường  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 8$ , **c** và **d** đi dọc theo các đường  $1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$ . Trong giai đoạn sau-sự-kiện mà thông tin đường chính xác được nhận, tức là sau ngưỡng thời gian  $\tilde{T}$ , phần sau của kế hoạch có thể được thay đổi cho thích ứng. Để đơn giản, hai tình huống ngẫu nhiên được tính đến để minh họa cho việc lập kế hoạch. Trong khi đó, Hình 5 cung cấp mạng không gian-thời gian của Hình 4 bằng cách rời rạc hóa khoảng thời gian thành nhiều đơn vị thời gian. Trong mạng lưới (network) này, trục hoành biểu thị khoảng thời gian và trục tung biểu thị các nút trong network. Rõ ràng trục hoành được chia thành hai giai đoạn theo ngưỡng thời gian  $\tilde{T}$ . Kế hoạch sơ tán được trình bày ở bên trái Hình 5 và kế hoạch thích ứng trong hai tình huống được biểu diễn ở bên phải Hình 5. Kế hoạch sơ tán cho bốn ô tô này được đánh dấu bằng các vạch màu trong các tình huống khác nhau. Cụ thể, phương án sơ tán trong tình huống 1 là ô tô **a** và **b** đi dọc theo đường  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 8$ , ô tô **c** và **d** đi dọc theo đường  $1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 8$ ; và trong tình huống 2, ô tô **a** và **b** đi dọc theo đường  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 8$ , **c** đi dọc theo đường  $1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 8$  và **d** đi dọc theo đường  $1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$ .

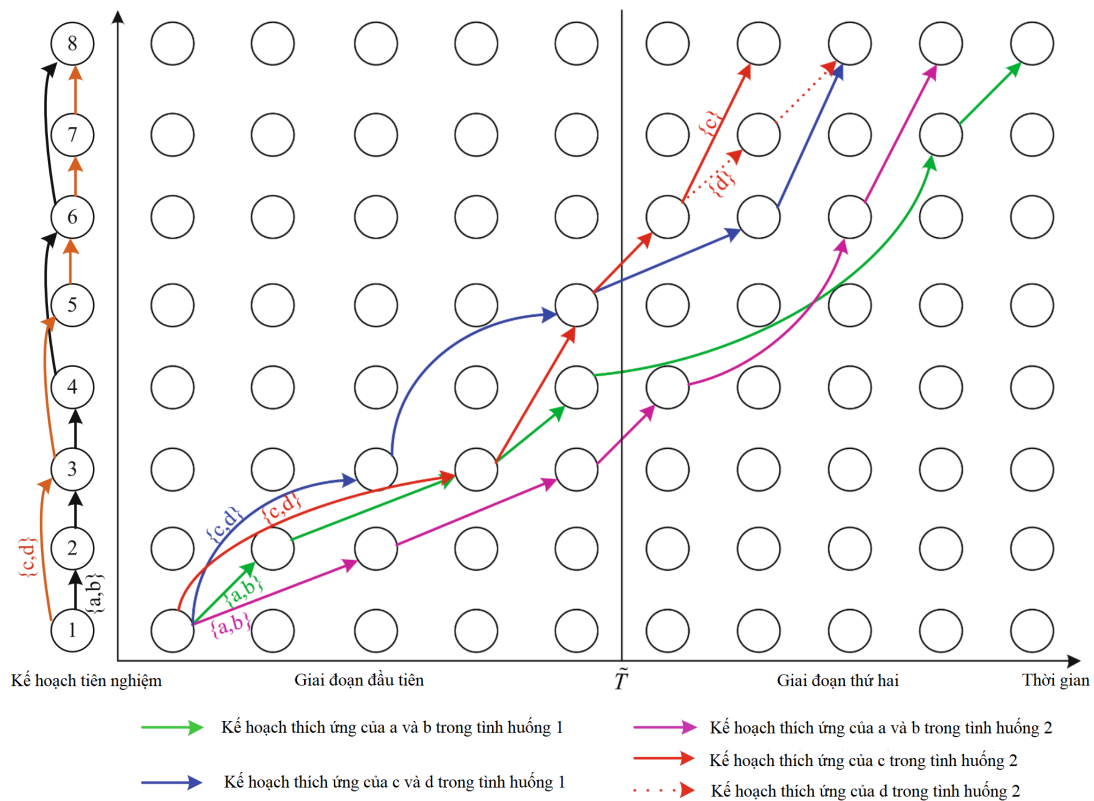
Từ Hình 5, có thể thấy rằng, trước ngưỡng thời gian  $\tilde{T}$ , hành trình con trong kế hoạch sơ tán thích ứng theo tình huống 1 và 2 giống như trong kế hoạch tiên nghiệm, cụ thể là ô tô a và b sơ tán dọc theo đường phụ  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ , c và d đi dọc theo đường dẫn phụ  $1 \rightarrow 3 \rightarrow 5$ . Khi có thông tin chính xác, tức là sau ngưỡng thời gian  $\tilde{T}$ , kế hoạch sơ tán thích ứng trong hai tình huống sẽ khác với kế hoạch sơ tán tiên nghiệm. Ví dụ: kế hoạch sơ tán tiên nghiệm cho a và b là đường phụ  $6 \rightarrow 8$ , trong khi kế hoạch thích ứng cho a và b lần lượt là đường phụ  $7 \rightarrow 8$  và  $6 \rightarrow 8$  trong hai tình huống.

Một ví dụ đơn giản được đưa ra để phân tích bài toán sơ tán ngẫu nhiên hai giai đoạn dựa trên tình huống. Kế hoạch sơ tán tiên nghiệm trong giai đoạn đầu tiên phải được thực hiện trước khi nắm được các thông số ngẫu nhiên. Sau đó, các quyết định ở giai đoạn thứ hai liên quan đến các kế hoạch sơ tán đối với các tình huống cụ thể. Bài báo này tập trung vào việc lập ra kế hoạch sơ tán tiên nghiệm bằng cách xem xét khả năng truy đòi trong giai đoạn thứ hai.

#### \* Mô tả bài toán sơ tán ngẫu nhiên hai giai đoạn bằng bài toán luồng chi phí tối thiểu:

Vấn đề cơ bản khi thảm họa xảy ra là giảm thiểu tổn thất về tính mạng. Trước một số thảm họa như động đất, rò rỉ hóa chất nguy hiểm và bão, người dân ở vùng bị ảnh hưởng phải sơ tán đến khu vực an toàn. Wang, Yang, Gao, Li, và cộng sự (2016) đã giải quyết các vấn đề sơ tán trong quản lý ứng phó thiên tai bằng cách xây dựng ba mô hình quy hoạch toán học ngẫu nhiên tùy theo quyết định của mỗi người. Các quyết định được thực hiện trong bài báo này không chỉ liên quan đến việc tối ưu hóa quyết định tiên nghiệm mà còn cả chiến lược chọn thích ứng tùy thuộc vào một tập hợp các tình huống rời rạc. Do đó, bài báo này xây dựng một bài toán quy hoạch số nguyên hỗn hợp ngẫu nhiên hai giai đoạn, cung cấp chiến lược ứng phó khẩn cấp trước-sơ-tán khi xảy ra động đất hoặc các thảm họa khác.

Bài báo này xây dựng kế hoạch di chuyển chi tiết của người dân từ khu vực nguy hiểm đến khu vực an toàn như một bài toán về luồng chi phí tối thiểu với thời gian và sức chứa ngẫu nhiên. Mục tiêu là di chuyển người dân từ vùng nguy hiểm đến vùng an toàn với thời gian sơ tán tối thiểu trong network sức chứa-chi phí  $G(V, A, C, U, D)$ , trong đó  $V$  là tập hợp các nút,  $A$  là tập hợp các liên kết có thời gian và sức chứa ngẫu nhiên,  $C(i, j)$  biểu thị thời gian di chuyển trên liên kết  $(i, j) \in A$ , ký hiệu là  $c_{ij}$ ,  $U(i, j)$  đại diện cho sức chứa của liên kết  $(i, j)$ , ký hiệu là  $u_{ij}$  và  $D(i)$  đại diện cho luồng tại nút  $i \in V$ , ký hiệu là  $d_i$ . Giả sử rằng thời gian di chuyển trên liên kết  $(i, j) \in A$  đều giống nhau nếu số người di chuyển trên đường không vượt quá sức



Hình 5: Kế hoạch sơ tán ngẫu nhiên hai giai đoạn trong network phụ thuộc thời gian.

chứa. Nghĩa là, thời gian di chuyển của liên kết không thay đổi theo số lượng người. Trong bối cảnh của vấn đề đã nêu, nút nguồn trong network đại diện cho các khu vực nguy hiểm, nút đích đại diện cho các khu vực an toàn và các nút khác là các nút trung gian của network. Các liên kết trong network đại diện cho các con đường.

Mô hình này khác với mô hình bài toán tối thiểu thường thấy. Cụ thể, mô hình quy hoạch ngẫu nhiên hai giai đoạn với truy đòi thể hiện một tình huống trong đó cả giai đoạn đầu tiên và giai đoạn thứ hai phát sinh ở các giai đoạn thời gian khác nhau trong cùng một network sơ tán. Trong giai đoạn đầu tiên, thời gian và sức chứa của liên kết chỉ được biết theo xác suất nhưng những người bị ảnh hưởng phải được sơ tán từ các nút nguồn đến các nút khác, tiên nghiệm thời gian di chuyển và sức chứa trong giai đoạn thứ hai. Ở giai đoạn thứ hai, kế hoạch sơ tán sẽ được xác định cùng với sự xác định thời gian di chuyển của liên kết và sức chứa. Lưu ý rằng kế hoạch sơ tán giai đoạn đầu có thể không khả thi đối với thông tin đã được xác định, tổng giai đoạn thứ hai thì kế hoạch sẽ được giải quyết với thời gian sơ tán ngắn hơn. Trong trường hợp này, hàm mục tiêu sẽ bao gồm chi phí phạt ở giai đoạn đầu tiên và giá trị kỳ vọng của chi phí truy đòi ở giai đoạn thứ hai.

### 2.1.3 Xây dựng mô hình

\* **Ký hiệu:** Để thuận tiện cho việc lập mô hình, Bảng 1 tóm tắt các ký hiệu liên quan được sử dụng trong công thức toán học.

**Bảng 1.** Chỉ số và tham số được sử dụng trong công thức toán học.

Ký hiệu	Định nghĩa
$V$	tập hợp các nút
$A$	tập hợp các liên kết
$i, j$	chỉ số của nút, $i, j \in V$
$(i, j)$	chỉ số của liên kết có hướng, $(i, j) \in A$
$s$	chỉ số của tình huống
$S$	tổng số tình huống
$v$	giá trị cung cấp của nút nguồn
$\tilde{T}$	ngưỡng thời gian
$T$	tổng số khoảng thời gian
$u_{ij}$	sức chứa của liên kết $(i, j)$
$u_{ij}^s(t)$	sức chứa của liên kết $(i, j)$ trong tình huống $s$ tại thời điểm $t$
$c_{ij}^s(t)$	thời gian di chuyển của liên kết $(i, j)$ trong tình huống $s$ tại thời điểm $t$
$\mu_s$	xác suất trong tình huống $s$

**Bảng 2.** Các biến quyết định được sử dụng trong công thức toán học.

Biến quyết định	Định nghĩa
$x_{ij}$	luồng trên liên kết $(i, j)$
$y_{ij}^s(t)$	luồng trên liên kết $(i, j)$ trong tình huống $s$ tại thời điểm $t$

\* **Các biến quyết định:**

#### Giai đoạn đầu tiên

Trong giai đoạn đầu tiên, con đường sơ tán có thể thực hiện được cần phải được xác định từ siêu-nguồn đến siêu-đích. Luồng trên mỗi liên kết phải thỏa mãn ràng buộc cân bằng luồng dưới đây:

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

trong đó  $d_i$  là tham số với định nghĩa dưới đây:

$$d_i = \begin{cases} v, & i = s \\ -v, & i = t \\ 0, & \text{các trường hợp còn lại} \end{cases}$$

Trong khi đó, luồng trên mỗi liên kết cũng phải thỏa mãn ràng buộc sức chứa:

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

Lưu ý, ràng buộc cân bằng luồng có thể tạo ra một đường đi với các vòng lặp và các chuyến đi phụ (sub-tour) nếu có các vòng lặp tiềm ẩn trong network sơ tán. Để loại bỏ các vòng lặp trên đường sơ tán vật lý được tạo, hình phạt liên kết  $p_{ij}$ ,  $(i, j) \in A$  được giới thiệu cụ thể. Với xem

xét này, hàm phạt có thể được định nghĩa như sau:

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

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

### Giai đoạn thứ hai

Giai đoạn thứ hai là sự đánh giá giai đoạn đầu tiên để đạt được các kế hoạch sơ tán tối ưu. Ở giai đoạn này, những người bị ảnh hưởng sẽ nhận các đường đi thích ứng với thời gian sơ tán tối thiểu theo thông tin thẩm họa thời gian thực. Trước thời điểm ngưỡng, những người bị ảnh hưởng sẽ được sơ tán theo kế hoạch sơ tán tiên nghiệm trong giai đoạn một; nói cách khác, kế hoạch sơ tán ở các tình huống khác nhau trước ngưỡng thời gian đều giống như đường đi tiên nghiệm. Theo như thảo luận ở trên, các ràng buộc ghép nối cho phương án sơ tán của từng tình huống trước ngưỡng thời gian  $\tilde{T}$  có thể được xây dựng như sau:

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

Ràng buộc ghép nối thể hiện mối quan hệ giữa liên kết vật lý và vòng cung không-thời gian trong kế hoạch sơ tán, nghĩa là nếu có các luồng lưu lượng trên liên kết  $(i, j)$ , chẳng hạn như  $x_{ij} = 2$ , thì luồng lưu lượng trên mỗi liên kết  $(i, j)$  trước ngưỡng thời gian  $\tilde{T}$  là 2, tức là  $y_{ij}^s(t) = 2$ . Nói cách khác, trước ngưỡng thời gian  $\tilde{T}$ , phương án sơ tán trong mỗi tình huống của giai đoạn thứ hai giống như kế hoạch sơ tán tiên nghiệm ở giai đoạn đầu tiên.

Sau đây, một mô hình giai đoạn hai được thiết lập với mục tiêu giảm thiểu tổng thời gian sơ tán người bị ảnh hưởng từ khu vực nguy hiểm đến khu vực an toàn trong từng tình huống:

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

$$\begin{aligned} & \text{s. t.} \\ & \sum_{(i_i, j') \in A_s} y_{ij'}^s(t) - \sum_{(i_t, i_t) \in A_s} y_{ij}^s(t') = d_i^s(t), \forall i \in V, t \in \{0, 1, \dots, T\}, \\ & s = 1, 2, \dots, S \end{aligned} \quad (6)$$

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

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

Hàm mục tiêu (5) là thời gian sơ tán tổng thể của tất cả các phương tiện giao thông trong tình huống giảm thiểu  $s$ . Các ràng buộc (6) và (7) là luồng ràng buộc cân bằng và ràng buộc sức chứa giao thông tương ứng. Ràng buộc (8) là một ràng buộc ghép nối để đảm bảo rằng sơ đồ sơ tán trong từng tình huống ở giai đoạn 2 trước ngưỡng thời gian  $\tilde{T}$  giống như kế hoạch tiên nghiệm ở giai đoạn đầu.

### \* Mô hình kế hoạch sơ tán ngẫu nhiên hai giai đoạn:

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 thích ứng ở giai đoạn thứ hai. Để đạt được mục tiêu này, bài báo đánh giá kế hoạch sơ tán giai đoạn một với thời gian sơ tán tổng thể dự kiến của đường sơ tán thích ứng của từng tình huống và xác suất xảy ra của từng tình huống  $s$  được mặc định là  $\mu_s, s = 1, 2, \dots, S$ . Để giảm thiểu mức phạt cho kế hoạch sơ tán tiên nghiệm và tổng thời gian sơ tán dự kiến của mỗi kế hoạch sơ tán thích ứng theo tình huống, mô hình kế hoạch sơ tán hai

giai đoạn trong môi trường ngẫu nhiên và phụ thuộc thời gian được xây dựng như sau:

$$\left\{ \begin{array}{l} \min \sum_{(i,j) \in A} p_{ij} x_{ij} + \sum_{s=1}^S \mu_s \cdot Q(Y, s) \\ s.t. \\ \sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} = d_i, \forall i \in V \\ 0 \leq x_{ij} \leq u_{ij}, \forall (i, j) \in A \\ \text{trong đó,} \\ Q(Y, s) = \min \sum_{(i,j) \in A_s} c_{ij}^s(t) \cdot y_{ij}^s(t) \\ s.t. \\ \sum_{(i_t, j_{t'}) \in A_s} y_{ij}^s(t) - \sum_{(i_{t'}, i_t) \in A_s} y_{ij}^s(t') = d_i^s(t), \forall i \in V, \\ t \in \{0, 1, \dots, T\}, s = 1, 2, \dots, S \\ 0 \leq y_{ij}^s(t) \leq u_{ij}^s(t), \forall (i, j) \in A, t \in \{0, 1, \dots, T\}, s = 1, 2, \dots, S \\ \sum_{t \leq \tilde{T}} y_{ij}^s(t) = x_{ij}, (i, j) \in A, s = 1, 2, \dots, S \end{array} \right. \quad (9)$$

Mô hình này là mô hình kết hoạch sơ tán phụ thuộc thời gian và ngẫu nhiên hai giai đoạn. Mục tiêu của mô hình là phát triển một kế hoạch sơ tán tối ưu mạnh mẽ nhằm cung cấp hướng dẫn cho những người bị ảnh hưởng trong sự kiện khẩn cấp. Vì giai đoạn thứ hai của mô hình có số lượng tình huống hạn chế, mô hình lập kế hoạch sơ tán hai giai đoạn ngẫu nhiên và phụ thuộc vào thời gian ở trên tương đương với mô hình đơn giai đoạn sau đây:

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

Rõ ràng, so với mô hình tất định, mô hình này đặc biệt được cung cấp trong môi trường ngẫu nhiên. Nghĩa là, cả thời gian di chuyển và sức chứa trên mỗi liên kết đều là các biến ngẫu nhiên dựa trên tình huống. Nếu thời gian và sức chứa của liên kết bị suy biến thành hằng số, tức là  $S = 1$ , thì mô hình hai giai đoạn ngẫu nhiên này tương đương với một mô hình tất định.

Ngoài ra, nếu thông tin theo thời gian thực đã được cập nhật từ nút ban đầu, tức là kế hoạch sơ tán tiên nghiệm là không cần thiết để xem xét, thì mô hình sơ tán dựa trên wait-and-see (WAS) có thể được xây dựng để giảm thiểu tổng thời gian di chuyển dự kiến trong nhiều tình huống, được thể hiện như sau:

$$\left\{ \begin{array}{l} \min \sum_{s=1}^S \left( \mu_s \cdot \sum_{(i,j) \in A_s} c_{ij}^s(t) \cdot y_{ij}^s(t) \right) \\ s.t. \\ \sum_{(i_t, j_{t'}) \in A_s} y_{ij}^s(t) - \sum_{(j_{t'}, i_t) \in A_s} y_{ij}^s(t') = d_i^s(t), \forall i \in V, \\ t \in \{0, 1, \dots, T\}, s = 1, 2, \dots, S \\ 0 \leq y_{ij}^s(t) \leq u_{ij}^s(t), \forall (i, j) \in A, t \in \{0, 1, \dots, T\}, s = 1, 2, \dots, S \end{array} \right. \quad (11)$$

Mô hình (11) được ký hiệu là WAS. Về bản chất, miền khả thi của mô hình WAS rõ ràng chứa mô hình gốc, và do đó mô hình WAS này là giới hạn dưới của mô hình gốc. Do các ràng buộc giữa các tình huống khác nhau không có mối quan hệ nào nên mô hình cũng có thể được phân tách thành  $S$  các bài toán con dựa trên tình huống. Đối với mỗi tình huống của mô hình



này, về cơ bản nó là bài toán luồng chi phí tối thiểu phụ thuộc vào thời gian.

#### 2.1.4 Thuật toán

Mô hình (10) về cơ bản là một mô hình quy hoạch số nguyên, chứa hai loại biến quyết định, tức là  $\mathbf{X} := \{x_{ij}\}_{(i,j) \in A}$  và  $\mathbf{Y} := \{y_{ij}^s(t)\}_{(i,j) \in A, t \in \{0,1,\dots,T\}, s=1,2,\dots,S}$ . Trong mô hình này, ràng buộc ghép là một ràng buộc phức tạp dẫn đến mô hình không thể thực hiện được trong thời gian đa thức. Vì vậy, phần này có ý định nối lỏng các ràng buộc ghép vào hàm mục tiêu bằng phương pháp Lagrangian relaxation. Sau đây, Phần 4.1 cung cấp hướng tiếp cận kép dựa trên Lagrangian relaxation để phân tích bài toán ban đầu thành hai bài toán con để giải quyết, có giá trị mục tiêu là giới hạn dưới của giá trị tối ưu của mô hình (10). Cụ thể, mô hình ban đầu được phân tách thành bài toán luồng chi phí tối thiểu tiêu chuẩn và bài toán phụ thuộc thời gian, có thể được giải quyết một cách hiệu quả bằng các thuật toán chính xác. Vấn đề then chốt của phương pháp Lagrangian relaxation là tìm ra giới hạn trên của vấn đề được quan tâm.

##### \* Phân rã mô hình

Từ mô hình ban đầu có thể thấy rằng ràng buộc ghép (4) là một ràng buộc cứng. Ràng buộc này đặc trưng cho mối quan hệ giữa việc lựa chọn một liên kết vật lý và các cung phụ thuộc thời gian dựa trên tình huống tương ứng. Do đó, chúng tôi giới thiệu phương trình nhân Lagrangian  $\alpha_{ij}^s(t)$ ,  $(i,j) \in A$ ,  $s = 1, 2, \dots, S$ ,  $t \leq T$  cho ràng buộc ghép, và sau đó ràng buộc này có thể được nối lỏng thành hàm mục tiêu dưới dạng sau:

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

Do đó, sau khi rút gọn công thức (12) về hàm mục tiêu, hàm relaxed của mô hình (10) có thể được biểu diễn như sau:

$$\left\{ \begin{array}{l} \min \sum_{(i,j) \in A} p_{ij} x_{ij} + \sum_{s=1}^S \left( \mu_s \cdot \sum_{(i,j) \in A_s} c_{ij}^s(t) \cdot y_{ij}^s(t) \right) + \\ \sum_{s=1}^S \sum_{t \leq T} \sum_{(i,j) \in A} \alpha_{ij}^s(t) (y_{ij}^s(t) - x_{ij}) \\ \text{s. t.} \\ \sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ij} = d_i, \forall i \in V \\ 0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in A \\ \sum_{(i,j) \in A_s} y_{ij}^s(t) - \sum_{(j,i) \in A_s} y_{ij}^s(t') = d_i^s(t), \forall i \in V, \\ t \in \{0, 1, \dots, T\}, s = 1, 2, \dots, S \\ 0 \leq y_{ij}^s(t) \leq u_{ij}^s(t), \forall (i,j) \in A, t \in \{0, 1, \dots, T\}, s = 1, 2, \dots, S \end{array} \right. \quad (13)$$

Điều đáng lưu ý là các biến  $\mathbf{X}$  và  $\mathbf{Y}$  có thể được tách rời với nhau trong mô hình relaxed ở trên (13). Tức là bằng cách kết hợp các thuật ngữ tương tự nhau, mô hình relaxed được phân tách thành hai bài toán con như sau:

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

Rõ ràng, bài toán con đầu tiên có thể được coi là bài toán luồng chi phí tối thiểu và dạng của nó được cho như sau:

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

Hàm mục tiêu của Bài toán con 1 có thể được định nghĩa là  $g_{ij} := p_{ij} - \sum_{s=1}^S \sum_{t \leq \tilde{T}} \alpha_{ij}^s(t)$  để biểu thị chi phí tổng quát của mỗi liên kết. Do đó, Bài toán con 1 có thể được giải bằng thuật toán đường đi ngắn nhất. Để thuận tiện, giá trị mục tiêu tối ưu của Bài toán con 1 được viết tắt là  $Z_{SP1}^*(\alpha)$ . Con đường ngắn nhất liên tiếp thuật toán thuật toán đường đi ngắn nhất được đề xuất bởi Jewell (1962), Iri (1960), Busacker và Gowen (1961), và cộng sự. Mục tiêu của thuật toán này là tính toán luồng chi phí tối thiểu trong network  $G = (V, A, C, U, D)$ . Cuối cùng, thuật toán đường đi ngắn nhất liên tiếp cho Bài toán con 1 được phát triển trong Thuật toán 1.

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

Bài toán con thứ hai của mô hình relaxed (13) liên quan đến các biến quyết định  $\mathbf{Y}$  và giá trị mục tiêu tối ưu của bài toán được viết tắt là  $Z_{SP2}^*(\alpha)$ , được thể hiện như sau:

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

### \* Thuật toán 1: Thuật toán đường đi ngắn nhất liên tiếp cho bài toán luồng chi phí tối thiểu.

**Bước 1:** Lấy biến  $x$  làm luồng khả thi giữa bất kỳ OD nào và nó có giá trị phân phối tối thiểu trong các luồng khả thi có cùng giá trị dòng chảy.

**Bước 2:** Thuật toán sẽ kết thúc nếu giá trị luồng của  $x$  đạt giá trị  $v$  hoặc không có đường đi chi phí tối thiểu trong phần network còn lại  $G = (V, A(x), C(x), U(x), D)$ ; mặt khác, con đường ngắn nhất với luồng tối đa được tính bằng thuật toán hiệu chỉnh nhãn, sau đó chuyển sang Bước 3. Các hàm  $A(x), C(x), U(x)$  trong network còn lại có thể được định nghĩa là:

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

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

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

**Bước 3:** Tăng luồng dọc theo đường chi phí tối thiểu. Nếu giá trị dòng chảy tăng lên không vượt quá giá trị  $v$ , chuyển sang Bước 2. Bài toán con 2 có thể được phân tách thành tổng  $S$  bài toán con, mỗi bài toán con trong số đó có thể được gọi là bài toán luồng chi phí tối thiểu với thời gian và sức chứa của liên kết phụ thuộc vào thời gian, cụ thể là:

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

Với mỗi tình huống  $s \in \{1,2,\dots,S\}$ , bài toán con (16) có cách giải tương tự cấu trúc như bài toán con (14) với chi phí liên kết phụ thuộc thời gian  $c_{ij}^s(t)$  và sức chứa liên kết  $u_{ij}^s(t)$  và chi phí tổng quát  $g_{ij}^s(t)$ . Kể từ khi xem xét khoảng thời gian  $T$  được chia thành hai giai đoạn thời

gian, chi phí tổng quát được định nghĩa là hàm từng phần:

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

Vì bài toán con (16) là bài toán luồng chi phí tối thiểu phụ thuộc vào thời gian, do đó thuật toán 1 nên được sửa đổi ở Bước 2. Đầu tiên, các tham số  $A(y(t))$ ,  $C(y(t))$ , và  $U(y(t))$  trong network còn lại  $N(y(t))$  được định nghĩa như sau:

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

Thứ hai, thuật toán sửa nhãn đã được sửa đổi ([Ziliaskopoulos & Mahmassani, 1992](#)) sẽ được áp dụng để tìm đường đi chi phí tối thiểu phụ thuộc vào thời gian trong network còn lại.

Bằng cách giải bài toán con (14) và (15) bằng nghiệm relaxed  $\mathbf{X}$  và  $\mathbf{Y}$ , giá trị mục tiêu tối ưu  $Z_{LR}^*$  cho mô hình relaxed (13) với một tập vectơ nhân Lagrange  $\alpha$  đã cho có thể được biểu diễn dưới dạng sau:

$$Z_{LR}^*(\alpha) = Z_{SP1}^*(\alpha) + Z_{SP2}^*(\alpha) \quad (17)$$

Rõ ràng, giá trị mục tiêu tối ưu của mô hình relaxed (13) là giới hạn dưới của giá trị mục tiêu tối ưu của mô hình ban đầu (10). Để có được nghiệm chất lượng cao cần có giới hạn dưới gần với giá trị mục tiêu tối ưu của mô hình ban đầu. Nghĩa là, phải đạt được giới hạn dưới lớn nhất có thể và biểu thức được đưa ra như sau:

$$Z_{LD}(\alpha^*) = \max_{\alpha \geq 0} Z_{LR}(\alpha) \quad (18)$$

## 2.2 Thuật toán Successive Shortest Path

**Thuật toán Successive Shortest Path** là một thuật toán đồ thị để giải bài toán luồng chi phí tối thiểu (Min-cost flow problem). Nó là phần mở rộng của thuật toán Ford-Fulkerson - thuật toán tìm luồng tối đa trong đồ thị với chi phí trên các cạnh.

Thay vì tìm kiếm luồng cực đại như thông thường, luồng từ  $s$  (source) đến  $t$  (sink) được gửi dọc theo đường đi ngắn nhất (liên quan đến chi phí cạnh). Phần còn lại của mạng (network) sẽ được cập nhật, đường đi ngắn nhất khác được tìm thấy và luồng lại được thêm vào,... Thuật toán kết thúc khi mạng dư không tìm thấy đường đi từ  $s$  đến  $t$  (luồng đạt cực đại). Vì luồng đạt cực đại nên nó tương ứng với giải pháp khả thi cho bài toán luồng chi phí tối thiểu ban đầu. Hơn nữa, giải pháp này sẽ tối ưu.

Thuật toán thực hiện tối đa  $O(nB)$  phép tăng, trong đó  $B$  được gán cho giới hạn trên (upper bound) của nguồn cung lớn nhất của bất kỳ node nào. Mỗi lần tăng thêm sẽ làm giảm sức chứa của cạnh nguồn (bằng với nguồn cung của node tương ứng) ít nhất một đơn vị. Sử dụng thuật toán  $O(nm)$  để tìm đường đi ngắn nhất, độ phức tạp của thuật toán successive shortest path là  $O(n^2mB)$ .

Tất cả các chi phí cạnh có thể được coi là không âm bằng cách sử dụng thuật toán Bellman-Ford. Vì việc tính toán chi phí còn lại không làm thay đổi đường đi ngắn nhất nên ta sẽ làm việc với mạng đã chuyển đổi và sử dụng thuật toán Dijkstra để tìm đường đi ngắn nhất liên tiếp hiệu quả hơn. Để giữ cho chi phí cạnh không âm, thể node và chi phí giảm được cập nhật ngay sau khi tìm thấy đường đi ngắn nhất. Với mỗi  $i$  (chỉ số của node) trong  $V$  (tập hợp các vector) thì thể  $p_i$  (thể của node  $i$ ) bằng độ dài đường đi ngắn nhất từ  $s$  tới  $i$ . Sau khi giảm chi phí của mỗi cạnh, tất cả các cạnh dọc theo đường đi ngắn nhất từ  $s$  tới  $i$  sẽ có chi phí bằng 0 trong khi các cạnh nằm ngoài đường đi ngắn nhất tới bất kỳ điểm nào đều có chi phí dương.

### Successive shortest path:

1. Chuyển đổi network  $G$  bằng cách thêm source  $s$  và sink  $t$ ;
2. Khởi tạo luồng  $x:=0$ ;
3. Sử dụng thuật toán Bellman-Ford để tìm thể  $p$ ;
4. **Reduce Cost** ( $p$ );
5. **While** ( $G_x$  chứa đường đi từ  $s$  đến  $t$ ) **do**
  6. Sử dụng thuật toán Dijkstra tìm đường đi ngắn nhất  $P$  từ  $s$  đến  $t$ ;
  7. **Reduce Cost** ( $p$ );
  8. Thêm luồng hiện tại  $x$  dọc theo  $P$ ;
  9. Cập nhật  $G_x$ ;
10. Quay lại bước 3

**Reduce Cost ( $p$ ):**

1. **For** (mỗi  $(i,j)$  trong  $G_x$ ) **do**
2.        $b_{ij}^p := b_{ij} + p_i - p_j$ ;
3.        $b_{ji}^p := 0$ ;

( $b_{ij}$ : Chi phí trên một đơn vị luồng trên cạnh  $(i,j)$ ;  $b_{ij}^p$ : Chi phí giảm của cạnh  $(i,j)$ )

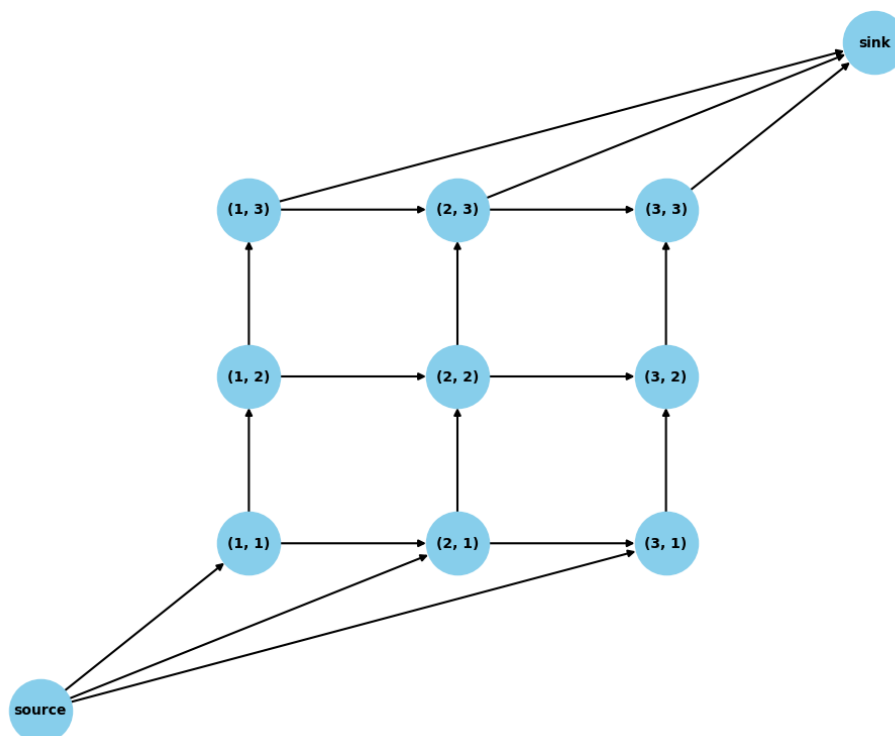
Trước khi bắt đầu chu trình ở dòng 5, thế nút được tính toán và tất cả chi phí đều không âm. Ở dòng 6, thuật toán Dijkstra được sử dụng để thiết lập đường đi ngắn nhất nhằm giảm chi phí.

Khi đó chi phí sẽ giảm và luồng được tăng thêm dọc theo đường đi. Sau khi thêm, tất cả các chi phí sẽ không âm và trong lần lặp tiếp theo, thuật toán Dijkstra sẽ hoạt động chính xác. Thuật toán Bellman-Ford chỉ được sử dụng một lần để tránh chi phí các cạnh âm. Nó mất thời gian là  $O(nm)$ . Sau đó, thuật toán Dijkstra mất  $O(nB)$ , mất thời gian  $O(n^2)$  (thực hiện đơn giản) hoặc  $O(m \log n)$  (thực hiện hàm băm cho mạng rời rạc). Tổng lại ta được  $O(n^3B)$  thời gian cho thực hiện đơn giản và  $O(nm \log n)$  nếu sử dụng hàm băm.

## 2.3 Demo thuật toán

### 2.3.1 Cách thức demo

Trong bài báo cáo này, nhóm chúng em tiến hành hành kiểm thử với mạng lưới grid 3x3, có source và sink node nối liền với các node thuộc cạnh trên và dưới.



Hình 6: Mạng lưới grid 3x3

Để thực hiện demo giải thuật này, ta sẽ sử dụng 3 package:

- **networkx**: để thao tác và đọc đồ thị.
- **matplotlib.pyplot**: để vẽ đồ thị.
- **random**: để tạo các giá trị ngẫu nhiên.

Đầu tiên, ta cần phải **import** các package và khởi tạo các giá trị ngẫu nhiên:

```

1 import networkx as nx
2 import matplotlib.pyplot as plt
3 import random
4 random.seed(1)

```

Program 6: Import package và khởi tạo giá trị ngẫu nhiên

Tiếp theo, ta sẽ khởi tạo kích thước đồ thị và kích thước lưới 3x3:

```
1 # plot size
2 plt.rcParams['figure.figsize'] = [12, 10])
3
4 grid_size = 3 # Kích thước lưới
5 rows = grid_size + 1
6 cols = grid_size + 1
7 G = nx.DiGraph()
8
9 positions = {} # Dictionary lưu trữ vị trí các nodes
```

Program 7: Khởi tạo kích thước đồ thị và kích thước lưới

Tạo grid 3x3 với các thông số capacity, weight (run time on link) là các giá trị ngẫu nhiên:

```
1 # Thêm các cạnh theo hướng từ trái sang phải và từ trên xuống dưới
2 for i in range(1, rows):
3     for j in range(1, cols):
4         # Thêm node vào đồ thị
5         G.add_node((i, j), demand=0, pos=(i, j))
6         positions[(i, j)] = (i, j)
7
8         # Thêm cạnh đi xuống (dưới) nếu không phải là hàng cuối cùng
9         if i < rows - 1:
10            G.add_edge((i, j), (i + 1, j), weight=random.randint(1, 10), capacity=random.randint(7*grid_size, 10*grid_size))
11
12            # Thêm cạnh đi qua phải nếu không phải là cột cuối cùng
13            if j < cols - 1:
14                G.add_edge((i, j), (i, j + 1), weight=random.randint(1, 10), capacity=random.randint(7*grid_size, 10*grid_size))
```

Program 8: Tạo grid 3x3

Thêm source-sink và nối chúng với đồ thị:

```
1 # Thêm source
2 G.add_node('source', demand=0, pos=(0, 0))
3 positions['source'] = (0, 0)
4
5 # Thêm sink
6 G.add_node('sink', demand=0, pos=(rows, cols))
7 positions['sink'] = (rows, cols)
8
9 # Thêm cạnh đi từ source đến hàng đầu tiên và từ hàng cuối đến sink
10 for i in range(1, rows):
11     G.add_edge('source', (i, 1), weight=0, capacity=9999)
12     G.add_edge((i, cols - 1), 'sink', weight=0, capacity=9999)
```

Program 9: Thêm source-sink vào đồ thị

Thêm thông số demand với giá trị là 20\*"số node trên một cạnh của grid":

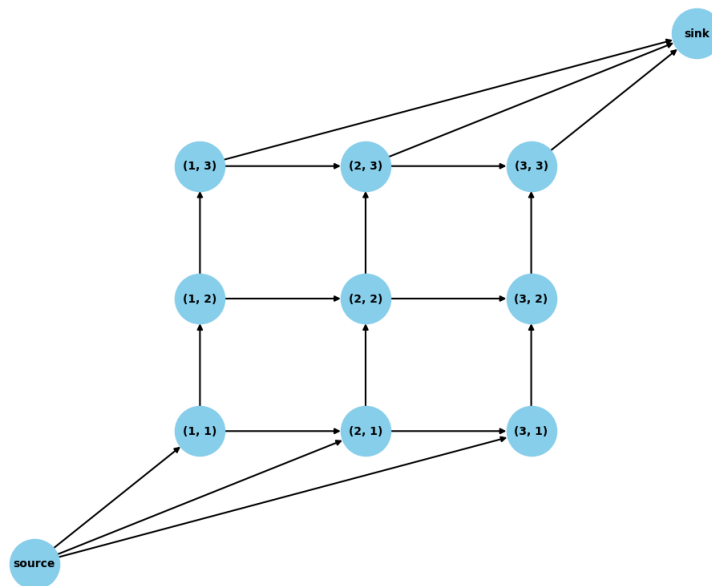
```
1 # Thêm demand cho source và sink
2 G.nodes['source']['demand'] = -15 * grid_size
3 G.nodes['sink']['demand'] = 15 * grid_size
```

Program 10: Thêm demand

Vẽ 3 đồ thị có hướng:

```
1 nx.draw(G, pos=positions, with_labels=True, node_size=2000, node_color='skyblue', font_weight='  
bold', font_color='black', font_size=10, width=1.5)  
2 nx.draw_networkx_edge_labels(G, pos=positions, edge_labels= nx.get_edge_attributes(G, 'weight'),  
font_size=15, font_color='red')  
3 nx.draw_networkx_edge_labels(G, pos=positions, edge_labels= nx.get_edge_attributes(G, 'capacity'),  
font_size=15, font_color='blue')  
4 plt.title('Directed Grid Graph\n Demand of Source: 45', fontweight='bold', fontsize=25)  
5 plt.show()
```

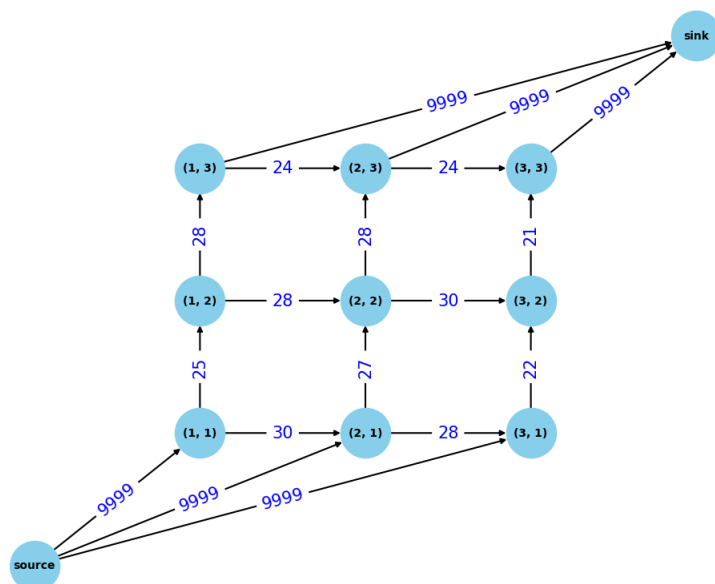
Program 11: Vẽ đồ thị



Hình 7: Đồ thị

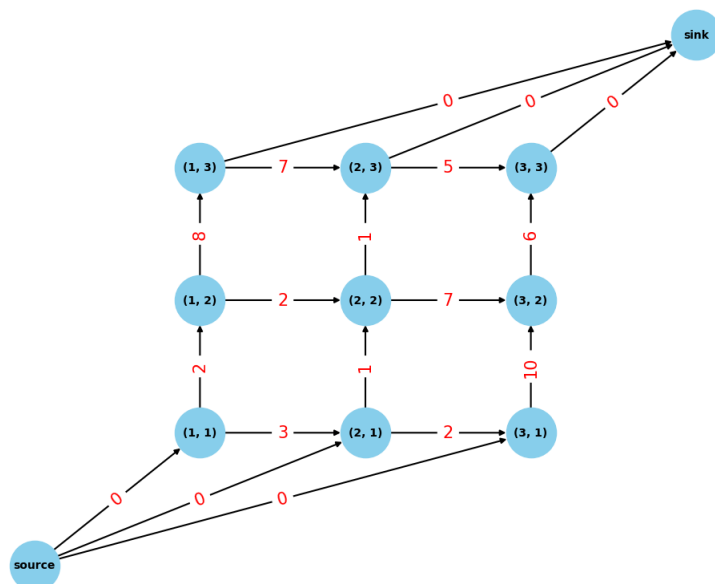


### Directed Grid Graph Demand of Source: 45



Hình 8: Đồ thị sức chứa

### Directed Grid Graph Demand of Source: 45



Hình 9: Đồ thị thời gian

```
1 # Tính toán luồng tối ưu
2 flow_cost, flow_dict = nx.capacity_scaling(G, demand='demand', capacity='capacity', weight='weight')
3
4 # Giá trị luồng tối ưu
5 print('Minimum cost:', flow_cost)
```

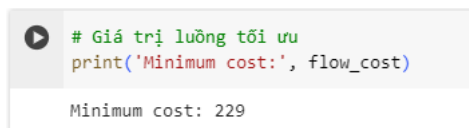
Program 12: Tính toán luồng tối ưu

```
1 # Tạo một danh sách các cạnh và trọng số tương ứng
2 edges = []
3 weights = []
4
5 for start_node, neighbors in flow_dict.items():
6     for end_node, weight in neighbors.items():
7         edges.append((start_node, end_node))
8         weights.append(weight)
9
10 # Vẽ đồ thị luồng tối ưu
11 H = nx.DiGraph()
12 for i in range(len(edges)):
13     H.add_edge(edges[i][0], edges[i][1], weight=weights[i])
14 nx.draw(H, pos=positions, with_labels=True, node_size=2000, node_color='skyblue', font_weight='bold', font_color='black', font_size=10, width=2)
15 nx.draw_networkx_edge_labels(H, pos=positions, edge_labels=nx.get_edge_attributes(H, 'weight'), font_size=15, font_color='green')
16 plt.title('Minimum Cost Flow Graph\n Demand of Source: 45', fontweight='bold', fontsize=25)
17 plt.show()
```

Program 13: Vẽ đồ thị luồng tối ưu

### 2.3.2 Kết quả demo

Sau khi chạy bài toán trên, ta tìm được giá trị luồng tối ưu là 229 và đồ thị như sau:



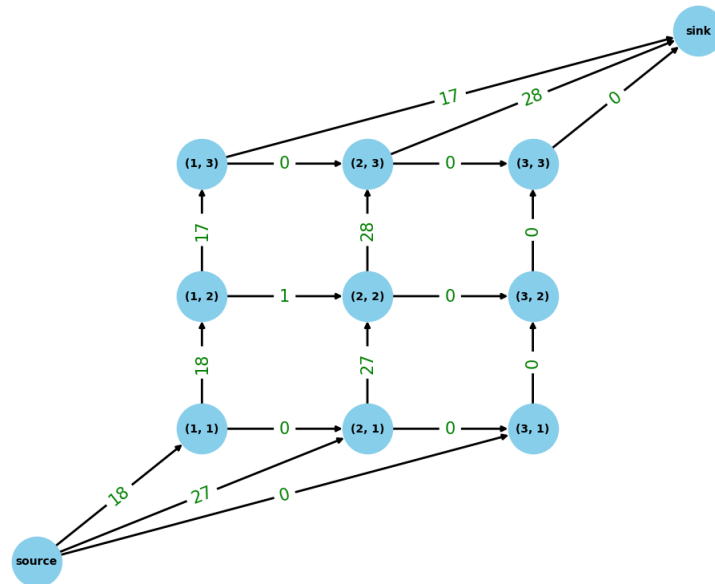
```
# Giá trị luồng tối ưu
print('Minimum cost:', flow_cost)

Minimum cost: 229
```

Hình 10: Giá trị luồng tối ưu

Giải thuật cho ra kết quả chính xác và không có lỗi khi chạy demo.

### Minimum Cost Flow Graph Demand of Source: 45

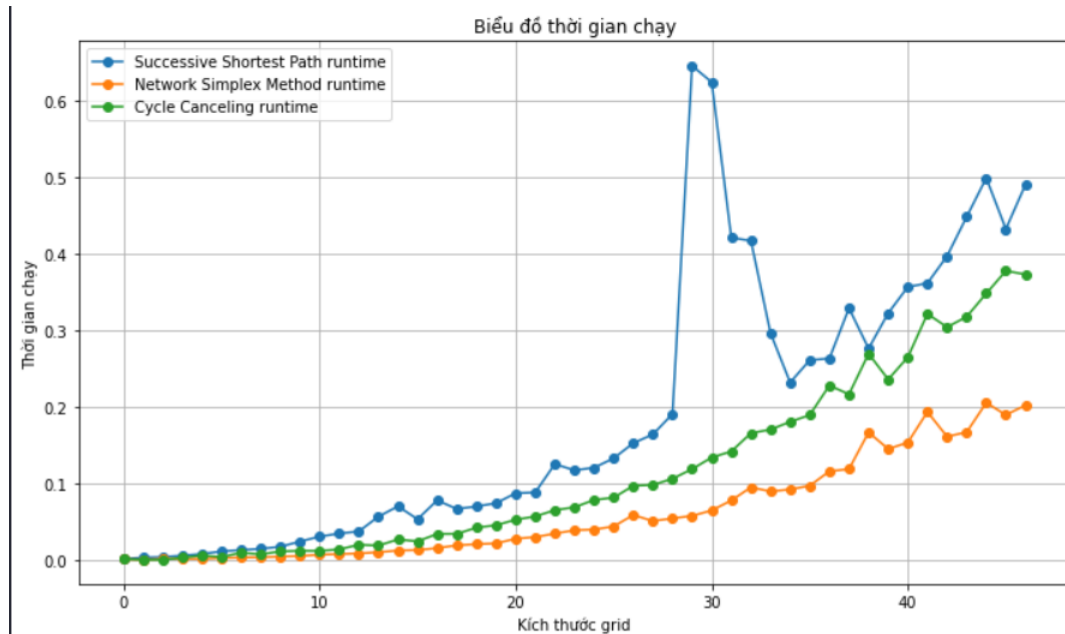


Hình 11: Đồ thị luồng tối ưu

#### 2.3.3 Độ hiệu quả của giải thuật

Để kiểm tra tính hiệu quả của giải thuật, ta tiến hành so sánh thời gian chạy và tài nguyên tiêu tốn của giải thuật với giải thuật Network Simplex Method và giải thuật Cycle Canceling. Ta so sánh bằng cách tăng kích thước của grid từ 3 đến 50, từ đó demand cũng thay đổi theo.

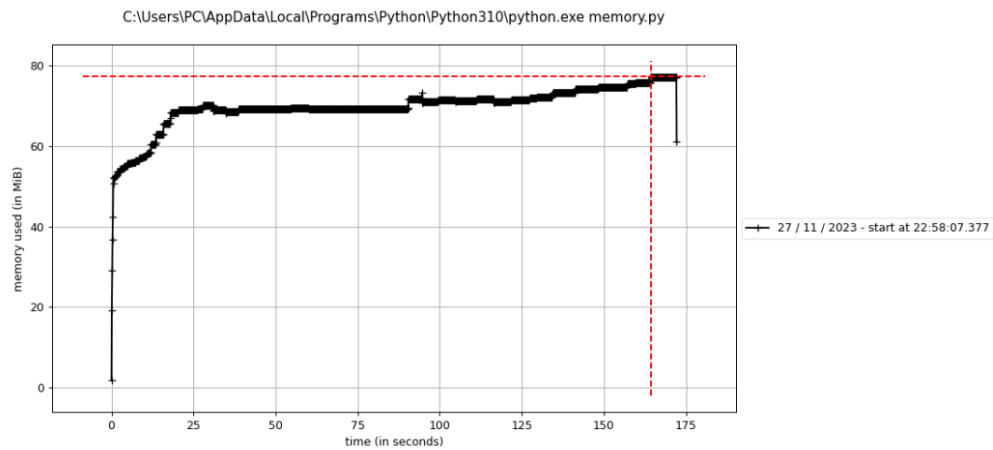
\*Thời gian chạy



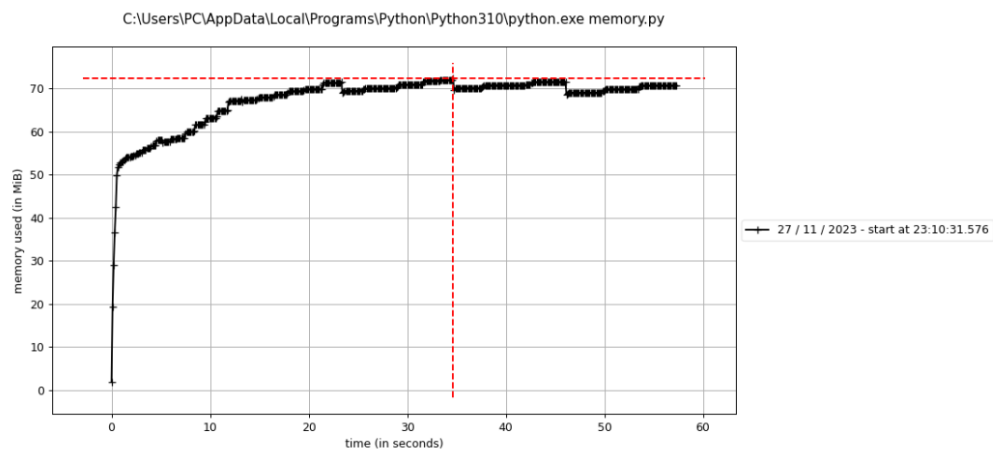
Hình 12: Biểu đồ thời gian chạy

Từ **Hình 12**, ta thấy giải thuật Successive shortest path tốn nhiều thời gian nhất, tiếp theo là giải thuật Cycle Canceling và giải thuật Network Simplex Method tốn ít thời gian nhất. Ở giải thuật Successive shortest path, khi grid có kích thước khoảng từ 29 đến 22 thì tốn thời gian hơn nhiều so với các giải thuật khác. Từ đó cho thấy sự thiếu ổn định của giải thuật.

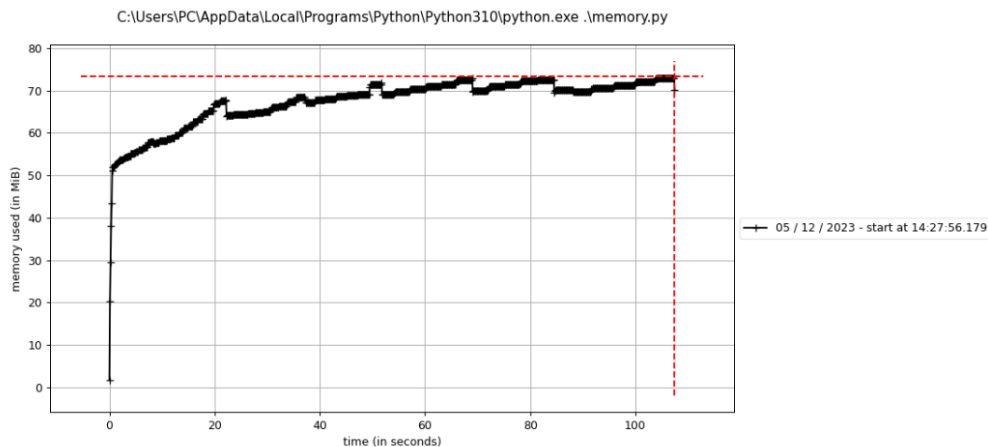
**\*Tài nguyên tiêu tốn**



Hình 13: Biểu đồ bộ nhớ sử dụng của giải thuật Successive shortest path



Hình 14: Biểu đồ bộ nhớ sử dụng của giải thuật Network simplex method



Hình 15: Biểu đồ bộ nhớ sử dụng của giải thuật Cycle canceling

Từ 3 biểu đồ trên, ta thấy giải thuật Successive shortest path tiêu tốn nhiều bộ nhớ nhất và giải thuật Network simplex method tốn ít bộ nhớ nhất.

Sau khi so sánh thời gian chạy và tiêu tốn tài nguyên thì thấy được giải thuật Successive short path chưa phải là giải thuật hiệu quả nhất.

## 2.4 Kết luận

### Ưu điểm

- Giải thuật này có thể tìm ra đường đi ngắn nhất từ một đỉnh nguồn đến tất cả các đỉnh khác trong đồ thị với độ phức tạp thời gian tốt.
- Giải thuật Successive shortest path có thể được áp dụng cho cả đồ thị có hướng và đồ thị vô hướng.
- Giải thuật này có thể xử lý các trọng số âm trong đồ thị, giúp tìm ra đường đi ngắn nhất trong trường hợp này.
- Giải thuật này dễ hiểu và triển khai, giúp cho việc sử dụng và cải tiến nó trở nên thuận tiện.

### Nhược điểm

- Mặc dù độ phức tạp thời gian của giải thuật này tốt, nhưng nó vẫn không tối ưu nhất so với một số giải thuật khác trong một số trường hợp cụ thể.
- Trong một số trường hợp, giải thuật này có thể bị mắc kẹt và không thể tìm ra đường đi ngắn nhất trong thời gian hợp lý.
- Với các đồ thị lớn và phức tạp, giải thuật Successive shortest path có thể trở nên không hiệu quả và tốn nhiều tài nguyên tính toán.

Từ những ưu và nhược điểm trên, giải thuật Successive shortest path khá phù hợp để giải bài toán đường đi ngắn nhất với các đồ thị có kích thước không quá lớn và phức tạp.

### 3 Tài liệu tham khảo

- [1]: Giordano, F. R., Fox, W. P., & Horton, S. B. (2013). *A first course in mathematical modeling*. Cengage Learning.
- [2]: Parpalea, M. (2009). Parpalea, M. (2009). Interactive tool for the successive shortest paths algorithm in solving the minimum cost flow problem. *Bulletin of the Transilvania University of Brasov. Series III: Mathematics and Computer Science*, 255-262.
- [3]: Wang, L. (2020). A two-stage stochastic programming framework for evacuation planning in disaster responses. *Computers & Industrial Engineering*, 145, 106458.
- [4]: Wang, L., Yang, L., Gao, Z., Li, S., & Zhou, X. (2016). Evacuation planning for disaster responses: A stochastic programming framework. *Transportation research part C: emerging technologies*, 69, 150-172.