**Đề bài :**

Một số thuật toán Giải pháp được đưa ra trong Phần 4 của Tài liệu tham khảo.1, và năm nay 2023 CSE - HCMUT sinh viên chỉ có thể thử cách tiếp cận đầu tiên (Thuật toán 1) dựa trên bài toán dòng chi phí tối thiểu.

Tìm hiểu, triển khai và xác minh tính hiệu quả của thuật toán đã nghiên cứu hoặc giải quyết mô hình lập kế hoạch sơ tán ngẫu nhiên hai giai đoạn chỉ trên mạng lưới nhỏ, với Thử nghiệm Cách tiếp cận thiết kế. Chính xác thì mô phỏng sẽ có tối đa 50 nút và 100 liên kết ([trong Mục 5.1 của Tài liệu tham khảo. 1])

**Giới thiệu:**

Các sự kiện cực đoan (do thiên tai hoặc do con người gây ra) như động đất, lũ lụt, lốc xoáy,… có thể ảnh hưởng nghiêm trọng đến mạng lưới và dịch vụ viễn thông ở các khu vực đô thị. Lập kế hoạch sơ tán sao cho hợp lý những thảm họa là một hành động quan trọng và cần được suy nghĩ cẩn thận (sớm) và thực hiện (vào ngày nào, thời gian nào), đặc biệt khi có sự không chắc chắn. Bài tập lớn này sẽ nghiên cứu rõ hơn bài báo của L. Wang, “A two-stage stochastic programming framework for evacuation planning in disaster responses,”(2020) về giải pháp kế hoạch sơ tán cụ thể cho những người bị ảnh hưởng từ khu vực nguy hiểm sang khu vực an toàn . Vì có rất nhiều tình huống cũng như những thiệt hại phát sinh trong thảm họa do đó vấn đề sơ tán này nên được coi là một vấn đề lập trình ngẫu nhiên. Lập trình ngẫu nhiên có hậu quả được sử dụng để tìm ra các quyết định không dự đoán được phải được thực hiện trước khi biết được các thực hiện của một số biến ngẫu nhiên sao cho tổng chi phí kỳ vọng của các hành động hậu quả có thể là nhỏ nhất. Dựa trên phân tích trên, vấn đề sơ tán trong bài tập lớn này được xây dựng thành một mô hình lập trình ngẫu nhiên hai giai đoạn dựa trên kịch bản để đại diện cho sự ngẫu nhiên xuất phát từ cường độ và ảnh hưởng.

**Mục tiêu:**

Lập kế hoạch sơ tán người dân khỏi những khu vực bị ảnh hưởng bởi các thiên tai như lũ lụt, động đất, núi lửa, ... Mô hình lập trình ngẫu nhiên hai giai đoạn mà xem xét cả lựa chọn đường sơ tán trước (trước thảm họa) và thích ứng (sau thảm họa) để cung cấp kế hoạch sơ tán trước mắt cho người bị ảnh hưởng từ các khu vực nguy hiểm sang các khu vực an toàn. Không giống như các mô hình xác định, nó liên quan đến sự khả dụng của các thời gian di chuyển và khả năng trên liên kết ngẫu nhiên dựa trên kịch bản.Trong giai đoạn thứ nhất, giả sử rằng người bị ảnh hưởng không thể thu thập được thông tin về mức độ thiên tai và mức độ hư hại của đường và sau một khoảng thời gian nhất định, họ có thể thu thập được thông tin mạng lưới đường bộ chính xác thông qua một số thiết bị giám sát thời gian thực sau đó thực hiện kế hoạch sơ tán mạnh mẽ cho tất cả các mức độ thảm họa. Các quyết định giai đoạn thứ hai (hậu quả) được phụ thuộc vào các quyết định giai đoạn đầu, được thực hiện sau khi thực hiện các thời gian di chuyển và khả năng ngẫu nhiên. Do đó, mục tiêu là làm cho kế hoạch sơ tán trước khi thực hiện việc tối ưu trong giai đoạn đầu, mà là dưới các điều kiện không chắc chắn sẽ phải đối mặt trong giai đoạn thứ hai. Hơn nữa, thay vì quyết định vị trí cơ sở và đặt trước hàng hóa cứu trợ khẩn cấp, kế hoạch này trình bày đường sơ tán cụ thể cho người bị ảnh hưởng.

**Các bước thực hiện :**

 Bước 1: Đề xuất mô hình lập trình ngẫu nhiên hai giai đoạn và lựa chọn lộ trình trước tai họa và sau tai họa để cung cấp kế hoạch sơ tán cho những người bị ảnh hưởng đến nơi an toàn.

Bước 2: Xây dựng kế hoạch di chuyển rõ ràng của những người dân bị ảnh hưởng bằng mô hình dòng chi phí tối thiểu từ đó chia thành 2 bài toán nhỏ là luồng có chi phí nhỏ nhất và bài toán phụ thuộc vào thời gian của nó.

Bước 3:  Để thu được nghiệm gần tối ưu, khung thuật toán tối ưu hóa đạo hàm riêng được sử dụng để giảm dần khoảng cách tương đối của giới hạn trên và dưới, tức là các giá trị mục tiêu của mô hình gốc và mô hình giải phóng; và hai bài toán con được giải quyết bằng thuật toán đường ngắn nhất liên tiếp.

**Mô tả bài toán :**

**Bảng kí hiệu:**

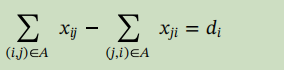
|  |  |
| --- | --- |
| Kí hiệu | Ý nghĩa |
| V | Tập hợp các nút(nodes) |
| A | Tập hợp các liên kết(links) |
| i,j | Vị trí của các nút, i,j thuộc V |
| (i,j) | Vị trí của các liên kết, (i,j) thuộc A |
| s | Số thứ tự của kịch bản |
| S | Tổng số kịch bản |
| v | Giá trị cung cấp của nút nguồn |
| T | Tổng số khoảng thời gian |
| uij | Dung lượng trên liên kết (i,j) |
| u | Dung lượng trên liên kết (i,j) trong kịch bản s ở thời gian t |
|  | Thời gian di chuyển trên liên kết (i,j) trong kịch bản s ở thời gian t |
| s | Xác suất trong kịch bản s |

|  |  |
| --- | --- |
| Biến quyết định | Ý nghĩa |
| xij | Luồng trên liên kết (i,j) |
| y | Luồng trên liên kết (i,j) trong kịch bản s ở thời gian t |

**Hệ thống các ràng buộc:**

**Giai đoạn 1:**

Trong giai đoạn đầu tiên, một sự sơ tán khả thi cần được xác định từ vùng nguy hiểm đến vùng an toàn. Mỗi luồng trên mỗi liên kết cần thỏa mãn ràng buộc về cân bằng luồng ( tổng luồng chạy vào 1 nút bằng tổng luồng chạy ra khỏi nút đó)



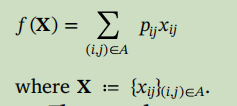
Trong đó di là tham số với :

di=

Ngoài ra luồng còn phải thỏa mãn ràng buộc về dung lượng tối đa(luồng chảy trên mỗi liên kết không vượt quá dung lượng tối đa của liên kết đó)

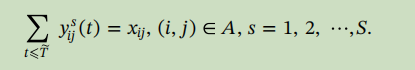
0<=xij<=uij với mọi i,j A

Lưu ý: Phạt liên kết (link penalty) pij i,jA là một giá trị được gán cho mỗi liên kết để đánh giá mức độ không mong muốn của việc sử dụng liên kết đó trong một đường dẫn sơ tán.Hàm liên kết có thể định nghĩa là

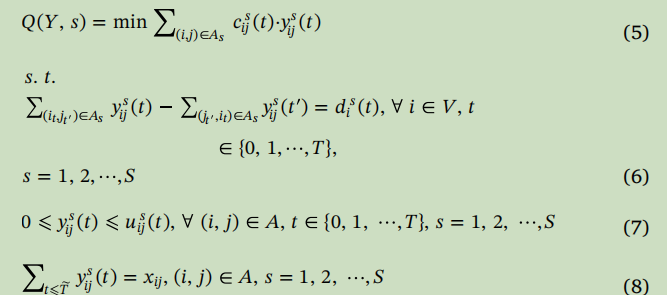


**Giai đoạn 2:**

Ở giai đoạn 2, những người bị ảnh hưởng sẽ nhận được các đường đi thích ứng với thời gian sơ tán tối thiểu theo thông tin thảm họa thời gian thực. Trước ngưỡng thời gian, những người bị ảnh hưởng sẽ được sơ tán theo quy định trước kế hoạch sơ tán giai đoạn một; nói cách khác, kế hoạch sơ tán ở các kịch bản khác nhau trước ngưỡng thời gian đều giống như tiên nghiệm con đường. Theo thảo luận ở trên, các ràng buộc ghép nối cho kế hoạch sơ tán của từng kịch bản trước ngưỡng thời gian có thể được xây dựng như sau:



Ràng buộc ghép nối thể hiện mối quan hệ giữa liên kết vật lí và vòng cung không gian- thời gian trong kế hoạch sơ tán nghĩa là nếu có các luồng truy cập trên liên kết (i,j) ví dụ xij=2 thì luồng truy cập trên mỗi liên kết trước ngưỡng thời gian là 2, y)=2. Nói cách khác, trước ngưỡng thời gian, kế hoạch sơ tán trong mỗi kịch bản của giai đoạn thứ hai giống như kế hoạch sơ tán ban đầu ở giai đoạn đầu tiên. Sau đây, một mô hình giai đoạn hai được thiết lập với mục tiêu nhằm giảm thiểu tổng thời gian di tản của những người bị ảnh hưởng khỏi khu vực nguy hiểm ở mỗi kịch bản:



Hàm mục tiêu (5) là thời gian sơ tán tổng thể của các phương tiện giao thông trong kịch bản giảm thiểu s

Ràng buộc (6) là ràng buộc cân bằng luồng

Ràng buộc (7) là ràng buộc về dung lượng tối đa

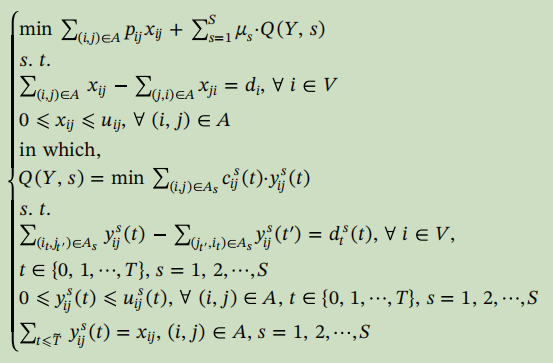
Ràng buộc (8) là một ràng buộc ghép nối để đảm bảo rằng việc sơ tán

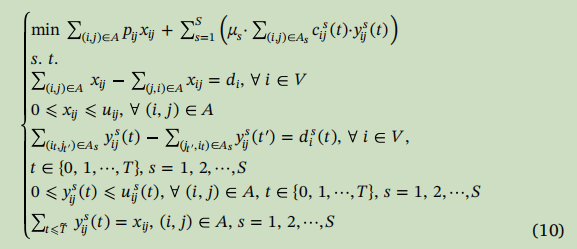
sơ đồ trong từng kịch bản ở giai đoạn thứ hai trước ngưỡng thời gian

giống như kế hoạch tiên nghiệm ở giai đoạn đầu.

**Mô hình lập kế hoạch sơ tán ngẫu nhiên hai giai đoạn**

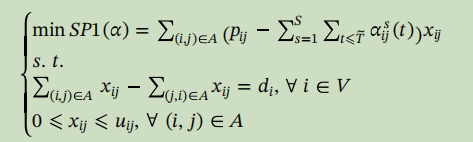
Vấn đề sơ tán được quan tâm là có được một kế hoạch vững chắc trong giai đoạn đầu tiên bằng cách đánh giá các kế hoạch thích ứng ở giai đoạn thứ hai.Để đạt được mục tiêu này, kế hoạch sơ tán giai đoạn một với thời gian sơ tán tổng thể dự kiến của đường sơ tán thích ứng của từng kịch bản và xác suất xảy ra của từng kịch bản s giả định là s,s=1,2,..S. Để giảm thiểu hình phạt đối với kế hoạch sơ tán trước đó và tổng thời gian sơ tán dự kiến của mỗi kế hoạch sơ tán thích ứng theo kịch bản, mô hình kế hoạch sơ tán hai giai đoạn trong môi trường phụ thuộc thời gian và ngẫu nhiên được xây dựng như sau:



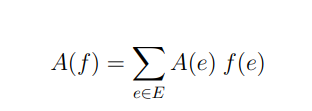


Để giải quyết mô hình 10 là một mô hình lập trình nguyên vốn chứa hai loại biến quyết định là X:={xij} và Y:={y} ta cần chia mô hình thành 2 bài toán nhỏ bằng phương pháp song phương nhằm nới lỏng ràng buộc ghép vào hàm mục tiêu bằng cách sử dụng phương pháp nới lỏng Lagrange. Cụ thể, mô hình ban đầu được phân tách thành một bài toán luồng có chi phí nhỏ nhất và một bài toán phụ thuộc vào thời gian, mà có thể được giải quyết hiệu quả bằng các thuật toán chính xác. Vấn đề then chốt của phương pháp nới lỏng Lagrange là tìm ra cận trên của bài toán quan tâm. Các giải pháp nới lỏng, ngẫu nhiên, là khả thi cho bài toán ban đầu.

**Vấn đề 1 : Bài toán luồng chi phí tối thiểu**



Cho một mạng G bao gồm n đỉnh và m cạnh mỗi cạnh có dung lượng A(e), chi phí trên mỗi luồng dọc theo cạnh là C(e) (số nguyên) và cân bằng đỉnh. Tổng chi phí của bất kì luồng f có thể định nghĩa là:



Đối với một giá trị nhất định  K , chúng ta phải tìm một luồng có đại lượng này và trong số tất cả các luồng có đại lượng này, chúng ta phải chọn luồng có chi phí thấp nhất. Nhiệm vụ này được gọi là **vấn đề luồng chi phí tối thiểu** .

Nhiệm vụ này có thể giải quyết bằng thuật toán đường đi ngắn nhất liên tiếp.

Đường đi ngắn nhất liên tiếp được xây dựng trên mạng luồng chung G=(V,E) với dung lượng f(.), giới hạn dưới, chi phí C(.) trên các cạnh và cân bằng (dòng vào bằng dòng ra tại bất kỳ nút nào). Giả sử dưới đây được thỏa mãn

1. Các cạnh có dung lượng không âm và giới hạn dưới bằng 0: 0<=f(u,v)<=c(u,v) với mọi u,v
2. Mọi luồng f phải thoả mãn ràng buộc bảo toàn số luồng vào bằng số luồng ra tại mỗi nút:

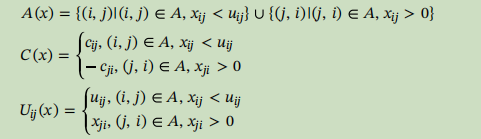


**Thuật toán đường đi ngắn nhất liên tiếp :**

Mục tiêu của bài toán này là sẽ tìm kiếm giá trị lớn nhất luồng và đồng thời tối ưu hóa hàm mục tiêu cho trước

* Bước 1: Chọn biến x làm một luồng khả thi, tức là một luồng thỏa mãn các ràng buộc về dung lượng, đối xứng, và bảo toàn luồng. Luồng này cũng phải có chi phí nhỏ nhất trong các luồng khả thi có cùng giá trị luồng.

Bước 2: Kiểm tra xem luồng đã đạt giá trị mong muốn hay chưa, hoặc có còn đường đi nào có chi phí nhỏ hơn trong mạng lưới dư hay không. Mạng lưới dư là một mạng lưới được tạo ra bằng cách thêm các cung ngược lại vào mạng lưới ban đầu, với chi phí và dung lượng phù hợp. Nếu luồng đã đạt giá trị mong muốn v, hoặc không còn đường đi nào có chi phí nhỏ hơn, thuật toán kết thúc. Nếu không, tìm đường đi ngắn nhất với luồng lớn nhất trong mạng lưới dư, bằng cách sử dụng thuật toán nhãn sửa, và sau đó chuyển sang Bước 3. Hàm A(x), C(x), U(x) trong phần còn lại mạng có thể được định nghĩa là:



* Bước 3: Tăng luồng dọc theo đường đi có chi phí nhỏ nhất, bằng cách thêm hoặc bớt luồng trên các cung thuộc đường đi đó. Nếu giá trị luồng tăng không vượt quá giá trị mong muốn, quay lại Bước 2.

Lưu ý rằng trong vòng lặp cuối cùng của thuật toán, chỉ cần tăng luồng thêm một lượng sao cho giá trị luồng cuối không vượt quá K.

**Thuật toán nhãn sửa( label correcting algorithm)**

Đầu vào: Cho G = (V, E) là một đồ thị (có thể vô hướng hoặc có hướng)

và {aij : (i, j) ∈ E} là tập hợp các trọng số không âm trên các cạnh. Chúng tôi phân biệt hai nút trong đồ thị, cụ thể là điểm gốc s ∈ V và điểm đích t ∈ V . Nút j ∈ V được gọi là con của nút i ∈ V nếu có cạnh từ i đến j. Lưu ý rằng nếu đồ thị vô hướng và (i, j) là một cạnh, thì i là con của j và j là con của i. Mục tiêu của chúng ta là tìm đường đi ngắn nhất từ s đến t

Ý tưởng đằng sau các thuật toán sửa nhãn là khám phá dần dần các đường đi ngắn hơn từ nguồn gốc của mọi nút khác i. Điều này đạt được bằng cách duy trì nhãn di cho mỗi nút i, nhãn này biểu thị giới hạn trên của khoảng cách ngắn nhất từ điểm gốc s đến nút i. Nhãn dt của đích t được duy trì trong một biến gọi là Upper. Như chúng ta sẽ thấy, biến Upper đóng vai trò một vai trò đặc biệt trong thuật toán. Khi thuật toán tiến triển, nếu một đường đi từ điểm gốc đến nút i được phát hiện và khoảng cách của nó nhỏ hơn di, thì nhãn của nút i sẽ được sửa. Để tạo thuận lợi cho việc ghi sổ, thuật toán duy trì một danh sách các nút được gọi là Open. Danh sách Open chứa các nút sẽ được kiểm tra bởi

thuật toán và là ứng cử viên có thể được đưa vào con đường ngắn nhất. Khi danh sách mở trống, thuật toán kết thúc.

Bước 1: Cho ds=0, di=∞ với mọi i ∈V \{s,t} và Upper=∞. Tạo danh sách Open={s}

Bước 2: xóa a node i khỏi Open

Bước 3: Với mỗi con j của i ta thực hiện :

-Nếu di + aij <min (dj,Upper) , dj=di+aij và cho i thành nút cha của j

-Nếu j khác đích t, thêm j vào danh sách Open nếu j chưa xuất hiện. Ngược lại nếu j là đích t, Upper = di + ait

Bước 4: Nếu danh sách Open rỗng kết thúc thuật toán nếu không quay lại bước 2

Về mặt hình thức, thuật toán sửa nhãn chung được đưa như trên. Ở bước 3 của thuật toán, chúng ta gặp một đường đi từ s đến j qua i và chúng ta kiểm tra xem đường dẫn này (i) có ngắn hơn không đường đi được xem xét trước đây từ s đến j (tức là nếu di + aij < dj ) và (ii) có khả năng dẫn đến a

đường đi ngắn nhất từ s đến t (tức là di + aij < Upper). Nếu vậy thì nhãn dj bị giảm và nút j được đặt vào danh sách Open sao cho các đường đi qua j và tới các phần tử con của j có thể được kiểm tra sau.

1.Trong suốt thuật toán, nhãn di của nút i tùy ý là ∞ (tương ứng

trong trường hợp tôi chưa vào danh sách Open) hoặc đó là độ dài của một số đường dẫn từ s đến i bao gồm các nút đã vào danh sách Open ít nhất một lần.

2. Nếu di < ∞, thì có thể thu được đường đi từ s đến i có độ dài di bằng cách truy tìm các nút cha bắt đầu với cha mẹ của i.

3. Thuật toán sẽ kết thúc sau một số bước hữu hạn. Hơn nữa, khi nó kết thúc,

biến Upper sẽ bằng khoảng cách ngắn nhất từ điểm gốc đến đích t (trường hợp này có ít nhất một đường đi từ s tới t), hoặc bằng ∞ (trường hợp này

khi không có đường đi từ s đến t).

4. Nút j mà di + aij ≥ Upper ở bước 3 chắc chắn sẽ không được đưa vào danh sách Open trong lần lặp hiện tại và có thể không được đưa vào danh sách trong bất kỳ lần lặp tiếp theo nào. Do đó, số lượng nút vào danh sách Mở có thể nhỏ hơn đáng kể so với tổng số nút. Đặc biệt, thuật toán sửa nhãn có thể được hiệu quả hơn thuật toán DP cơ bản.

Lưu ý rằng ở bước 2 của Thuật toán 1, có một số quyền tự do trong việc chọn nút nào để loại bỏ từ danh sách Open trong mỗi lần lặp. Mỗi lựa chọn tương ứng với một cách triển khai khác nhau của thuật toán sửa nhãn. Một số lựa chọn nổi tiếng hơn được đưa ra như sau:

1. (Tìm kiếm theo chiều rộng–Đầu tiên) Danh sách mở được duy trì dưới dạng hàng đợi vào trước ra trước (FIFO), tức là nút luôn được xóa khỏi đầu Open và mỗi nút vào Open được đặt tại phần dưới cùng của Open.

2. (Tìm kiếm theo chiều sâu-Đầu tiên) Danh sách mở được duy trì dưới dạng hàng đợi vào trước ra trước (LIFO), tức là nút luôn được xóa khỏi đầu Open và mỗi nút vào Open được đặt tại đỉnh của Open.

3. (Tìm kiếm tốt nhất-đầu tiên) Trong mỗi lần lặp, một nút có nhãn nhỏ nhất sẽ bị xóa khỏi danh sách. Đây được gọi là phương pháp Dijkstra. Có thể chỉ ra rằng trong phương pháp này,mỗi nút sẽ chỉ vào Open tối đa một lần

**Giải thuật cho đồ thị vô hướng, đa đồ thị**

Trong trường hợp đồ thị vô hướng hay đa đồ thị không khác nhau nhiều về mặt khái niệm của thuật toán trên. Tuy nhiên, việc thực hiện thuật toán sẽ khó khăn hơn một chút.

* Đầu tiên, luồng cho từng cạnh phải giữ riêng biệt.
* Thứ hai, khi tìm kiếm đường đi ngắn nhất, cần phải tính đến việc điều quan trọng là cạnh nào trong số nhiều cạnh được sử dụng trong đường dẫn. Do đó, thay vì mảng tổ tiên thông thường, chúng ta phải lưu trữ số cạnh mà chúng ta xuất phát cùng với mảng tổ tiên.
* Thứ ba, khi luồng tăng dọc theo một cạnh nhất định thì cần phải giảm luồng dọc theo cạnh sau. Vì chúng ta có nhiều cạnh nên chúng ta phải lưu trữ số cạnh của cạnh đảo ngược cho mỗi cạnh.

**Kết quả :**

**Đánh giá :**

Thuật toán Dijkstra yêu cầu ít lần lặp hơn Bellman-Ford.Tuy nhiên, Bellman-Ford yêu cầu ít công việc hơn cho mỗi lần lặp.Bạn có thể dừng thuật toán của Dijkstra sau khi đã xác định xong đích đến nút.

Bạn không thể sử dụng thuật toán của Dijkstra nếu có chi phí liên kết âm.

Còn Bellman-Ford vẫn hoạt động.

**Độ phức tạp:** phụ thuộc vào cấp số nhân của đầu vào, trong trường hợp xấu nhất nó có thể đẩy tối đa 1 đơn vị lưu lượng trên 1 lần lặp, lấy O(F) lặp đi lặp lại để tìm ra luồng kích thước có chi phí tối thiểu F, làm cho tổng thời gian chạy là O(F.T). T là thời gian cần để tìm đường đi ngắn nhất từ nguồn tới đích.

Nếu thuật toán [Bellman-Ford](https://cp-algorithms.com/graph/bellman_ford.html) được sử dụng cho việc này, nó sẽ làm cho thời gian chạy  là O(mn). Cũng có thể sửa đổi [thuật toán Dijkstra](https://cp-algorithms.com/graph/dijkstra.html) để nó cần  O(nm)  tiền xử lý là bước đầu tiên và sau đó hoạt động trong O(mlogn) mỗi lần lặp, làm cho thời gian tổng thể là O(mn+Fmlogn). [Đây](http://web.archive.org/web/20211009144446/https://min-25.hatenablog.com/entry/2018/03/19/235802) là một trình tạo biểu đồ, trên đó thuật toán như vậy sẽ yêu cầu  O(2n/2n2logn).

Thuật toán Dijkstra đã sửa đổi sử dụng cái gọi là tiềm năng từ [thuật toán Johnson](https://en.wikipedia.org/wiki/Johnson's_algorithm) . Có thể kết hợp ý tưởng của thuật toán này và thuật toán Dinic để giảm số lần lặp từ  F đến min(F,nC) với C là chi phí tối đa được tìm thấy giữa các cạnh.

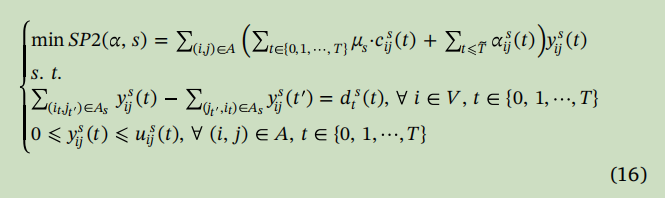
**Ưu điểm :**

* Thuật toán này có thể áp dụng cho các đồ thị có trọng số âm, miễn là không có chu trình âm. Điều này làm cho thuật toán này linh hoạt hơn so với các thuật toán chỉ áp dụng cho các đồ thị có trọng số dương, như thuật toán Dijkstra.
* Thuật toán này có thể tận dụng được các cấu trúc đặc biệt của đồ thị, như đồ thị thưa hay đồ thị đầy đủ, để cải thiện hiệu suất. Ví dụ, nếu đồ thị là thưa, ta có thể dùng cấu trúc dữ liệu hàng đợi ưu tiên để tìm đường đi ngắn nhất nhanh hơn. Nếu đồ thị là đầy đủ, ta có thể dùng ma trận để lưu trữ và cập nhật các nhãn.
* Thuật toán này có thể dễ dàng được mở rộng để giải quyết các biến thể của bài toán tối thiểu luồng chi phí, như bài toán luồng cực đại chi phí nhỏ nhất, bài toán luồng cân bằng, hay bài toán luồng có giới hạn.

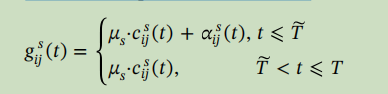
**Nhược điểm :**

* Thuật toán này chỉ áp dụng được cho các đồ thị không có chu trình âm. Nếu đồ thị có chu trình âm, thuật toán sẽ không hội tụ và không tìm được lời giải. [Do đó, ta cần kiểm tra trước xem đồ thị có chu trình âm hay không, hoặc sử dụng các thuật toán khác như Cycle canceling hay Out-of-kilter](https://vnoi.info/wiki/algo/graph-theory/shortest-path.md" \t "https://www.bing.com/_blank).
* Thuật toán này có thể bị ảnh hưởng bởi sự thay đổi của trọng số của các cạnh. Nếu trọng số của một cạnh giảm, thuật toán có thể tiếp tục chạy và cập nhật lời giải. [Nhưng nếu trọng số của một cạnh tăng, thuật toán sẽ phải khởi tạo lại và chạy lại từ đầu](https://vnoi.info/wiki/algo/graph-theory/breadth-first-search.md" \t "https://www.bing.com/_blank).

**Vấn đề 2: Bài toán luồng chi phí tối thiểu với thời gian và dung lượng di chuyển của liên kết phụ thuộc vào thời gian**

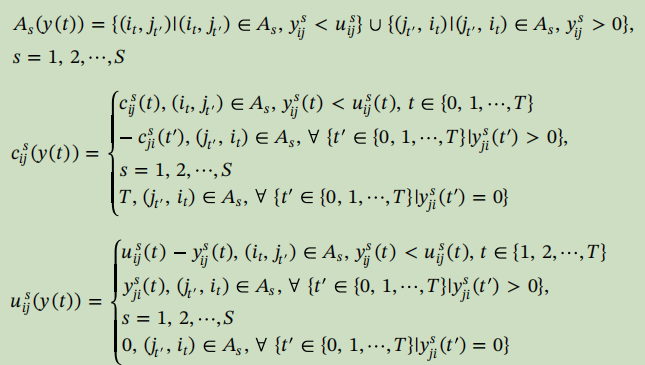


Với mỗi kịch bản s ∈{1,2,3...,S} vấn đề (16) có cùng cấu trúc với vấn đề (14) với chi phí phụ thuộc thời gian , dung lượng liên kết và chi phí tổng quát . khoảng thời gian T được chia thành hai giai đoạn thời gian, chi phí tổng quát được định nghĩa là hàm từng phần:



Vì vấn đề 2 là bài toán phụ thuộc vào thời gian vì vậy chúng ta cần phải thay đổi ở bước 2.

Đầu tiên chúng ta cần chỉnh sửa các tham số A(y(t)), C(y(t)), U(y(t)) trong mạng dư N(y(t)) như sau :



Tiếp theo chúng ta cần sửa đỗi thuật toán nhãn sửa theo (Ziliaskopoulos & Mahmassani, 1992) để tìm đường dẫn chi phí tối thiểu phụ thuộc vào thời gian trong mạng dư.

Thuật toán tìm đường ngắn nhất phụ thuộc thời gian được giới thiệu trong bài báo của (Ziliaskopoulos & Mahmassani, 1992) tính toán đường đi ngắn nhất phụ thuộc thời gian từ tất cả các nút trong mạng đến một nút đích cho mỗi bước thời gian trong một khoảng thời gian nhất định trong một mạng với chi phí cạnh phụ thuộc thời gian. Khác với các thuật toán phụ thuộc thời gian khác, phương pháp này có thể xử lý các mạng trong đó chi phí đi lại không nhất thiết phải là thời gian đi lại chính nó. Thuật toán này dựa trên nguyên tắc tối ưu của Bellman. Nó rời rạc hóa khoảng thời gian quan tâm thành các khoảng thời gian nhỏ và tính toán các đường đi hoạt động ngược lại từ nút đích.

**Thuật toán tìm đường ngắn nhất phụ thuộc thời gian**

Giải thuật này có thể được giải thích như sau:

Bước 1: Tạo một danh sách SE rỗng.

Bước 2: Chèn nút nguồn (source node) N vào danh sách SE.

Bước 3: Lặp lại cho đến khi danh sách SE trống:

Xóa nút đầu tiên (current node) khỏi danh sách SE.

Duyệt qua tất cả các nút J có thể đến được từ nút hiện tại (current node).

Gán next node bằng J.

Gán insert in SE list bằng FALSE.

Duyệt qua tất cả các bước thời gian (time step) t từ 1 đến M.

Tính thời gian di chuyển (travel time) từ next node đến current node tại bước thời gian t.

Tính nhãn mới (new label) bằng cách cộng nhãn của current node tại bước thời gian t cộng với thời gian di chuyển.

Nếu nhãn của next node tại bước thời gian t lớn hơn nhãn mới, thì cập nhật nhãn của next node bằng nhãn mới và gán insert in SE list bằng TRUE. Đồng thời, lưu lại con trỏ đường dẫn (path pointer) từ next node đến current node.

Nếu insert in SE list bằng TRUE, thì chèn next node vào danh sách SE.

Bước 4: Sau khi kết thúc vòng lặp, ta có thể truy vết đường đi ngắn nhất từ nút đích (destination node) đến nút nguồn (source node) bằng cách sử dụng các con trỏ đường dẫn đã lưu trữ.

**Đánh giá**

**Độ phức tạp:**

[Độ phức tạp thời gian là O(N2M), trong đó N là số đỉnh, M là số bước thời gian](https://vnoi.info/wiki/algo/graph-theory/shortest-path.md" \t "https://www.bing.com/_blank). Đây là một độ phức tạp cao, đặc biệt khi đồ thị có nhiều đỉnh và số bước thời gian lớn.

Độ phức tạp không gian của thuật toán là O(NM), trong đó N là số đỉnh, M là số bước thời gian

**Ưu điểm:**

* [Thuật toán có thể xử lý được đồ thị có trọng số âm](https://vnoi.info/wiki/algo/graph-theory/shortest-path.md" \t "https://www.bing.com/_blank)
* [Thuật toán tìm đường đi ngắn nhất phụ thuộc vào thời gian cũng có thể tìm được đường đi ngắn nhất trong đồ thị có hướng và có chu kỳ](https://vnoi.info/wiki/algo/graph-theory/shortest-path.md" \t "https://www.bing.com/_blank)

**Nhược điểm**

* [Thuật toán có thể không tìm được đường đi ngắn nhất nếu đồ thị có chu trình âm](https://vnoi.info/wiki/algo/graph-theory/shortest-path.md).
* [Thuật toán cần phải biết trước các thông tin về đồ thị, như số đỉnh, số cạnh, trọng số của các cạnh, số bước thời gian](https://ichi.pro/vi/diem-manh-va-diem-yeu-cua-thuat-toan-toi-uu-hoa-duoc-su-dung-cho-hoc-may-97385885559261" \t "https://www.bing.com/_blank). Điều này có thể không phù hợp với một số bài toán thực tế, khi các thông tin này có thể thay đổi theo thời gian hoặc không được biết trước.

Bằng cách giải (14) và (15) bằng phương pháp nới lỏng Larangian, giá trị tối ưu cho mô hình (13) với 1 tập vecto nhân tử Larangian có thể biểu diễn dưới dạng sau :



Rõ ràng, giá trị mục tiêu tối ưu của mô hình thoải mái (13) là

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

