DYNAMIC PROGRAMMING METHOD USING FORMULATING TECHNIQUE TO SOLVE SOME TYPICAL PROBLEM IN GRAPH THEORY

Nguyen Van Nui[1*], Nguyen Thi Hang[2]
[1]TNU - University of Information and Communication Technology, [2]Xuan Giang High School, Ha Noi

ARTICLE INFO

Received: 18/10/2022
Revised: 26/12/2022
Published: 26/12/2022

KEYWORDS

Optimization
Dynamic programming
Formulating technique
Optimal solution
Graph theory

ABSTRACT

Dynamic programming has been proven to be an effective method to solve class of optimization problems in the recent years. The study of specific techniques of dynamic programming to solve optimization problems is a really necessary issue. In this paper, we present a dynamic programming method using formulating technique to solve some typical problems in graph theory. Detailed steps of formulating technique have been studied and synthesized to effectively solve a class of typical problems in graph theory. The analysis in oder to select suitable data structure and establish the optimal formula to effectively solve the problem is also presented. Besides, the experiments using python programming language has been conducted for visualizing the results of dynamic programing method with three typical problems in graph theory: shortest path finding, minimum spanning tree finding, maximum network flow finding. The obtained results show that the dynamic programming method using the formulating technique helps to solve some typical problems of graph theory effectively.

PHƯƠNG PHÁP QUY HOẠCH ĐỘNG SỬ DỤNG KỸ THUẬT LẬP HỆ THỨC GIẢI MỘT SỐ BÀI TOÁN TIÊU BIỂU TRONG LÝ THUYẾT ĐỒ THỊ

Nguyễn Văn Núi[1*], Nguyễn Thị Hằng[2]
[1]Trường Đại học Công nghệ Thông tin và Truyền thông – ĐH Thái Nguyên
[2]Trường Trung học phổ thông Xuân Giang – Hà Nội

THÔNG TIN BÀI BÁO

Ngày nhận bài: 18/10/2022
Ngày hoàn thiện: 26/12/2022
Ngày đăng: 26/12/2022

TỪ KHÓA

Tối ưu hóa
Quy hoạch động
Kỹ thuật lập hệ thức
Nghiệm tối ưu
Lý thuyết đồ thị

TÓM TẮT

Quy hoạch động đã được chứng minh là một phương pháp hiệu quả để giải các lớp bài toán tối ưu trong những năm gần đây. Việc nghiên cứu các kỹ thuật cụ thể của quy hoạch động để giải các bài toán tối ưu là một vấn đề thực sự cần thiết. Trong bài báo này, chúng tôi trình bày phương pháp quy hoạch động sử dụng kỹ thuật lập hệ thức để giải một số bài toán điển hình trong lý thuyết đồ thị. Các bước chi tiết của kỹ thuật lập công thức đã được nghiên cứu và tổng hợp để giải một lớp bài toán điển hình trong lý thuyết đồ thị một cách hiệu quả. Phần phân tích nhằm lựa chọn cấu trúc dữ liệu phù hợp và thiết lập công thức tối ưu để giải bài toán một cách hiệu quả cũng được trình bày. Bên cạnh đó, các thực nghiệm sử dụng ngôn ngữ lập trình python đã được tiến hành để trực quan hóa kết quả phương pháp quy hoạch động với 3 bài toán điển hình trong lý thuyết đồ thị: tìm đường đi ngắn nhất, tìm cây khung nhỏ nhất, tìm luồng cực đại. Kết quả thu được cho thấy phương pháp quy hoạch động sử dụng kỹ thuật lập công thức giúp giải hiệu quả một số bài toán điển hình của lý thuyết đồ thị.

DOI: https://doi.org/10.34238/tnu-jst.6715

[*] Corresponding author. *Email: nvnui@ictu.edu.vn*

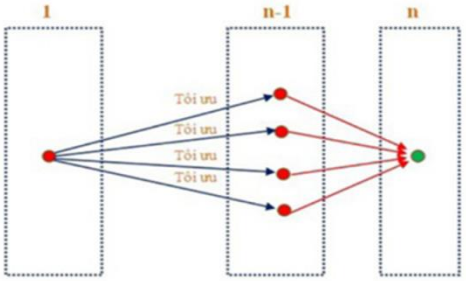
1. Giới thiệu chung

Bài toán tối ưu là bài toán thường có nhiều phương án chấp nhận được và mỗi nghiệm có một giá trị đánh giá. Mục tiêu đặt ra của bài toán là tìm kiếm lời giải tốt nhất, tối ưu nhất.

Phương pháp quy hoạch động dùng để giải bài toán tối ưu có tính chất đệ quy, tức là việc tìm phương án tối ưu cho bài toán đó có thể đưa về tìm phương án tối ưu của một số hữu hạn các bài toán con [1] – [5]. Đối với nhiều thuật toán đệ quy, nguyên lý chia để trị (divide and conquer) thường đóng vai trò chủ đạo trong việc thiết kế thuật toán. Để giải quyết một bài toán lớn, ta chia nó thành nhiều bài toán con cùng dạng với nó để có thể giải quyết độc lập. Trong phương pháp quy hoạch động, nguyên lý này càng được thể hiện rõ: Tại mỗi bước, bài toán cần giải hoặc có lời giải ngay, hoặc sẽ được chia thành nhiều bài toán con nhỏ hơn, vì vậy, việc giải bài toán này chính là giải tập các bài toán con của nó. Tuy nhiên, khi không biết chính xác cần phải giải những bài toán con cụ thể nào, ta sẽ đi giải quyết tất cả các bài toán con và lưu trữ những lời giải hay đáp số của chúng với mục đích sử dụng lại theo một sự phối hợp nào đó để giải quyết những bài toán tổng quát hơn. Đó chính là điểm khác nhau giữa Quy hoạch động và phép đệ quy và cũng là nội dung phương pháp quy hoạch động. Phép đệ quy bắt đầu từ bài toán lớn phân rã thành nhiều bài toán con và đi giải từng bài toán con đó. Việc giải từng bài toán con lại đưa về phép phân rã tiếp thành nhiều bài toán con nhỏ hơn và lại đi giải quyết bài toán nhỏ hơn đó bất kể nó đã được giải hay chưa. Quy hoạch động bắt đầu từ việc giải tất cả các bài toán nhỏ nhất (bài toán cơ sở) để từ đó từng bước giải quyết những bài toán lớn hơn, cho tới khi giải được bài toán lớn nhất (bài toán ban đầu).

Sự xuất hiện của phương pháp quy hoạch động gắn liền với tên tuổi của nhà bác học người Mỹ, R.Bellman [6], [7] mà trong những năm 50 của thế kỷ này đã áp dụng thành công cho lớp các bài toán tối ưu. Chính nhờ sự đơn giản và tường minh của nguyên tắc này mà phương pháp quy hoạch động tỏ ra đặc biệt hấp dẫn và phù hợp cho việc giải các lớp bài toán tối ưu, trong đó có một số bài toán tiêu biểu của lý thuyết đồ thị.

Hình 1 thể hiện ý tưởng của Nguyên lý tối ưu R. Bellman. "Tối ưu bước thứ n bằng cách tối ưu tất cả con đường tiến đến bước n-1 và chọn con đường có tổng chi phí từ bước 1 đến bước n-1 và từ n-1 đến n là thấp nhất (nhiều nhất)".



Hình 1. *Sơ đồ nguyên lý của quy hoạch động*

Nguyên lý R. Bellman cũng có thể được phát biểu theo một cách khác như sau: "Với mỗi quá trình điều khiển tối ưu, bắt đầu từ trạng thái A_0. Khi xây dựng trạng thái A_k bất kỳ trong quá trình đó từ trạng thái A_{k-1}, nếu A_{k-1} là tối ưu thì A_k xây dựng được sẽ tối ưu. Vậy trạng thái A_n cuối cùng sẽ là tối ưu".

Ưu điểm của phương pháp quy hoạch động: Chương trình, thuật toán được giải bằng quy hoạch động chạy nhanh hơn so với phương pháp giải thông thường không sử dụng quy hoạch động [3], [5], [7] – [14]. Tác giả Nguyễn Xuân Huy [5] đã trình bày về một số vấn đề tư duy sáng tạo trong thuật toán và lập trình. Trong bài báo này, tác giả đã chỉ ra rằng phương pháp quy hoạch động và một trong những phương pháp nhanh và hiệu quả nhất để giải quyết các bài toán tối ưu do việc lưu trữ các kết quả đã giải trước đó để làm cơ sở hỗ trợ cho việc xác định lời giải tối ưu ở bước tiếp theo. Nhóm tác giả Nguyễn Xuân My, Hồ Sĩ Đàm, Trần Đỗ Hưng, Lê Sĩ Quang [7]

cũng đã hệ thống một số bài toán chọn lọc trong tin học và phương pháp giải chúng, trong đó nhấn mạnh đến phương pháp quy hoạch động có hiệu quả nhất đối với lớp các bài toán tối ưu và bài toán giải gần đúng. Trong [11], [12], các tác giả đã chứng tỏ sự phù hợp và hiệu quả của phương pháp quy hoạch động trong giải quyết các bài toán tiêu biểu của lý thuyết đồ thị.

Bên cạnh những ưu điểm, phương pháp quy hoạch động vẫn tồn tại một số hạn chế nhất định trong những trường hợp sau đây: (i) Không tìm được công thức truy hồi; (ii) Việc tìm công thức, phương trình truy toán hoặc tìm cách phân rã bài toán nhiều khi rất tốn kèm, dễ bị sai sót. Đồng thời, không phải lúc nào cũng có thể kết hợp việc giải bài toán con đều cho kết quả của bài toán lớn hơn. (iii) Có thể tốn không gian bộ nhớ do cần phải dùng thêm một biến trung gian lưu trữ giá trị, lời giải của các bài toán con.

Từ những phân tích ở trên có thể thấy rằng, phương pháp quy hoạch động là một trong những phương pháp có hiệu quả cao trong việc giải quyết nhiều bài toán thực tế, đặc biệt là lớp các bài toán tối ưu và các bài toán tiêu biểu của lý thuyết đồ thị. Việc nghiên cứu các phương pháp, kỹ thuật để giải quyết những hạn chế ở trên, từ đó hệ thống hóa được một cách có hệ thống việc giải bài toán bằng phương pháp quy hoạch động là một vấn đề thực sự cần thiết. Trong nghiên cứu này, nhóm tác giả sẽ nghiên cứu và trình bày phương pháp quy hoạch động sử dụng kỹ thuật lập hệ thức để giải một số bài toán tiêu biểu của lý thuyết đồ thị. Các kết quả nghiên cứu thực nghiệm cũng được trình bày và mô phỏng cho sự hiệu quả, trực quan của quy hoạch động trong việc giải quyết một số bài toán tiêu biểu của lý thuyết đồ thị.

2. Phương pháp nghiên cứu

2.1. Kỹ thuật lập hệ thức và kỹ thuật tổ chức dữ liệu trong giải bài toán bằng quy hoạch động

2.1.1. Kỹ thuật lập hệ thức

Lập hệ thức là kỹ thuật biểu thị sự tương quan quyết định của bước đang xử lý với các bước đã xử lý trước đó. Hệ thức này là hàm đệ quy hoặc công thức truy hồi.

Dựa vào nguyên lý tối ưu tìm cách chia quá trình giải bài toán thành từng giai đoạn, sau đó tìm hệ thức biểu diễn tương quan quyết định của bước đang xử lý với các bước đã xử lý trước đó. Hoặc tìm cách phân rã bài toán thành các "bài toán con" tương tự có kích thước nhỏ hơn, tìm hệ thức nêu quan hệ giữa kết quả bài toán kích thước đã cho với kết quả của các "bài toán con" cùng kiểu có kích thước nhỏ hơn của nó nhằm xây dựng phương trình truy toán (dạng hàm hoặc thủ tục đệ quy).

2.1.2. Kỹ thuật tổ chức dữ liệu

Phương pháp quy hoạch động lưu giữ kết quả đã tìm kiếm được vào một "*Bảng phương án*" làm giả thiết cho việc tìm kiếm những kết quả của trường hợp sau. "*Bảng phương án*" sẽ được làm đầy giá trị bởi kết quả của những trường hợp trước đã được giải. Như vậy phương pháp quy hoạch động chính là điền vào "*Bảng phương án*" những kết quả lời giải của bài toán con đã giải với mục đích tránh khỏi việc tính toán thừa. Việc sử dụng kỹ thuật tổ chức dữ liệu và cấu trúc phù hợp cho "*Bảng phương án*" này có tính chất quan trọng trong giải quyết bài toán quy hoạch động.

Trong lập trình, "*Bảng phương án"* này thường được biểu diễn bằng mảng hai chiều để lưu trữ giá trị của các phương án trước đó. Để thu hẹp số chiều quy hoạch động, một trong những kỹ thuật phổ biến được sử dụng đó là kỹ thuật chồng mảng. Kỹ thuật này giúp đưa thuật toán quy hoạch động hai chiều về quy hoạch động một chiều. Bảng phương án (hay hàm quy hoạch động, hàm mục tiêu) được tính theo cột (hoặc hàng) mà cột sau (hàng sau) xác định chỉ thông qua một cột liền trước (hàng trước).

2.2. Ứng dụng quy hoạch động giải một số bài toán tiêu biểu trong lý thuyết đồ thị

2.2.1. Bài toán Tìm các đường ngắn nhất

Bài toán: Cho một đồ thị có hướng gồm n đỉnh, mã số từ 1..n với các cung (u, v) có hướng đi từ đỉnh u đến đỉnh v và có chiều dài thể hiện đường đi nối từ đỉnh u đến đỉnh v. Tìm mọi đường đi ngắn nhất từ một đỉnh s cho trước tới các đỉnh còn lại của đồ thị.

Phương pháp Quy hoạch động

Phương pháp quy hoạch động được trình bày trên cơ sở thuật toán Dijkstra, bản chất của thuật toán là sửa đỉnh, chính xác là sửa trọng số của mỗi đỉnh.

Đối tượng mà ta quan tâm ở đây là với đỉnh i nào đó ($i \neq s$) thì đường đi ngắn nhất từ đỉnh s đến đỉnh i là bao nhiêu.

Theo sơ đồ giải các bài toán quy hoạch động trước hết ta xây dựng hệ thức cho bài toán.

Gọi $p(i)$ là độ dài đường ngắn nhất từ đỉnh s đến đỉnh i, $1 \leq i \leq n$. Ta thấy, hàm $p(i)$ phải thoả mãn các tính chất sau:

a) $p(s) = 0$: đường ngắn nhất từ đỉnh xuất phát s đến chính đỉnh đó có chiều dài 0.

b) Với $i \neq s$, muốn đến được đỉnh i ta phải đến được một trong các đỉnh sát trước đỉnh i. Nếu j là một đỉnh sát trước đỉnh i, theo điều kiện của đầu bài ta phải có: a[j,i] > 0, trong đó $a[j, i]$ chính là chiều dài cung ($j \rightarrow i$).

Trong số các đỉnh j sát trước đỉnh i ta cần chọn đỉnh nào?

Kí hiệu *path(x, y)* là đường đi ngắn nhất qua các đỉnh, xuất phát từ đỉnh từ x và kết thúc tại đỉnh $y \neq x$. Khi đó đường từ s đến i sẽ được chia làm hai đoạn, đường từ s đến j và cung ($j \rightarrow i$):

$$path(s,i) = path(s,j) + path(j,i) \quad (1)$$

trong đó path(j, i) chỉ gồm một cung: $path(j,i) = (j \rightarrow i)$

Do $p(i)$ và $p(j)$ phải là ngắn nhất, tức là phải đạt các trị min, ta suy ra điều kiện để chọn đỉnh j sát trước đỉnh i là tổng chiều dài đường từ s đến j và chiều dài cung ($j \rightarrow i$) là ngắn nhất. Ta thu được hệ thức sau:

$$p(i) = \{p(j) + a[j,i] | \; a[j,i] > 0; j = 1..n\} \quad (2)$$

Để ý rằng điều kiện $a[j, i] > 0$ cho biết j là đỉnh sát trước đỉnh i.

Điều tài tình là Dijkstra đã cung cấp thuật toán tính đồng thời mọi đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị.

Thuật toán thực hiện n lần lặp, mỗi lần lặp ta chọn và xử lí một đỉnh của đồ thị. Tại lần lặp thứ k ta khảo sát phần của đồ thị gồm k đỉnh với các cung liên quan đến k đỉnh được chọn trong phần đồ thị đó. Ta gọi phần này là đồ thị con thu được tại bước xử lý thứ k của đồ thị ban đầu và kí hiệu là G(k). Với đồ thị này ta hoàn tất bài giải tìm mọi đường đi ngắn nhất từ đỉnh xuất phát s đến mọi đỉnh còn lại của G(k). Chiều dài thu được ta gán cho mỗi đỉnh i như một trọng số $p[i]$. Ngoài ra, để chuẩn bị cho bước tiếp theo ta đánh giá lại trọng số cho mọi đỉnh kề sau của các đỉnh trong G(k).

Khởi trị: Gán trọng số $p[i] = \infty$ cho mọi đỉnh, trừ đỉnh xuất phát s, gán trị $p[s] = 0$.

Ý nghĩa của thao tác này là khi mới đứng ở đỉnh xuất phát s của đồ thị con G(0), ta coi như chưa thăm mảnh nào của đồ thị nên ta chưa có thông tin về đường đi từ s đến các đỉnh còn lại của đồ thị ban đầu. Nói cách khác ta coi như chưa có đường đi từ s đến các đỉnh khác s và do đó, độ dài đường đi từ s đến các đỉnh đó là ∞.

Giá trị ∞ được chọn trong chương trình là: init = 1e7

Tại bước lặp thứ k ta thực hiện các thao tác sau:

+ Trong số các đỉnh chưa xử lí, tìm đỉnh i có trọng số min.

+ Với mỗi đỉnh j chưa xử lí và kề sau với đỉnh i, ta chỉnh lại trọng số p[j] của đỉnh đó theo tiêu chuẩn sau:

Nếu $p[i] + a[i, j] < p[j]$ thì gán cho $p[j]$ giá trị mới: $p[j]=p[i]+a[i,j]$

Ý nghĩa của thao tác này là: nếu độ dài đường đi path(s, j) trong đồ thị con G(k - 1) không qua đỉnh i mà lớn hơn độ dài đường đi mới path(s, j) có qua đỉnh i thì cập nhật lại theo đường mới đó.

+ Sau khi cập nhật ta cần lưu lại vết cập nhật đó bằng lệnh gán before[i] = j với ý nghĩa là, đường ngắn nhất từ đỉnh s tới đỉnh j cần đi qua đỉnh i.

+ Đánh dấu đỉnh i là đã xử lí.

Mảng p[i] lưu giữ kết quả của bài toán, độ dài đường đi ngắn nhất từ đỉnh s, tới đỉnh i (i =1,n)

Độ phức tạp: O(n^2) [7]

Như vậy, ta có sơ đồ giải quyết bài toán tìm đường đi ngắn nhất trên đồ thị áp dụng phương pháp quy hoạch động sử dụng kỹ thuật lập hệ thức như sau.

Bước 1. Lập hệ thức quy hoạch động

$$p(i) = \{p(j) + a[j,i] | \, a[j,i] > 0; j = 1..n\} \quad (3)$$

Bước 2. Tổ chức dữ liệu: Sử dụng mảng một chiều p[i] để lưu kết quả của bài toán

2.2. Bài toán Tìm cây khung nhỏ nhất

Bài toán: Cho G=(V, E) là đồ thị vô hướng, liên thông. Với mỗi cạnh e∈ E người ta cho tương ứng với một số w(e) gọi là trọng số.

Yêu cầu: Tìm cây khung có trọng số nhỏ nhất trong số các cây khung của đồ thị G.

2.2.1. Thuật toán Prim

- Bước 1: Chọn cạnh bất kỳ có trọng số nhỏ nhất, đặt nó vào cây.
- Bước 2: Ghép vào cây cạnh có trọng số nhỏ nhất liên thuộc với một đỉnh của cây và không tạo ra chu trình trong cây.
- Bước 3: Nếu (*n*-1) cạnh được ghép vào cây thì thuật toán dừng, nếu chưa thì quay lại tiếp tục bước 2.

2.2.2. Thuật toán Kruskal

- Bước 1: Chọn cạnh bất kỳ có trọng số nhỏ nhất ghép vào cây.
- Bước 2: Ghép cạnh có trọng số tối thiểu vào cây và không tạo thành chu trình.
- Bước 3: Nếu (n-1) cạnh được ghép vào cây thì thuật toán dừng, nếu chưa thì quay lại tiếp tục bước 2.

2.2.3. Phương pháp Quy hoạch

Bước 1: Lập hệ thức.

Gọi T(k) (k=1, 2, ...,n), đồ thị nhận được trong vòng lặp thứ k, là một đồ thị con của cây khung nhỏ nhất của G, do đó T(n) chính là một cây khung nhỏ nhất của G.

T(1) chỉ gồm đỉnh v_* của G, do đó T(1) là đồ thị con của mọi cây khung của G. Giả sử T(i) (1≤i<n) là một đồ thị con của một cây khung nhỏ nhất của G. $E_{T(1)=} \emptyset$

Theo thuật toán Prim và Karuskal ta có cùng một hệ thức truy hồi:

$$E_{T(i+1)}=E_{T(i)} \cup \{e_{i+1}\}, \; i=\overline{1,n-1} \quad (4)$$

(Với e_{i+1} là cạnh ngắn nhất trong tất cả các cạnh có một đầu mút thuộc $V_{T(i)}$, đầu mút kia không thuộc $V_{T(i)}$. Nói cách khác e_{i+1}= (u,v) thì u∈$V_{T(i)}$ và v∉$V_{T(i)}$ T - Theo thuật toán Karuskal, e_{i+1} là cạnh ngắn nhất trong tất cả các cạnh –Theo thuật toán Prim).

Khi đó: $T(i+1) = T(i) \cup \{u,v\}$

Bước 2: Tổ chức dữ liệu: Ta sử dụng mảng T[i] để lưu kết quả bài toán.

2.3. Bài toán Tìm luồng cực đại trong mạng

2.3.1. Một số định nghĩa

Định nghĩa 1. Ta gọi mạng là đồ thị có hướng G=(V,E), trong đó có duy nhất một đỉnh s không có cung đi vào gọi là điểm phát, duy nhất một đỉnh t không có cung đi ra, gọi là điểm thu và mỗi cung e=(v,w) ∈ E được gán với một số không âm c(e)= c(v,w) gọi là khả năng thông qua của cung e.

Định nghĩa 2. Giả sử cho mạng G=(V,E), ta gọi luồng f trong mạng G=(V,E) là ánh xạ f: E→ R_+ gán cho mỗi cung e=(v,w) ∈ E một số thực không âm f(e)=f(v,w), gọi là luồng trên cung e thỏa mãn các điều kiện sau:

1) Luồng trên mỗi cung e không vượt quá khả năng thông qua của nó.

$$0 \le f(e) \le c(e)$$

2) Điều kiện cân bằng luồng trên mỗi đỉnh của mạng: Tổng luồng trên các cung đi vào đỉnh v, bằng tổng luồng trên các cung đi ra khỏi đỉnh v nếu v ≠ s,t:

$$Div_f(v) = \sum_{w \in \Gamma^-(v)} f(w,v) - \sum_{w \in \Gamma^+(v)} f(v,w) \quad (5)$$

Trong đó :

$\Gamma^-(v)$ - tập đỉnh của mạng mà từ đó có cung đến v,

$\Gamma^+(v)$- tập đỉnh của mạng mà từ v có cung đến nó.

$$\Gamma^-(v) = \{ w \in V: (w,v) \in E\}$$
$$\Gamma^+(v) = \{ w \in V: (v,w) \in E\}$$

Định nghĩa 3. Giá trị luồng: Ta gọi giá trị luồng f là số $Val(f)$ được tính như sau:

$$Val(f) = \sum_{w \in \Gamma^-(s)} f(s,w) = \sum_{w \in \Gamma^+(t)} f(w,t) \quad (6)$$

2.3.2. Bài toán: Cho mạng G=(V,E). Hãy tìm luồng f* trong mạng với giá trị luồng Val(f*) là lớn nhất.

2.3.3. Phương pháp quy hoạch động

Bước 1: Lập hệ thức.

Gọi f(u,v) là một luồng trong mạng.

Xuất phát: f(u,v)=0 với $\forall u, v \in V$

Giả sử P=(s=v_0, v_1, v_2,…v_k=t) là một đường đi từ s đến t trong đồ thị tăng luồng G_f.

Gọi δ là giá trị nhỏ nhất của các trọng số của các cung trên đường đi P. Khi đó xây dựng luồng f' trên mạng G theo quy tắc sau:

$$f'(u,v) = \{f(u,v) + \delta, (u,v) \in P \text{ là } cung\ thuận\ f(u,v) - \delta, (u,v) \in P \text{ là } cung\ nghịch\ f(u,v)$$

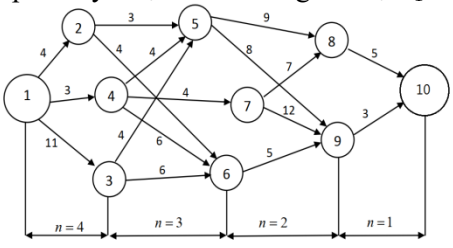
Bước 2: Tổ chức dữ liệu:

Để lưu kết quả bài toán ta dùng biến Val (max_flow), các phép xử lý thực chất là thay đổi trọng số của các cung trên mỗi bước giống thuật toán Dijkstra.

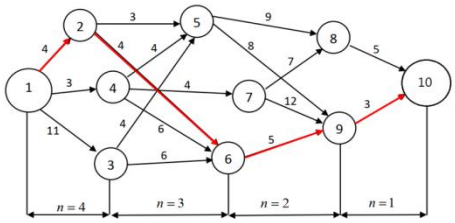
3. Một số kết quả thực nghiệm

3.1. Bài toán Tìm các đường ngắn nhất

Một công ty cần chở hàng từ thành phố ❶ đến thành phố ❿. Mạng các con đường nối 2 thành phố này được mô tả bằng đồ thị G_1 như Hình 2.



Hình 2. *Đồ thị G_1 mô tả mạng các thành phố*

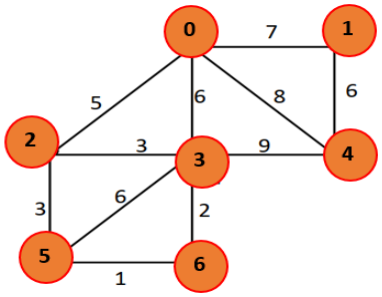


Hình 3. *Minh họa đường đi ngắn nhất từ đỉnh (1) đến đỉnh (10)*

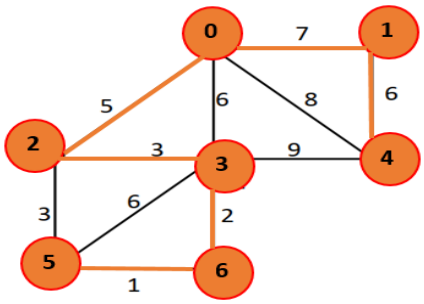
Hình 2 mô tả kết quả tìm đường đi ngắn nhất trên đồ thị G_1 bằng phương pháp quy hoạch động. Hình 3 mô tả đường đi ngắn nhất từ phố ❶ đến thành phố ❿ chính là đường đi từ đỉnh ❶ → ❷ →❻→❾→❿, với tổng giá trị là 32 đơn vị độ dài.

3.2. Bài toán Tìm cây khung nhỏ nhất

Bài toán: Tìm cây khung nhỏ nhất của đồ thị có trọng số G_2 như hình bên dưới:



Hình 4. *Đồ thị G_2*

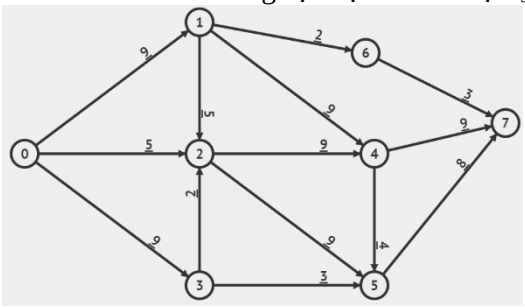


Hình 5. *Minh họa kết quả cây khung nhỏ nhất của đồ thị G_2*

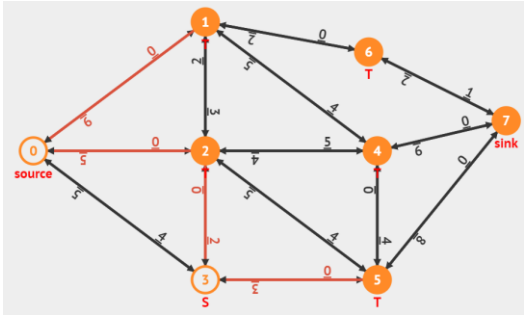
Hình 4 mô tả đồ thị G_2 có 7 đỉnh (đánh số từ 0 đến 6) có các cạnh với trọng số tương ứng. Sau khi giải bằng quy hoạch động để tìm cây khung nhỏ nhất của đồ thị G_2, kết quả nhận được chính là cây khung nhỏ nhất được mô tả ở Hình 5 (Cây khung có đầy đủ các đỉnh với các cạnh được tô màu, trọng số của cây khung này là 24).

3.3. Bài toán Tìm luồng cực đại trong mạng

Bài toán: Tìm luồng cực đại của đồ thị G_3 như hình bên dưới:



Hình 6. *Đồ thị luồng G_3*



Hình 7. *Minh họa kết quả tìm luồng cực đại của đồ thị G_3*

Hình 6 mô tả đồ thị G_3 có 8 đỉnh (đánh số từ 0 đến 7) có các cung với trọng số tương ứng. Sau khi giải bằng quy hoạch động, luồng cực đại của G_3 được mô tả ở Hình 7 (Luồng cực đại có các cạnh đi qua đường cắt ST được tô màu; tổng "giá trị" luồng cực đại là 19).

4. Kết luận

Quy hoạch động là một trong những phương pháp có hiệu quả cao trong việc giải quyết nhiều bài toán thực tế, đặc biệt là lớp các bài toán tối ưu và các bài toán tiêu biểu của lý thuyết đồ thị. Với các bài toán tối ưu và các bài toán giải gần đúng, thuật toán quy hoạch động luôn được ưu tiên lựa chọn để tìm ra lời giải tối ưu nhất. Tuy nhiên, phương pháp quy hoạch động không phải là phương pháp thần thánh có thể giải được tất cả các bài toán. Do vậy, việc nghiên cứu các phương pháp, kỹ thuật cụ thể, từ đó hệ thống hóa được một cách có hệ thống việc giải bài toán bằng phương pháp quy hoạch động là một vấn đề thực sự cần thiết. Trong bài báo này, chúng tôi trình bày phương pháp quy hoạch động sử dụng kỹ thuật lập hệ thức để giải một số bài toán tiêu biểu trong lý thuyết đồ thị. Các bước và kỹ thuật cụ thể để giải quy hoạch động nói chung, giải một số bài toán tiêu biểu của lý thuyết đồ thị nói riêng đã được nghiên cứu trong bài báo này. Kết quả thực nghiệm cho thấy phương pháp quy hoạch động sử dụng kỹ thuật lập hệ thức giúp giải hiệu quả một số bài toán tiêu biểu của lý thuyết đồ thị.

TÀI LIỆU THAM KHẢO/REFERENCES

[1] J. Wengrow, *A Common-Sense Guide to Data Structures and Algorithms, Second Edition: Level Up Your Core Programming Skills*, Pragmatic Bookshelf, 2020.

[2] M. X. Vu, N. H. Nguyen, and V. T. Nguyen, *Fundamental of Algorithms and Programming*. University of Education Publishing House, 2014.

[3] E. A. G. Souza, M. S. Nagano, and G. A Rolim, "Dynamic Programming algorithms and their applications in machine scheduling: A review," *Expert Systems with Applications,* vol. 190, March 2022, Art. no. 116180.

[4] H. Rahmanian and M. K. Warmuth, "Online Dynamic Programming," *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017*), Long Beach, CA, USA, 2017, pp. 2824–2834.

[5] X. H. Nguyen, *Creativity in Algorithms and Programming*, 5th Edition, Information and Communications Publishing House, 2019.

[6] H. D. Nguyen, *Some problems about Algorithms*. Education Publishing House, 2005.

[7] X. M. Nguyen, S. D. Ho, D. H. Tran, and S. Q. Le, *Some selected problems in Informatic*s. Education Publishing House, 2002.

[8] D. G. Do, *Discrete Math applied in Informatics*. Education Publishing House, 2021.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms.* The MIT Press, Cambridge, London, England, 2022.

[10] J. Rue, I. Sau, and D. M. Thilikos, "Dynamic Programming for Graphs on Surfaces," *Proceedings of ICALP'2010*, France, 2011, pp. 01-32.

[11] N. Hoch, U. Montanari, and M. Sammartino, "Dynamic Programming on Nominal Graphs," in *Graphs as Models 2015* (*GaM'15*) - *Electronic Proceedings in Theoretical Computer Science EPTCS*, Open Publishing Association, 2015, pp. 80–96.

[12] P. Recht, "A Dynamic Programming Approach for the Max-Min Cycle Packing Problem in Even Graphs," *Open Journal of Discrete Mathematics*, vol. 6, pp. 340-350, 2016.

[13] R. Rodriguez-Puente and M. S. Lazo-Cortes, *Algorithms for shortest path search in Geographic information systems by using reduced graphs*. Springer, 2013.

[14] K. R. Saoub, *Graph Theory: An introduction to proofs, algorithms, and applications*. CRC Press, Taylor & Francis Group, A Chapman & Hall Book, 2021.