TẠP CHÍ KHOA HỌC, Đại học Huế, Tập 74B, Số 5, (2012), 81-92

SONG SONG HÓA THUẬT TOÁN DIJKSTRA TÌM ĐƯỜNG ĐI NGẮN NHẤT TỪ MỘT ĐỈNH ĐẾN TẤT CẢ CÁC ĐỈNH

Nguyễn Đình Lầu, Trần Ngọc Việt

Trường Cao đẳng Giao thông Vận tải II

Tóm tắt. Nội dung chính của bài báo tập trung xây dựng thuật toán song song tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh của đồ thị liên thông dựa trên thuật toán tuần tự Dijkstra. Ý tưởng của thuật toán là sử dụng m bộ xử lý tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh trên đồ thị. Trong m bộ xử lý chọn một bộ xử lý đóng vai trò trung tâm thực hiện việc quản lý dữ liệu, chia n đỉnh và ma trận trọng số của đồ thị cho m bộ xử lý để tìm đường đi ngắn nhất.

1. Giới thiệu

Bài toán tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh là một trong số những bài toán tối ưu trên đồ thị và được ứng dụng rộng rãi trong thực tế cũng như các ứng dụng thú vị trong ngành toán học rời rạc. Bài toán được đề xuất và giải quyết bởi nhà khoa học máy tính người Hà Lan Edsger Dijkstra và được gọi là thuật toán Dijkstra. Thuật toán có độ phức tạp là $O(n^2)$, với độ phức tạp tính toán cao của thuật toán này cũng như đòi hỏi về mặt thời gian, việc giải bài toán này với tính chất tuần tự của giải thuật sẽ gặp phải những vấn đề về thời gian thực hiện chương trình, tốc độ xử lý, khả năng lưu trữ của bộ nhớ, xử lý dữ liệu với quy mô lớn,… kích thước của bài toán tăng lên và không gian tìm kiếm càng lớn, yêu cầu phải song song hóa giải thuật để tăng tốc độ và hiệu quả của giải thuật [1], [2].

Thuật toán đã giải quyết trên đồ thị với thời gian chạy khá lâu trên đồ thị có số đỉnh lớn và dễ dàng tìm thấy nhiều ứng dụng trong các lĩnh vực khoa học kỹ thuật, y tế, sinh vật và đặc biệt trong mạng giao thông vận tải. Tuy nhiên, có rất nhiều ứng dụng cần xử lý nhanh trên đồ thị có số đỉnh lớn thì thuật toán tuần tự không đáp ứng được. Vì vậy ta phải tìm cách giải quyết bài toán với số đỉnh lên đến hàng chục ngàn đỉnh mà thời gian chạy phải được rút gọn. Điều này đặt ra là ta phải chia đồ thị cho nhiều bộ xử lý đồng thời tham gia tính toán, dẫn đến ta phải xây dựng thuật toán song song trên đa bộ xử lý, điều này thuật toán tuần tự chạy trên một bộ xử lý không thể thực hiện được. Hiện nay thuật toán song song được phát triển nhiều ở Việt Nam cũng như trên thế giới và đặc biệt ở các trường đại học ở Mỹ, Nhật… Tại các trường đại học này đều có hệ thống máy tính song song để chạy thử nghiệm các thuật toán song song. Để xâm nhập

vào các hệ thống này đòi hỏi ta phải có tài khoản và các phần mềm tương ứng. Trong khi đang nghiên cứu các hệ thống máy tính song song, chúng tôi được cấp tài khoản để kết nối đến hệ thống cụm máy tính của trường đại học Sư phạm Hà Nội với trên toàn cầu là *ccs1.hnue.edu.vn* hệ thống này cho phép chúng tôi thử nghiệm thuật toán song song trong bài báo này rất tốt.

Hiện nay, mô hình xử lý song song đã và đang phát triển mạnh mẽ giải quyết những vấn đề bế tắc mà mô hình xử lý tuần tự gặp phải như: vấn đề thời gian thực hiện chương trình, tốc độ xử lý, khả năng lưu trữ của bộ nhớ và xử lý dữ liệu với quy mô lớn.

Trong bối cảnh đó chúng tôi xây dựng thuật toán *"Song song hóa thuật toán Dijkstra tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh"* trên đồ thị với m bộ xử lý nhằm khắc phục được các vấn đề tồn tại đã nêu ở trên.

2. Thuật toán tuần tự Dijkstra tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh

Đầu vào: Đồ thị liên thông G(V,E,w), $w(i,j) > 0 \ \forall (i,j) \in E$, đỉnh nguồn a.

Đầu ra: Chiều dài đường đi ngắn nhất và đường đi ngắn nhất từ đỉnh a đến tất cả các đỉnh trên đồ thị.

+ Phương pháp:

Bước 1. Gán L(a):=0. Với mọi đỉnh $x \neq a$ gán $L(x) = \infty$. Đặt T:=V.

Bước 2. Chọn $v \in T$, v chưa xét sao cho L(v) có giá trị nhỏ nhất. Đặt T:=T\{v}, đánh dấu đỉnh v đã xét.

Bước 3. Nếu $T=\phi$, kết thúc. L(z), $\forall z \in V$, $z \neq a$ là chiều dài đường đi ngắn nhất từ a đến z. Từ z lần ngược theo đỉnh được ghi nhớ ta có đường đi ngắn nhất. (L(z) không thay đổi, nếu $L(z)= \infty$ thì không tồn tại đường đi_(Not Path)).

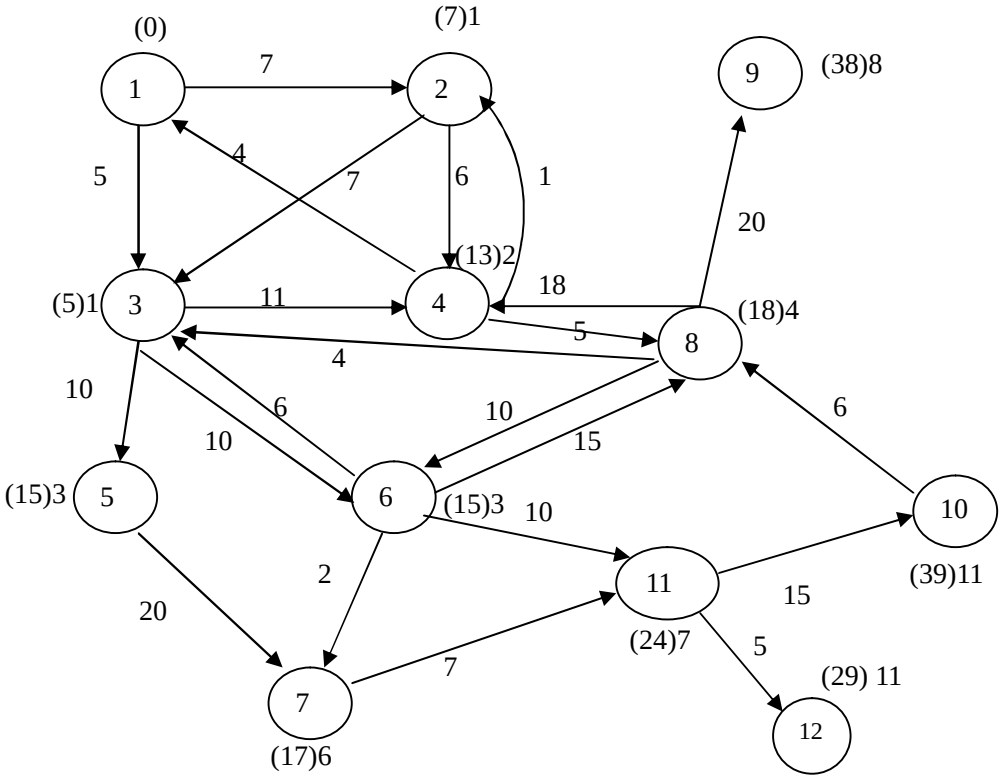
Ngược lại sang bước 4.

Bước 4. Với mỗi $x \in T$ kề v gán $L(x):= \min\{L(x), L(v)+w(v,x)\}$. Nếu L(x) thay đổi thì ghi nhớ đỉnh v cạnh đỉnh x bằng mảng truoc[] (với truoc[] của đỉnh 1= 0) để sau này xây dựng đường đi ngắn nhất.

Quay về bước 2.

Độ phức tạp của thuật toán Dijkstra là $O(n^2)$ [3].

Ví dụ: Cho đồ thị được biểu diễn như sau, sau khi thuật toán thực hiện xong thì kết quả được ghi nhớ lên các nhãn đỉnh tương ứng.



Hình 1. *Ghi nhớ kết quả tính được trên đồ thị.*

Mảng truoc[]=0 1 1 2 3 3 6 4 8 11 7 11 (Mảng ghi nhớ truoc [] dùng để tìm đường đi, với truoc[1]=0)

Mảng độ dài L = 0 7 5 13 15 15 17 18 38 39 24 29

Vậy kết quả từ đỉnh 1 đến tất cả các đỉnh là:

đến 2=7 (1→2)

đến 3=5 (1→3)

đến 4=13 (1→2→4)

đến 5=15 (1→3→5)

đến 6=15 (1→3→6)

đến 7=17 (1→3→6→7)

đến 8=18 (1→2→4→8)

đến 9=38 (1→2→4→8)

đến 10=39 (1→3→6→7→11→10)

đến 11= 24 (1→3→6→7→11)

đến 12=29 (1→3→6→7→11→12)

Với thuật toán tuần tự như trên, giải thuật có độ phức tạp là $O(n^2)$ khi n tăng lên quá lớn (khoảng vài chục ngàn đỉnh) thì thời gian xử lý sẽ chậm đi đánh kể, điều này không đáp ứng được những ứng dụng cho giải thuật Dijkstra đòi hỏi chạy với thời gian nhanh hơn. Hơn nữa hiện nay mô hình xử lý song song cũng như các hệ thống xử lý song song được phát triển mạnh mẽ trên thế giới cũng như ở Việt Nam. Xử lý song song tốn ít thời gian hơn và thời gian là khác nhau tùy theo hệ thống có bao nhiêu bộ xử lý (BXL) và chỉ ra được trong tính toán song song thì thời gian thực hiện của bài toán phụ thuộc vào thời gian truyền dữ liệu trong hệ thống cộng với thời gian thực hiện tính toán trong các BXL [8]. Vì vậy chúng tôi xây dựng giải thuật này theo hướng song song hóa.

3. Thuật toán song song Dijkstra tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh

Ý tưởng: Chia đồ thị ban đầu cho m bộ xử lý (P_0, P_1,…,P_{m-1}), cùng tính toán đồng thời, mỗi bộ BXL đảm nhận n/m đỉnh của đồ thị (n là số đỉnh trên đồ thị, m là số BXL) và ma trận trọng số n/m cột n dòng

(nếu n chia hết cho m). Ngược lại trường hợp không chia hết thì ta sẽ thực hiện theo công thức (*) ở bước 1 trong thuật toán song song. với m BXL, mỗi bộ xử lý sẽ thực hiện tính min L(x) với x là những đỉnh kề với đỉnh mà nó đang nhận để xét. Sau đó BXL trung tâm (P_0) sẽ tìm min của các L(x) trên các BXL để tiếp tục gửi đỉnh x lên các BXL để thực hiện. [7]

Đầu vào: Đồ thị liên thông G(V,E,w), $w(i,j) > 0 \ \forall (i,j) \in E$, đỉnh nguồn a, 1 bộ xử lý chính và m-1 bộ xử lý phụ.

Đầu ra: Chiều dài đường đi ngắn nhất là đường đi ngắn nhất từ đỉnh a đến tất cả các đỉnh trên đồ thị.

+ Phương pháp:

Bước 1. Bộ xử lý chính thực hiện

- Gán L(a):=0. Với mọi đỉnh $x \neq a$, x thuộc bộ xử lý chính gán $L(x)= \infty$.

- Chia đều số đỉnh và ma trận trọng số để gửi cho m BXL.

Cách chia đều như sau: Giả sử ta có n đỉnh và m bộ xử lý P_0,P_2,…,P_{m-1}.

Gọi n_i là số đỉnh của bộ xử lý P_i (i=0,…,m-1).

- Nếu n chia hết cho m thì

For i=0 to m-1 do

$n_i = n/m$

- Nếu n không chia hết cho m thì

For i=0 to m-2 do

$$n_i = \left\lfloor \frac{n}{m} \right\rfloor$$

$$n_{m-1} = n - (m-1).\left\lfloor \frac{n}{m} \right\rfloor (*)$$

- Ta xây dựng T_i (i=0,…,m-1) là tập đỉnh mà bộ xử lý P_i sẽ nhận như sau:

BN=0; kt là số phần tử mà các bộ xử lý nhận

for (k=0; k<m; k++)

{

$T_k=\phi$; $BN = k.\left\lfloor \frac{n}{m} \right\rfloor$

for (j=0; j<kt; j++)

$T_k=T_k+\{j+ BN+1\}$

}

- Ta xây dựng A_i (i=0,…,m-1) là ma trận trọng số mà bộ xử lý P_i sẽ nhận như sau:

Gọi A là ma trận trọng số của đồ thị đầu vào thì $A=(w_{ij})_{nxn}$.

Suy ra: $A_i = \left(w_{kj}\right)_{\overline{k=1,n};\, j\in T_i}$ sẽ được bộ xử lý P_i (i=0,…,m-1) nhận.

- Gửi T_i (i=1,..,m-1), A_i (i=1,..,m-1), L(x), $x \in T_i$ (i=1,..,m-1), cho m-1 BXL phụ $P_1,P_2,\ldots,P_{m-1}$.

Bước 2. m bộ xử lý thực hiện

- Trong m bộ xử lý (trừ bộ xử lý chính), gán $L(x_i)= \infty$, sao cho x_i thuộc T_i (i=1,…,m-1).

- Trên m bộ xử lý tìm $L_i(x_i)= \min\{L(x_i), x_i \in T_i$ (i=0,…,m-1), x_i chưa xét$\}$.

- Các bộ xử lý phụ gửi $L_i(x_i)$ về bộ xử lý chính.

Bước 3. Bộ xử lý chính thực hiện

- Bộ xử lý chính tìm *l(v)*= $\min\{L_i(x_i)$, i=0,..,m-1$\}$trên m bộ xử lý.

- Chuyển đỉnh v lên m bộ xử lý để thực hiện các bước tiếp theo.

Bước 4. m bộ xử lý thực hiện

- Trên m BXL kiểm tra nếu $v \in T_i$, $T_i=T_i\backslash\{v\}$ (i=0,…,m-1), đánh đấu đỉnh v đã xét.

- Kiểm tra nếu T_i (i=0,..,m-1) = ϕ, thì bộ xử lý thứ i kết thúc, sang bước 6.

- Ngược lại, sang bước 5.

Bước 5. m bộ xử lý thực hiện

for x ∈ T_i (i=0,…,m-1) kề với v

if *L(x)>L(v)+w(v,x)*

{ L(x) := L[v] + w(v,x)

Truoc[x]=v // ghi nhớ đỉnh v vào x }

quay lại bước 2.

Bước 6. Bộ xử lý chính thực hiện

Nếu tất cả m-1 bộ xử lý phụ kết thúc thì bộ xử lý chính thực hiện: nhận kết quả từ các bộ xử lý phụ và kết luận chiều dài đường đi ngắn nhất từ a đến tất cả các đỉnh và đường đi ngắn nhất qua các đỉnh đã ghi nhớ. Đỉnh nào có nhãn không thay đổi (bằng ∞) thì không tồn tại đường đi_(Not Path). Hệ thống kết thúc.

Chúng tôi quy ước tỷ lệ giữa thời gian tính toán tuần tự (T_s) với thời gian tính toán song song (T_p) để đánh giá thời gian thực hiện của thuật toán song song so với thuật toán tuần tự [4], [5] .

Chúng tôi tạo ngẫu nhiên đồ thị gồm 1000 đỉnh và cho thực hiện trên 1 bộ xử lý (tuần tự), 2, 4, 6, 8 bộ xử lý trên hệ thống của trường đại học Sư phạm Hà Nội thì kết quả được cho ở bảng sau:

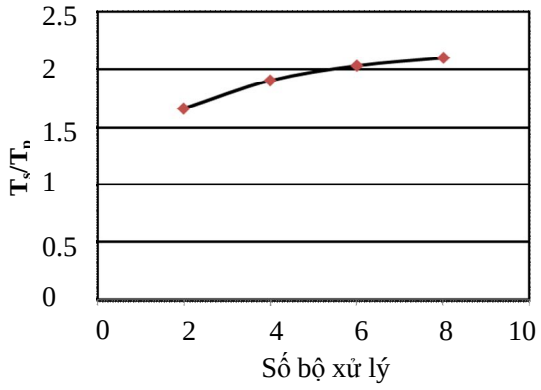
Số bộ xử lý	1	2	4	6	8
T_s/T_p	1	1.66	1.90	2.03	2.10

Bảng đánh giá thời gian thực hiện thuật toán song song so với thuật toán tuần tự, trong đó:

Ts: Sequential time (Thời gian chạy tuần tự)

Tp: Parallel time (thời gian chạy song song)

Kết quả đánh giá được biểu diễn bằng biểu đồ sau:



Hình 2. *Biểu diễn sự đánh giá thuật toán song song so với tuần tự*

Nhận xét: Từ đồ thị và bảng kết quả trên, nhận thấy rằng tốc độ xử lý được tăng lên đáng kể khi xử lý trên 2 BXL thì thời gian giảm hơn 1.6 lần so với xử lý tuần tự.

4. Thực nghiệm thuật toán song song

Tìm đường đi ngắn nhất từ đỉnh nguồn a=1 đến tất cả các đỉnh theo thuật toán song song trên đồ thị (n=12 đỉnh) dưới đây cho m=2 bộ xử lý (P_0, P_1), trong đó: P_0 là bộ xử lý chính và P_1 là bộ xử lý phụ.

Bước 1. Bộ xử lý chính P_0 thực hiện.

Phân công T_0={1,2,3,4,5,6}, A_0 cho chính P_0 và phân công T_1={7,8,9,10,11,12}, A_1 cho P_1

Gán L(1)=0; L(2)=L(3)=L(4)=L(5)=L(6)

Bộ xử lý chính P_0 nhận đỉnh a=1, L(1)=0, L(2)=L(3)=L(4)=L(5)=L(6)=∞, T_0={1,2,3,4,5,6} và A_0 .

Bộ xử lý chính P_0 thực hiện gửi L(7)=L(8)=L(9)=L(10)=L(11)=∞, T_1={7,8,9,10,11,12} và A_1 đến bộ xử lý P_1.

<u>Bộ xử lý P_0 thực hiện</u>

Bước 2.

Tìm *minl*= min{L(x), x ∈ T_0} → *minl*=0.

<u>Bộ xử lý P_1 thực hiện</u>

Bước 2.

Gán L(7)=L(8)=L(9)=L(10)=L(11)=L(12)= ∞

Tìm *minl*=min{L(x), x ∈ T_1}= ∞

Gửi *minl* về P_0

<u>Bộ xử lý P_0 thực hiện</u>

Bước 3.Tìm min của *minl* đã chuyển đến ở bước 2 với *minl* mà P_0 giữ

→Min=Lv=L(1)=0, gán đỉnh v = 1

Chuyển v = 1 lên P_0 và P_1

<u>Bộ xử lý P_0 thực hiện</u>

Bước 4.

T_0=T_0\{v}={1,2,3,4,5,6}\{1}={2,3,4,5,6}, đánh dấu đỉnh 1 đã xét.

$T_0 \neq \phi$ sang bước 5.

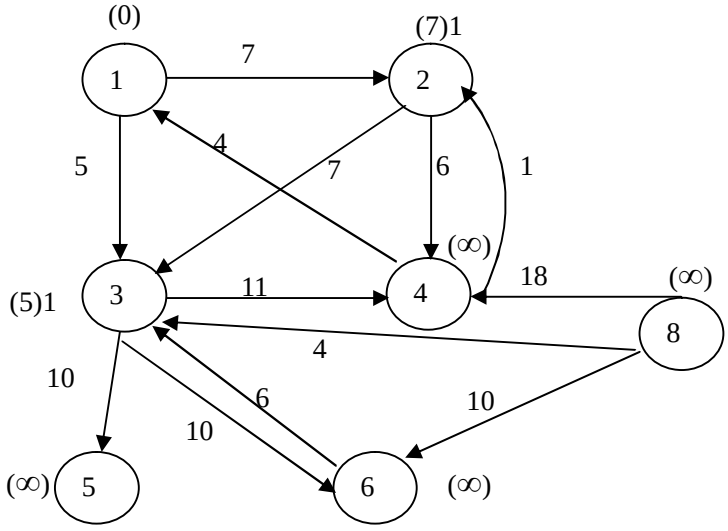
Bộ Xử lý P_0 thực hiện

Bước 5.

Đỉnh 2,3 kề với đỉnh 1.

L(2)=min{L(2), w(1,2)+L(1)}=7 thay đổi. L(3)=min{L(3), w(1,3)+L(1)}= 5 thay đổi.

Ghi nhớ đỉnh 1 vào đỉnh 2,3 đồ thị có các nhãn như sau:



Hình 3. *Đồ thị ghi nhớ trên bộ xử lý chính (P_0)*

Bộ xử lý P_1 thực hiện

Bước 4.

v=1, đỉnh 1 $\notin T_1$, suy ra T_1={7,8,9,10,11,12}.

$T_1 \neq \phi$, sang bước 5.

Bước 5.

không có đỉnh nào kề với đỉnh 1, đồ thị có nhãn không thay đổi

Bộ xử lý P_0 thực hiện

Bước 2.

Bộ xử lý P_0 tìm *minl*=min{L(2), L(3), L(4),L(5),L(6)}= L(3)=5, gửi về bộ xử lý chính.

Bộ xử lý P_1 thực hiện

Bước 2.

Bộ xử lý P_1 tìm *minl*=min{L(7), L(8), L(9),L(10),L(11),L(12)}= ∞, gửi về bộ xử lý chính.

Bộ xử lý P_0 thực hiện

Bước 3.

Tìm min của minl đã chuyển đến ở bước 2 với minl mà P_0 giữ

Min=Lv=L(3)=5, gán đỉnh v=3

Chuyển v=3 lên P_0 và P_1

Bộ xử lý P_0 thực hiện

Bước 4.

$T_0=T_0\backslash\{v\}=\{1,2,3,4,5,6\}\backslash\{3\}=\{2,4,5,6\}$, đánh dấu đỉnh 1 đã xét.

$T_0 \neq \phi$ sang bước 5.

Bộ xử lý P_0 thực hiện

Bước 5.

Đỉnh 4,5,6 kề đỉnh 3

L(4)=min{L(4),w(3,4)+L(3)}=16 thay đổi, L(5)=min{L(5),w(3,5)+L(3)}=15 thay đổi

L(6)=min{L(6),w(3,6)+L(3)}=15 thay đổi

Ghi nhớ đỉnh 3 vào đỉnh 4,5,6 . Quay lại bước 2.

Bộ xử lý P_1 thực hiện

Bước 4.

v=3, đỉnh 3 $\notin T_1$, suy ra $T_1=\{7,8,9,10,11,12\}$.

$T_1 \neq \phi$, sang bước 5.

Bộ xử lý P_1 thực hiện

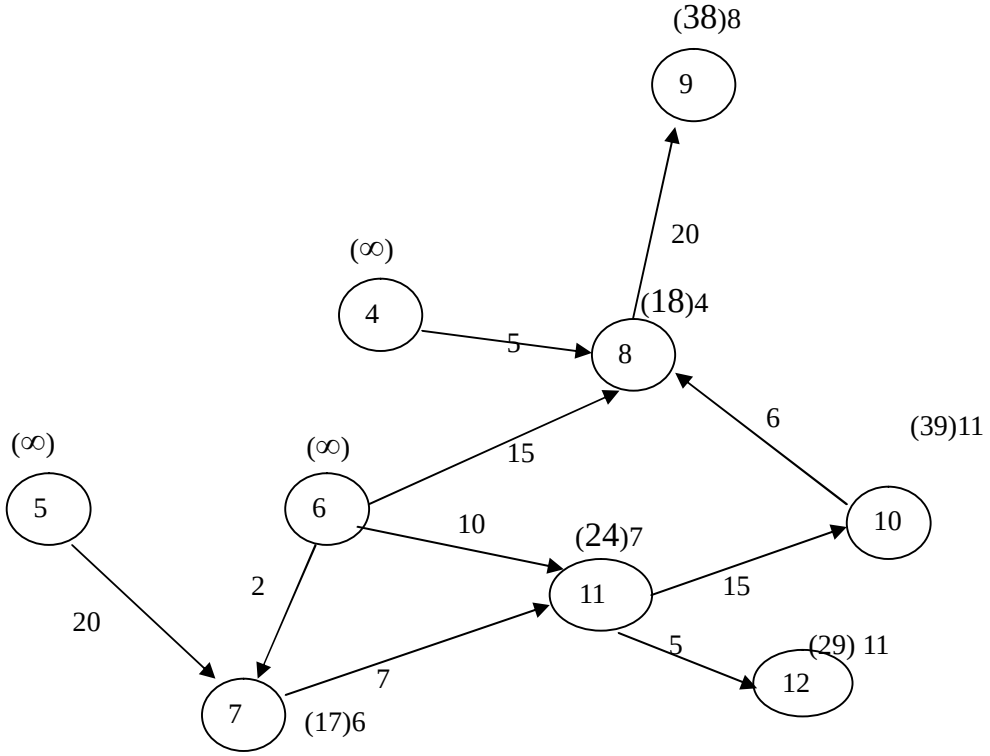
Bước 5.

không có đỉnh nào kề với đỉnh 3, đồ thị có nhãn không thay đổi.

Quay lại bước 2.

Cứ tiếp tục các bước trên cho hai bộ xử lý ta thu được kết quả ở hai bộ xử lý

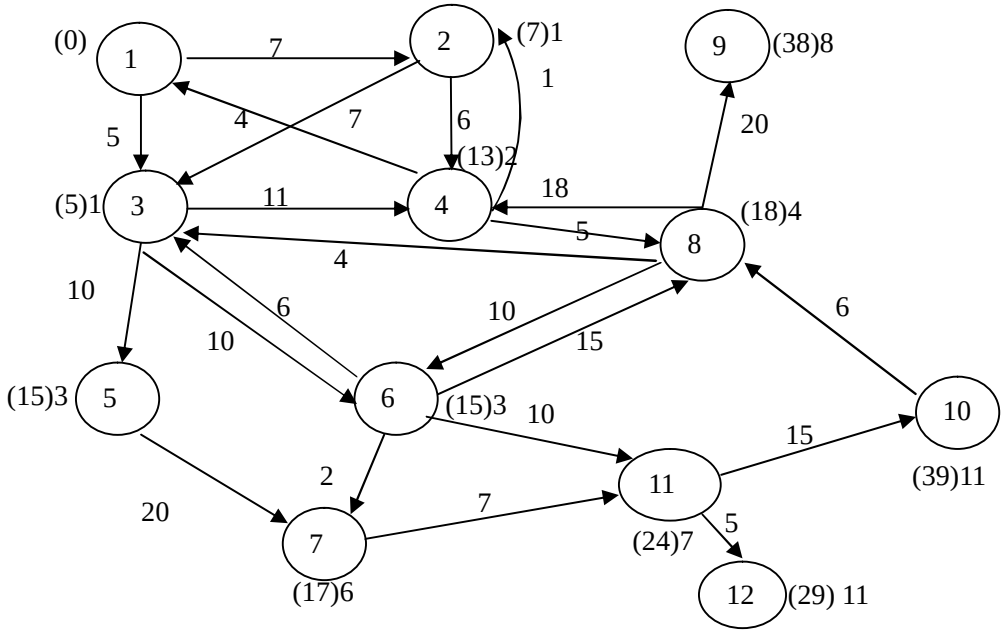
- Ở bộ xử lý phụ P_1 sẽ nhận kết quả được biểu diễn như trong đồ thị dưới đây:



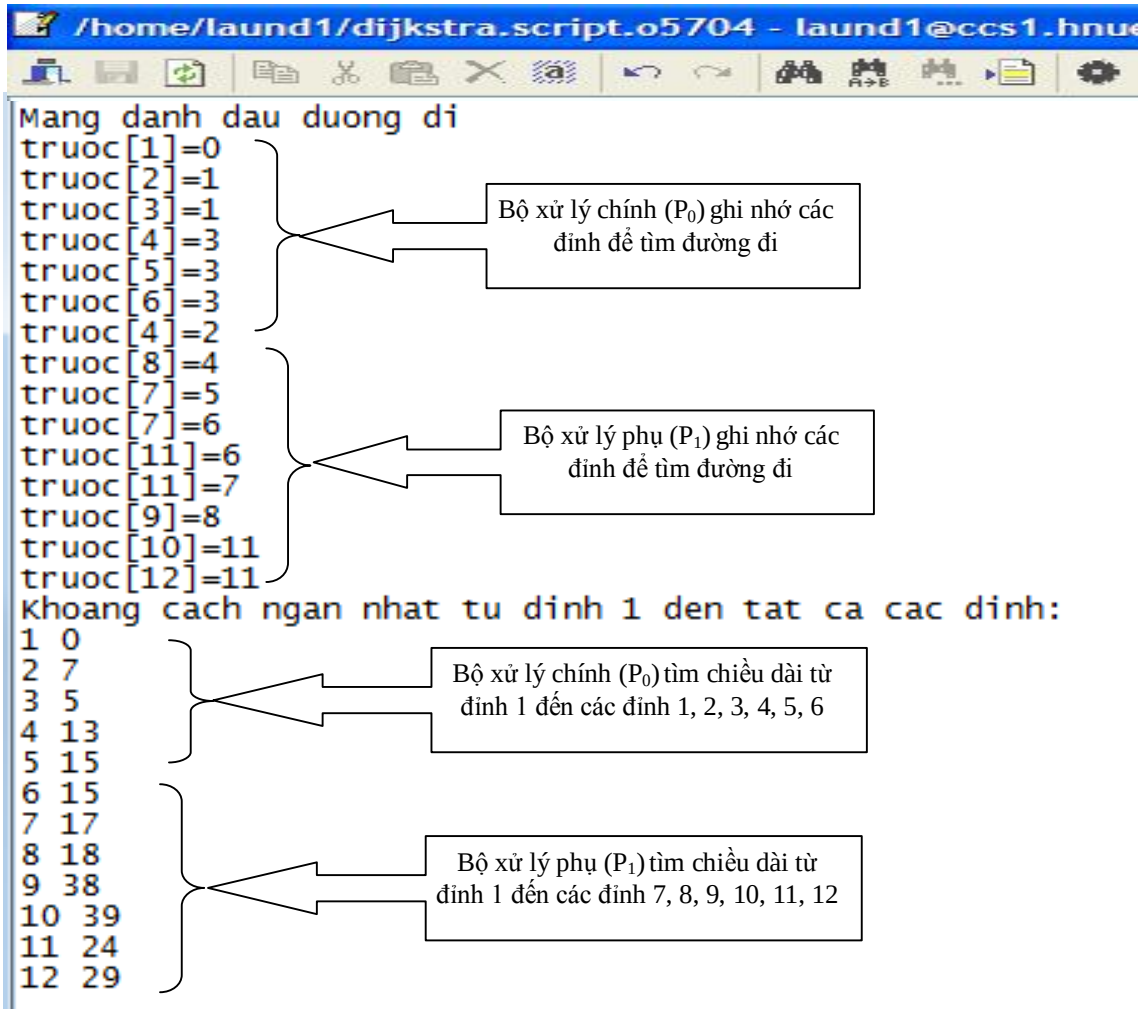
Hình 4. *Đồ thị ghi nhớ trên bộ xử lý chính (P_1)*

Bước 6.

Bộ xử lý chính kiểm tra bộ xử lý phụ kết thúc thì bộ xử lý chính sẽ nhận được kết quả mà bộ xử lý phụ gửi đến và hiển thị kết quả theo yêu cầu. Kết thúc hệ thống.



Hình 5. *Đồ thị hiển thị kết quả cuối cùng*



Hình 6. *Kết quả Demo*

Kết quả Demo trong phần thực nghiệm trên đồ thị 12 đỉnh được chạy trên cụm máy tính song song (Parallel computer cluster) của trường Đại Học Sư phạm Hà Nội với hệ thống toàn cầu *ccsl.hnue.edu.vn* tài khoản để chạy trên hệ thống này là *laundl*.

5. Kết luận

Với việc song song hóa thuật toán tuần tự Dijsktra tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh sẽ giúp ta giải quyết được các vấn đề bế tắc mà thuật toán tuần tự gặp phải như thời gian, dữ liệu đầu vào. Tuy nhiên để cài đặt thuật toán này đòi hỏi phải có cụm máy tính song song, cụ thể trong bài báo này chúng tôi dùng cụm máy tính song song của trường Đại học Sư phạm Hà Nội để chạy Demo. Thuật toán cho kết quả với thời gian xử lý nhanh hơn thuật toán tuần tự khi dữ liệu đầu vào lớn (chẳng hạn với số đỉnh của đồ thị từ 1000 trở đi).

TÀI LIỆU THAM KHẢO

[1]. Trần Quốc Chiến, Hồ Xuân Bình, *Thuật toán song song tìm luồng cực đại*, Tạp chí Khoa học & Công nghệ, Đại học Đà Nẵng, 5(22), (2007), 37-42.

[2]. Trần Quốc Chiến, Trần Thị Mỹ Dung, *Ứng dụng thuật toán tìm đường đi ngắn nhất Đa nguồn đích tìm luồng cực đại đa hàng hóa đồng thời*, Kỷ yếu hội thảo khoa học Công nghệ Thông tin, Trường Đại học Sư Phạm Đà Nẵng, 11/2011.

[3]. Tom Wilson, Nicholas Hofbauer, *Dijkstra's Algorithm in Parallel*, team report 2008 (http://www.cs.rit.edu/~ark/winter2008/531/team/u8/TeamReport.pdf).

[4]. A. Grama, A.Gupta, G.Karypis, V.Kumar, *Introduction to Parallel Computing*, 2003.

[5]. Seyed H. Roosta, *Parallel Processing and Parallel Algorithms, Theory and Computation*, Springer 1999.

[6]. Joseph JáJá, *An Introduction to Parallel Algrithms*, Addison - Wesley, 1992.

[7]. Alistair K Phipps, *Parallel algorithms for geometric shortest path problems*, Master of Science Computer Science School of Informatics University of Edinburgh, 2004.

[8]. Behrooz Parhami, *Introduction to Parallel Processing* (Algorithms and Architectures), University of California at Santa Barbara Santa Barbara, California, 2002.

PARALLELIZING ALGORITHM DIJKSTRA'S FINDING THE SHORTEST PATHS FROM A VERTEX TO ALL VERTICES

Nguyen Dinh Lau, Tran Ngoc Viet

College of Transport number II

Abstract. This article mainly focuses on the building of parallel algorithm in order to find the shortest paths from a vertice to all ones based on Dijkstra algorithm. The idea of this algorithm is using m processors to find the shortest paths from a vertice to other ones in a graph. Among k processors, we choose one central processor which plays a role in managing data, dividing n vertices and *weight matrix* into m processors to find the shortest paths.