Thuật toán song song phân luồng tuyến tính tối ưu trên mạng giao thông mở rộng

Parallel Algorithm to Divide Optimal Linear Flow on Extended Traffic Networks

Nguyễn Đình Lầu, Trần Quốc Chiến và Lê Mạnh Thạnh

Abstract: *Sequential algorithm to divide optimal linear flow on extended traffic network has been used in the project of Da Nang city, namely "Dividing traffic flow in Da Nang city". Furthermore, when sequential algorithms are applied to divide flow, a problem arises as there are a great number of roads and a growing number of the new routes built that leads to a huge number of variables (up to thousands of variables) on extended traffic network. So to process faster as well as take advantage of multi-core architecture, to process data with large scale with good results that requires the construction of parallel algorithm [6,7,8,9,10,11,12]. In this paper we build parallel algorithm to divide optimal linear flow on extended traffic network. The results in this paper are basically systematized and proven.*

Keywords: *processor, algorithm, maximun flow, optimal, parallel.*

I. GIỚI THIỆU

Trong các công trình [2,3,4,5] của chúng tôi và công trình [13] của *Naveen Garg, Jochen Könemann* đã xây dựng các bài toán *tìm luồng cực đại đa hàng hóa, tìm luồng cực đại đa hàng hóa đồng thời* và *tìm luồng cực đại đa hàng hóa đồng thời chi phí cực tiểu.* Các công trình này chỉ xét trên mạng giao thông bình thường, nghĩa là mạng giao thông chỉ xét đến khả năng thông hành của cạnh và chi phí cạnh mà chưa xét đến khả năng thông hành của đỉnh và chi phí tại mỗi đỉnh.

Trong công trình [1,10] chúng tôi định nghĩa đồ thị mở rộng, tức là đồ thị có thêm chi phí đỉnh và từ đó xây dựng thuật toán tuần tự và song song tìm đường đi ngắn nhất trên đồ thị mở rộng. Dựa vào đồ thị mở rộng chúng tôi định nghĩa mạng giao thông mở rộng cũng như tìm đường đi ngắn nhất trong mạng giao thông mở rộng.

Trong thực tế thời gian đi qua ngã tư trên mạng giao thông phụ thuộc vào hướng di chuyển của phương tiện giao thông: rẽ phải, đi thẳng hay rẽ trái, thậm chí có hướng bị cấm. Vì vậy cần xây dựng một mô hình mạng mở rộng để có thể áp dụng mô hình hóa các bài toán thực tế chính xác và hiệu quả hơn.

Thuật toán phân luồng tuyến tính tối ưu trên mạng giao thông mở rộng được giải quyết sẽ cải tiến hơn so với [2,3,4,5,13], vì trong [2,3,4,5,13] chỉ giải quyết cho mạng giao thông bình thường (không có khả năng thông hành đỉnh và chi phí tại mỗi đỉnh). Hơn nữa bài toán *phân luồng tuyến tính tối ưu* được phát triển từ thuật toán *tìm luồng cực đại tuyến tính đồng thời chi phí cực tiểu*, còn trong [2,3,4,5,13] các bài toán chỉ nghiên cứu ở mức cao nhất là *tìm luồng cực đại tuyến tính đồng thời chi phí cực tiểu* trên mạng giao thông bình thường.

Tuy nhiên khi chúng tôi thực nghiệm thuật toán tuần tự *phân luồng tuyến tính tối ưu trên mạng giao thông mở rộng* áp dụng cho các tuyến đường ở thành phố Đà nẵng (khảo sát chỉ hai Quận) thì thời gian cho ra kết quả là gần 100 phút, điều này nẩy sinh các vấn đề là khi chúng tôi khảo sát thêm các Quận khác, hoặc các tuyến đường ngày càng được bổ sung, hoặc áp dụng cho các thành phố lớn hơn thành phố Đà Nẵng thì chắc chắn thời gian xử lý sẽ rất lớn, thậm chí thuật toán tuần tự không xử lý được.

Vấn đề trên đòi hỏi chúng tôi phải xây dựng thuật toán song song nhằm giảm đi thời gian tính toán so với thuật toán tuần tự.

Bài báo gồm 3 phần chính, thứ nhất xây dựng thuật toán tuần tự, thứ hai là xây dựng thuật toán song song tương ứng, cuối cùng là kết luận. Các kết quả trong bài báo cơ bản được hệ thống và chứng minh.

II. THUẬT TOÁN TÌM LUỒNG CỰC ĐẠI ĐỒNG THỜI CHI PHÍ GIỚI HẠN

Trong phần này chúng tôi định nghĩa mạng giao thông mở rộng và xây dựng thuật toán tuần tự *tìm luồng cực đại đồng thời chi phí cực tiểu*.

II.1. Mạng giao thông mở rộng

Cho mạng là đồ thị hỗn hợp G=(V, E) với tập nút V và tập cạnh E. Các cạnh có thể có hướng hoặc vô hướng. Có nhiều loại phương tiện lưu hành trên mạng. Những cạnh vô hướng biểu diễn tuyến hai chiều, trong đó các phương tiện trên cùng tuyến nhưng ngược hướng lưu hành chia sẻ khả năng thông hành của tuyến. Trên mạng cho các hàm sau.

Hàm khả năng thông hành cạnh c_E: $E \to R^*$, với $c_E(e)$ là khả năng thông hành cạnh $e \in E$.

Hàm khả năng thông hành nút c_V: $V \to R^*$, với $c_V(u)$ là khả năng thông hành nút $u \in V$.

Hàm chi phí cạnh b_E:$E \to R^*$, với $b_E(e)$ là chi phí phải trả để chuyển một đơn vị phương tiện qua cạnh e. Lưu ý rằng với những tuyến hai chiều thì chi phí hai hướng có thể khác nhau. Với mỗi nút $v \in V$, ký hiệu E_v là tập tất cả các cạnh đi vào nút v hoặc đi ra từ nút v.

Hàm chi phí nút b_V: $V \times E_v \times E_v \to R^*$, với $b_V(u,e,e')$ là chi phí phải trả để chuyển một đơn vị phương tiện từ tuyến e qua nút u sang tuyến e'.

Bộ (V, E, c_E, c_V, b_E, b_V) gọi là *mạng giao thông mở rộng*.

Cho p là đường đi từ nút u đến nút v qua các cạnh e_i, $i = 1, \ldots, h+1$, và các nút u_i, $i = 1, \ldots, h$, như sau

$$p = [u, e_1, u_1, e_2, u_2, \ldots, e_h, u_h, e_{h+1}, v]$$

Định nghĩa *chi phí lưu hành* một đơn vị phương tiện qua đường đi p, ký hiệu $b(p)$, theo công thức sau:

$$b(p) = \sum_{i=1}^{h+1} b_E(e_i) + \sum_{i=1}^{h} b_V(u_i, e_i, e_{i+1}) \qquad (1)$$

II.2. Phát biểu bài toán luồng cực đại tuyến tính đồng thời chi phí cực tiểu

Cho mạng giao thông mở rộng G = (V, E, c_E, c_V, b_E, b_V). Giả thiết G có k cặp nút nguồn-đích (s_j, t_j), mỗi cặp được gán một loại phương tiện j, j=1, …, k. Ký hiệu Π_j là tập hợp các đường đi từ s_j đến t_j trong G, $j = 1, \ldots, k$, và đặt $\Pi = \bigcup_{j=1}^{k} \Pi_j$ Với mỗi đường đi $p \in \Pi_j$, $j=1, \ldots, k$, ký hiệu biến $x(p)$ là luồng phương tiện loại j lưu hành dọc theo đường đi p.

Ký hiệu Π_e là tập hợp các đường đi trong Π đi qua cạnh e, $\forall e \in E$.

Ký hiệu Π_v là tập hợp các đường đi trong Π đi qua nút v, $\forall v \in V$.

Mỗi loại phương tiện j có yêu cầu lưu hành $d(j)$ đơn vị phương tiện từ nút nguồn s_j đến nút đích t_j, $\forall j = 1, ..., k$. Cho giới hạn chi phí B. Nhiệm vụ của bài toán là tìm một số λ lớn nhất sao cho tồn tại một luồng chuyển $\lambda.d(j)$ đơn vị phương tiện j qua luồng, $\forall j = 1, ..., k$. Đồng thời, tổng chi phí của luồng không vượt quá B.

Bài toán được biểu diễn bằng mô hình qui hoạch tuyến tính như sau:

Tìm hệ $\{x(p) \mid p \in \prod_j, j = 1,..., k\}$ thỏa

$$\left.\begin{aligned} & \lambda \to max \\ & \sum_{p \in \Pi_e} x(p) \le c_E(e), \forall e \in E \\ & \sum_{p \in \Pi_v} x(p) \le c_V(v), \forall v \in V \\ & \sum_{p \in \Pi_j} x(p) \ge \lambda.d(j), \forall j = 1, \ldots, k \\ & \sum_{p \in \Pi} b(p).x(p) \le B \\ & x \ge 0, \lambda \ge 0 \end{aligned}\right\} \quad (P)$$

Bài toán qui hoạch tuyến tính đối ngẫu với (P), gọi là (D), được xây dựng như sau: mỗi cạnh $e \in$ E được gán một biến đối ngẫu *le*(*e*), mỗi nút $v \in$ V được gán một biến đối ngẫu *lv*(*v*), mỗi phương tiện *j*=1, ..., *k* được gán một biến đối ngẫu *z*(*j*) và biến đối ngẫu φ gán với ràng buộc về chi phí. Bài toán (D) phát biểu như sau:

$$\left.\begin{array}{l} D(le, lv, \varphi)= \sum_{e\in E} c_E(e).le(e) + \sum_{v\in V} c_V(v).lv(v) + B.\varphi \\ \rightarrow min \\ \sum_{e\in p} le(e) + \sum_{v\in p} lv(v) + b(p).\varphi \geq z(j), \forall j=1...k, \forall p \in \Pi \\ \sum_{j=1}^{k} d(j).z(j) \geq 1 \\ le, lv, z, \varphi \geq 0 \end{array}\right\} (D)$$

Bây giờ, cho *p* là đường đi từ nút *u* đến nút *v* qua các cạnh e_i, i = 1, …, h+1, và các nút u_i, i = 1, …, h, như sau

$$p = [u, e_1, u_1, e_2, u_2, \ldots, e_h, u_h, e_{h+1}, v]$$

Ta định nghĩa độ dài đường đi *p*, ký hiệu *length*(*p*), phụ thuộc các biến *le*, *lv* và φ theo công thức sau:

$$length(p) = \sum_{i=1}^{h+1} le(e_i) + \sum_{i=1}^{h} lv(u_i) + b(p).\ \phi$$

$$= \sum_{i=1}^{h+1} [\varphi.b_E(e_i) + le(e_i)] + \sum_{i=1}^{h} [\varphi.b_V(u_i, e_i, e_{i+1}) + lv(u_i)] \quad (2)$$

Ký hiệu $dist_j(le,lv,\varphi)$ là độ dài đường đi ngắn nhất từ s_j đến t_j tính theo hàm độ dài *length*(*p*), $\forall j$ = 1, ..., *k*.

Đặt $\alpha(le,lv,\varphi) = \sum_{j=1}^{k} d(j).dist_j(le,lv,\varphi)$.

Xét bài toán:

$$\beta = \min\left\{ \frac{D(le,lv,\varphi)}{\alpha(le,lv,\varphi)} \middle| le: E \rightarrow R^*, lv: V \rightarrow R^*, \varphi \geq 0 \right\} \quad (D_\alpha)$$

Bổ đề 1. Bài toán (D) tương đương với bài toán (D_α) theo nghĩa trị tối ưu của chúng bằng nhau và từ nghiệm tối ưu của bài toán này suy ra nghiệm tối ưu của bài toán kia và ngược lại.

Việc chứng minh bổ đề 1 được lập luận tương tự như trong [2, 3, 4, 5, 13].

II.3. Thuật toán tìm luồng cực đại tuyến tính đồng thời chi phí cực tiểu.

Ký hiệu $fe_j(a)$ là luồng phương tiện *j* đi qua cạnh *a*, *j* = 1, ...,*k*, $a \in$ E.

$fv_j(u,a,a')$ là luồng phương tiện *j* đi trên cạnh *a* vào nút *u* ra cạnh *a'*, *j* = 1, ..., *k*,

$u \in$ V, $a, a' \in E_u$.

Thuật toán tìm luồng

$$F = \{fe_j(a), fv_j(u,e,e') | \ \forall a \in E, \ \forall (e,u,e') \in \text{Bảng } b_V, j=1,...,k\}$$

Luồng F có thể vi phạm ràng buộc về khả năng thông qua cũng như ràng buộc về chi phí. Tuy nhiên, theo mục E ở phần II trong bài báo này, từ luồng *F* ta nhận được luồng tối ưu sau khi chia nó cho một hằng số. Khởi tạo: $fe_j(a)=0 \quad \forall a \in E$, $fv_j(u,e,e')=0$ $\forall (e,u,e') \in$ Bảng b_V, j=1,...,k

Chọn $\varepsilon > 0$ và $\delta > 0$ (giá trị ε và δ xác định ở phần phân tích sau).

Thuật toán được thực hiện bởi một số *giai đoạn*, mỗi giai đoạn gồm *k vòng lặp* (*k* là số phương tiện). Ở vòng lặp thứ *j*, *j* = 1, ..., *k*, của giai đoạn thứ *i* ta chuyển *d*(*j*) đơn vị phương tiện thứ *j* qua luồng. Việc này được thực hiện trong một số *bước*.

Vòng lặp thứ *j* của giai đoạn *i* kết thúc sau *q*(*i*,*j*) bước, khi mà $d_{i,j}^{q(i,j)} = 0$. Tổng luồng gửi qua mạng ở mỗi vòng lặp không vượt quá *d*(*j*) và chi phí ở mỗi bước không vượt quá B. Giai đoạn *i* kết thúc, khi vòng lặp thứ *k* của giai đoạn *i* kết thúc.

Hàm đối ngẫu *l* được tính như sau:

Hàm đối ngẫu ban đầu:

$le_{1,0}^0 = \delta c_E(e), \forall e \in$ E, $lv_{1,0}^0 = \delta c_V(v), \forall v \in$ V.

Hàm đối ngẫu ở đầu vòng lặp đầu tiên của mỗi giai đoạn *i* bằng hàm đối ngẫu ở cuối giai đoạn trước (*i*–1)

$$le_{i,0}^0 = le_{i-1,k}^{q(i-1,k)}, \ lv_{i,0}^0 = lv_{i-1,k}^{q(i-1,k)}$$

Hàm đối ngẫu ở đầu mỗi vòng lặp tiếp theo j của giai đoạn i có giá trị bằng hàm đối ngẫu ở cuối vòng lặp trước (j–1) của giai đoạn i:

$le^0_{i,j} = le^{q(i,j-1)}_{i,j-1}$, $lv^0_{i,j} = lv^{q(i,j-1)}_{i,j-1}$.

Tương tự, $\varphi^0_{1,0} = \delta$/B, $\varphi^0_{i,0} = \varphi^{q(i-1,k)}_{i-1,k}$, $\varphi^0_{i,j} = \varphi^{q(i,j-1)}_{i,j-1}$

Suy ra

$$D\left(le^0_{1,0}, lv^0_{1,0}, \varphi^0_{1,0}\right) = \sum_{e\in E} le^0_{1,0}(e).c_E(e) + \sum_{v\in V} lv^0_{1,0}(v).c_V(v) + \mathrm{B}.\varphi^0_{1,0}$$

$$= \sum_{e\in E}\frac{\delta}{c_E(e)}c_E(e) + \sum_{v\in V}\frac{\delta}{c_V(v)}c_V(v) + \mathrm{B}.\delta/\mathrm{B}$$

$$= m.\delta + n.\delta + \delta = (m+n+1)\delta$$

với m là số cạnh, n là số nút của mạng.

Ký hiệu le_i, lv_i, φ_i, $D(i)$, $\alpha(i)$ là hàm giá trị các đại lượng ở cuối mỗi giai đoạn i. Thuật toán dừng sau giai đoạn t, khi mà $D(t) \geq 1$.

II.4. Thuật toán tìm luồng cực đại tuyến tính đồng thời chi phí cực tiểu tựa ngôn ngữ C.

Thuật toán này đã được trình bày trong mục II.3 và chúng tôi trình bày lại nhưng tựa theo ngôn ngữ lập trình C như sau:

• *Đầu vào*:

Mạng mở rộng G = (V, E, c_E, c_V, b_E, b_V).

Nhu cầu (s_j, t_j, d_j), j=1, …, k.

Chi phí giới hạn B.

Hệ số xấp xỉ ω > 0.

• *Đầu ra*:

1) Hệ số λ cực đại: λ_{max}

2) Luồng thực tế {$fe_j(a)$, $fv_j(u,e,e')$| $a\in E$, $(e,u,e')\in$ Bảng b_v, j=1,...,k}.

3) Chi phí thực tế $B_f \leq B$.

• *Các bước*:

// *Khởi tạo các giá trị ban đầu*

Đặt $\varepsilon = 1 - \sqrt[3]{\frac{1}{1+\omega}}$; $\delta = \left(\frac{m+n+1}{1-\varepsilon}\right)^{-\frac{1}{\varepsilon}}$;

$le(e) = \delta/c_E(e)$, $\forall e \in$ E; $lv(v) = \delta/c_V(v)$, $\forall v \in$ V; φ = δ/ B;

D = $(m+n+1)\delta$;

$fe_j(a) = 0$; $\forall a\in E$,

$fv_j(u,e,e') = 0$; $\forall u\in$ V,$\forall (e,u,e')\in$ Bảng b_v, j=1,...,k

t = 1;//biến đếm giai đoạn

B_{ex} = 0;// Chi phí tạm tính

while ($D < 1$) // mức giai đoạn

{

for j = 1 to k do // mức vòng lặp ứng với j

{

$d' = d_j$ // lượng phương tiện cần chuyển từ s_j đến t_j

while $d' > 0$ do // mức các bước trong giai đoạn

{

Gọi *thủ tục tìm đường đi ngắn nhất* tìm đường đi ngắn nhất p từ s_j đến t_j theo hàm *length* công thức

$$length(p) = \sum_{i=1}^{h+1} le(e_i) + \sum_{i=1}^{h} lv(u_i)$$

$$+ \mathrm{b(p)}.\varphi = \sum_{i=1}^{h+1}\left[\varphi.b_E(e_i) + le(e_i)\right]$$

$$+ \sum_{i=1}^{h}\left[\varphi.b_V(u_i, e_i, e_{i+1}) + lv(u_i)\right]$$

Tính f'=min{d',$c_E(e)$,$c_V(v)$|$e\in p$, $v\in p$};

B' = $b(p)$*f';

// $b(p)$ tính theo công thức (1)

if B' > B {f' = f'*B/B'; B' = B};

// hiệu chỉnh luồng

$fe_j(a) = fe_j(a) + f'; \forall a \in p$

$fv_j(u,e,e') = fv_j(u,e,e') + f'; \forall (e,u,e') \in p$

// hiệu chỉnh các tham số khác

$d' = d' - f'$; $\varphi = \varphi*(1+\varepsilon*B'/B)$,

$le(e) = le(e)*(1+\varepsilon*f'/c_E(e)); \forall e \in p$

$lv(v) = lv(v)*(1+\varepsilon*f'/c_V(v)); \forall v \in p$

$D = D + \varepsilon*f'*length(p)$;

$B_{ex} = B_{ex} + B'$;

} //End while $d' > 0$

} //End for

t = t + 1;

} //End $D < 1$

// hiệu chỉnh luồng thực tế

$$c' = \max\{\frac{le(e)}{\delta / c_E(e)}, \frac{lv(v)}{\delta / c_V(v)}, \frac{\varphi}{\delta / B} | e \in E, v \in V\};$$

$c_{ex} = \log_{1+\varepsilon} c'$;

$fe_j(a) = fe_j(a)/c_{ex}; \forall a \in E, j=1,...,k$

$fv_j(u,e,e') = fv_j(u,e,e')/c_{ex}; \forall u \in V, \forall (e,u,e') \in$ Bảng $b_v, j=1,...,k$

$B_f = B_{ex}/c_{ex}$;// chi phí thực tế

$\lambda_{max} = \frac{t}{c_{ex}}$;// Tỉ lệ lớn nhất

// Hiệu chỉnh luồng trên các đoạn tuyến hai chiều có luồng cùng nguồn đích ngược chiều

for $e \in E$, $e = (u, v)$ hai chiều

for $j = 1$ to k do

if $(fe_j(u,v) > fe_j(v,u))$ and $(fe_j(v,u) > 0)$

{

$fe_j(u,v) = fe_j(u,v) - fe_j(v,u)$;

$B_f = B_f - (b_E(u,v) + b_E(v,u))* fe_j(v,u)$;

$fe_j(v,u) = 0$;

}

if $(fe_j(v,u) >= fe_j(u,v))$ and $(fe_j(u,v) > 0)$

{

$fe_j(v,u) = fe_j(v,u) - fe_j(u,v)$;

$B_f = B_f - (b_E(u,v) + b_E(v,u))* fe_j(u,v)$;

$fe_j(u,v) = 0$;

}

• *Nhận xét:* Trong (t–1) giai đoạn thực hiện thuật toán trên, $\forall j = 1, .., k$, ta đã chuyển $(t–1).d(j)$ đơn vị phương tiện qua luồng. Tuy nhiên, luồng đã chuyển có thể vượt quá khả năng thông qua của các cạnh và nút. Bổ đề sau đây giải quyết vấn đề trên.

• *Bổ đề 2.* $\lambda > \frac{t-1}{\log_{1+\varepsilon}(1/\delta)}$.

• *Bổ đề 3.* Cho $\omega > 0$. Khi đó tồn tại ε và δ sao cho luồng tìm được của thuật toán, sau khi chia cho $\log_{1+\varepsilon}(1/\delta)$, là luồng cực đại đồng thời chi phí cực tiểu với tỉ lệ xấp xỉ là $(1+\omega)$.

Việc chứng minh bổ đề 2, bổ đề 3 được lập luận tương tự như trong [2, 3, 4, 5, 13].

• *Bổ đề 4.* Tổng chi phí luồng trong (t–1) vòng lặp không vượt quá $B.\log_{1+\varepsilon}(1/\delta)$. Nghĩa là, tổng chi phí của luồng sau khi chia cho $\log_{1+\varepsilon}(1/\delta)$ không vượt quá B.

Chứng minh: Ta có $\varphi_{1,0} = \delta B$. Sau (t–1) giai đoạn được thực hiện, ta có $D(t–1) < 1$, tức là

$$\sum_{e \in E} le_{t-1}(e).c_E(e) + \sum_{v \in V} lv_{t-1}(v).c_V(v) + B.\varphi_{t-1} < 1.$$

Suy ra $\varphi_{t-1} < 1/B$. Hơn nữa, mỗi khi chuyển luồng qua mạng mà tổng chi phí tăng lên B, thì φ tăng lên một thừa số không nhỏ hơn $(1+\varepsilon)$. Vì vậy, gọi x là số lần thuật toán làm tăng chi phí lên B đơn vị, ta có $\varphi_{1,0}.(1+\varepsilon)^x \le \varphi_{t-1} \le 1/B \Rightarrow x \le \log_{1+\varepsilon}(1/\delta)$.

Vậy tổng chi phí của quá trình thực hiện là $B.\log_{1+\varepsilon}(1/\delta)$. Khi chia luồng đạt được cho $\log_{1+\varepsilon}(1/\delta)$ ta đồng thời có tổng chi phí giảm đi thừa số $\log_{1+\varepsilon}(1/\delta)$, thỏa mãn yêu cầu đặt ra của bài toán.

• ***Định lí 1.*** Thuật toán tính được luồng xấp xỉ cực đại với tỉ lệ $(1+\omega)$ có độ phức tạp là

$$O(\omega^{-2}.(2k.\log_2 k+m).\log_2 m.n^3).$$

trong đó k là số phương tiện, m là số cạnh đồ thị, n là số nút đồ thị.

Chứng minh : Trước tiên, chúng ta tìm số giai đoạn mà thuật toán đã thực hiện. Theo bổ đề 2 và do $\beta = \lambda$, ta có $1 \le \gamma = \frac{\beta}{t-1}\log_{1+\varepsilon}\frac{1}{\delta}$

$$\Rightarrow t \le 1 + \beta.\log_{1+\varepsilon}\frac{1}{\delta} \quad \Rightarrow t = \left\lceil \beta.\log_{1+\varepsilon}\frac{1}{\delta} \right\rceil,$$

trong đó, ε và δ phụ thuộc vào ω. Ngoài ra, t còn phụ thuộc vào β.

Nhìn lại, $\beta = min_l \frac{D(l)}{\alpha(l)} = \frac{\sum_{e\in E} c(e).l(e)}{\sum_{j=1}^{k} d(j).dist_j(l)}$

Ta có thể tăng hoặc giảm β bởi một thừa số r bằng cách nhân các $c(e)$ hoặc các $d(j)$ lên một thừa số r tương ứng (mà việc nhân này sẽ không ảnh hưởng đến kết quả của bài toán, vì sau đó ta có thể giảm hoặc tăng λ bởi thừa số r).

Gọi z_i là luồng cực đại của phương tiện i, $i = 1, \ldots, k$. Đặt $z = min_i z_i/d(i)$. Khi đó z chính là phân số của các yêu cầu cần vận chuyển các phương tiện một cách độc lập. Từ $\beta = \lambda$, suy ra $z/k \le \beta \le z$. Ta có thể chia các $c(e)$ hoặc các $d(j)$ cho một số sao cho $z/k = 1$, để thỏa mãn giả thuyết đưa ra ban đầu là $\beta \ge 1$. Lúc này ta cũng có $\beta \le k$.

Đặt $T = 2.\left\lceil \log_{1+\varepsilon}\frac{1}{\delta} \right\rceil$ tương ứng với $\beta \ge 2$.

Nếu thuật toán không dừng lại ở T giai đoạn thực hiện, thì chúng ta nhân đôi tất cả các $d(j)$, (tương đương với chia đôi β, nhưng vẫn thỏa $\beta \ge 1$), rồi thực hiện thuật toán tiếp T giai đoạn nữa. Nếu thuật toán vẫn chưa dừng, ta tiếp tục giải pháp trên đến khi thuật toán dừng và lưu ý lúc này vẫn thỏa $\beta \ge 1$.

Ta thấy mỗi khi thực hiện T giai đoạn thì β giảm đi một nửa. Nếu x là số lần thực hiện T giai đoạn, thì β giảm đi một thừa số 2^x. Ta có

$$1 \le \beta/2^x \Rightarrow 2^x \le \beta \le k \Rightarrow x \le \log_2 k$$

Vậy $t = T.x = 2.\log_2 k.\left\lceil \log_{1+\varepsilon}\frac{1}{\delta} \right\rceil$.

Thay $\delta = \left(\frac{m}{1-\varepsilon}\right)^{-\frac{1}{\varepsilon}}$ vào ta được

$$t = 2.\log k.\left\lceil \frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon} \right\rceil$$

Mặt khác, mỗi giai đoạn ta thực hiện k vòng lặp, nên số vòng lặp là $k.t$. Hơn thế, ở mỗi vòng lặp, ta thực hiện một số bước. Tiếp theo, ta đi tìm tổng số bước thực hiện của thuật toán. Ở mỗi bước, trừ bước sau cùng của mỗi vòng lặp, ta tăng độ dài của ít nhất một cạnh lên $(1+\varepsilon)$ lần. Xét cạnh e bất kì, ta có $l_0(e) = \delta/c(e)$ và $l_t(e) < 1/c(e)$ (do $D(t-1) < 1$). Gọi $t1$ là số bước thực hiện mà trong đó e là cạnh có khả năng thông qua cực tiểu trên đường được chọn tương ứng, suy ra

$$l_0(e).(1+\varepsilon)^{t1} \le l_t(e) < 1/c(e) \Rightarrow t1 < \log_{1+\varepsilon}(1/\delta)$$

Thay $\delta = \left(\frac{m}{1-\varepsilon}\right)^{-\frac{1}{\varepsilon}}$ vào ta được

$$t1 \le \left\lceil \frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon} \right\rceil.$$

Hơn nữa, vì đồ thị gồm m cạnh, ta có tổng số bước thực hiện, (trừ số bước sau cùng của mỗi vòng lặp, con số này bằng số vòng lặp), là $m.t1$. Vậy, tổng số bước thực hiện là

$$m.\left\lceil \frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon} \right\rceil + 2k.\log_2 k.\left\lceil \frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon} \right\rceil$$

$$= (2k.\log_2 k+m).\left\lceil \frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m}{1-\varepsilon} \right\rceil$$

$$= (2k.\log_2 k+m).\left\lceil \frac{\log_2 m}{\varepsilon.\log_2(1-\varepsilon)} \right\rceil$$

với $\varepsilon \le 1 - \sqrt[3]{\frac{1}{1+\omega}}$ = O(ω), trong đó mỗi bước thực hiện chính là tìm đường ngắn nhất giữa các cặp nút tương ứng. Việc tìm đường ngắn nhất được thực hiện qua *k* lần thực hiện thuật toán tìm đường đi ngắn nhất trên mạng mở rộng có độ phức tạp là $O(n^3)$ [1,10], suy ra độ phức tạp của thuật toán là $O(\omega^{-2}.(2k.\log_2 k+m).\log_2 m.n^3)$.

III. THUẬT TOÁN PHÂN LUỒNG ĐA PHƯƠNG TIỆN TUYẾN TÍNH TỐI ƯU TRÊN MẠNG GIAO THÔNG MỞ RỘNG

Thuật toán này được xây dựng dựa trên cơ sở sử dụng thuật toán tìm luồng tuyến tính cực đại đồng thời chi phí cực tiểu đã trình bày ở mục II. Vì đây là phương pháp xấp xỉ, nên mục tiêu thuật toán là tìm luồng tối ưu với hệ số cực đại đồng thời λ xấp xỉ 1, lớn hơn hệ số cực đại giới hạn $\lambda_{inf} \approx 1$.

Ý tưởng thuật toán thực hiện thuật toán tìm luồng cực đại đồng thời chi phí cực tiểu với chi phí giới hạn ban đầu tạm tính. Nếu hệ số cực đại đồng thời λ nhỏ hơn hệ số giới hạn λ_{inf} , thì tăng chi phí giới hạn và thực hiện thuật toán tìm luồng cực đại đồng thời chi phí cực tiểu với chi phí giới hạn mới. Quá trình này lặp lại cho đến khi đạt được hệ số cực đại đồng thời λ lớn hơn hoặc bằng hệ số giới hạn λ_{inf}.

- *Đầu vào*:

  Mạng mở rộng G = (V, E, c_E, c_V, b_E, b_V).

  Nhu cầu (s_j, t_j, d_j), j =1,…,k.

  Hệ số cực đại đồng thời giới hạn $\lambda_{inf} \approx 1$.

  Hệ số xấp xỉ ω > 0.

- *Đầu ra*:

1) Luồng tối ưu {$fe_j(a)$, $fv_j(u,e,e')$| $a \in E$, $(e,u,e') \in$ Bảng b_v, j=1,...,k}.

2) Chi phí cực tiểu B_{min}.

- *Các bước*:

// *Khởi tạo các giá trị ban đầu*

Chọn hệ số xấp xỉ ω > 0;

Chọn hệ số cực đại đồng thời giới hạn $\lambda_{inf} \approx 1$.

//Khởi tạo chi phí giới hạn B:

B = 0;

for j = 1 to k do

{ tìm đường đi ngắn nhất p từ s_j đến t_j
theo hàm chi phí $b(p)$;

B = B + d_j*$b(p)$;}

do {

Thực hiện chương trình tìm luồng cực đại đồng thời chi phí cực tiểu với tham số đầu vào B và ω, và cho hệ số cực đại λ_{max};

if ($\lambda_{max} < \lambda_{inf}$){ B = B / λ_{max} } }

while ($\lambda_{max} < \lambda_{inf}$)

//Hiệu chỉnh luồng tối ưu và chi phí cực tiểu

if ($\lambda_{max} > 1$)

{ $fe_j(a) = fe_j(a) / \lambda_{max}$; $\forall a \in$ E, j=1,...,k

$fv_j(u,e,e') = fv_j(u,e,e') / \lambda_{max}$;

$\forall u \in$ V,$\forall (e,u,e') \in$ Bảng b_v, j=1,...,k

$B_{min} = B_f / \lambda_{max}$;// chi phí thực tế }

Sau đây là ví dụ nhỏ minh họa thuật toán. Cho mạng giao thông mở rộng ở Hình 1. Mạng có 6 nút, 6 cạnh có hướng và 3 cạnh vô hướng. Chi phí cạnh w_E cho ở Bảng 1 và chi phí nút w_V cho ở Bảng 2. Khả năng thông qua của mỗi cạnh là *10*, khả năng thông qua của mỗi nút là *20*. Có 3 cặp nút nguồn đích (1, 5), (1, 4) và (3, 6) với lượng phương tiện tương ứng $d(1) = 15$, $d(2) = 8$ và $d(3) = 25$. Chọn hệ số xấp xỉ ω = 0.1.

Chương trình cho kết quả phân luồng tối ưu cho phương tiện đi từ nguồn 1 đến đích 5, nguồn 1 đến đích 4 và nguồn 3 đến đích 6 cho tương ứng ở Hình 2, Hình 3 và Hình 4.

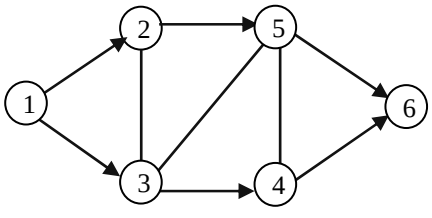
Tổng chi phí tối ưu là 692.

Bảng 1. Chi phí cạnh

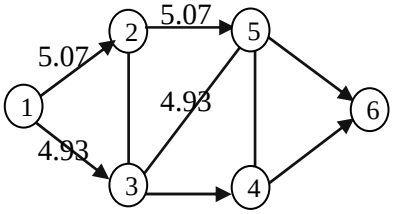
Cạnh	w_E
(1,2)	10
(1,3)	9
(2,3)	10
(2,5)	10
(3,4)	15
(3,5)	11
(4,6)	10
(4,5)	10
(5,6)	10

Bảng 2. Chi phí đỉnh

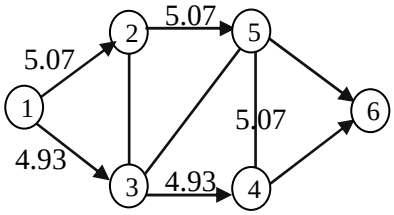
Nút	Cạnh 1	Cạnh 2	w_V
2	(1,2)	(2,3)	1
2	(1,2)	(2,5)	1
2	(3,2)	(2,5)	1
3	(1,3)	(3,4)	1
3	(1,3)	(3,5)	1
3	(1,3)	(3,2)	2
3	(5,3)	(3,2)	1
3	(5,3)	(3,4)	1
3	(2,3)	(3,4)	1
3	(2,3)	(3,5)	1
4	(3,4)	(4,6)	1
4	(3,4)	(4,5)	1
4	(5,4)	(4,6)	1
5	(2,5)	(5,3)	1
5	(2,5)	(5,4)	2
5	(2,5)	(5,6)	3
5	(3,5)	(5,4)	1
5	(3,5)	(5,6)	1
5	(4,5)	(5,3)	1
5	(4,5)	(5,6)	1



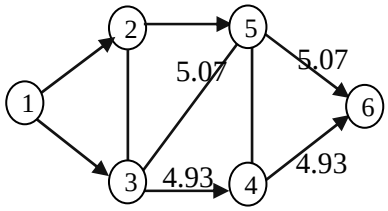
Hình 1. Mạng giao thông mở rộng



Hình 2. Phân luồng nguồn 1 đến đích 5



Hình 3. Phân luồng nguồn 1 đến đích 4



Hình 4. Phân luồng nguồn 3 đến đích 6

• ***Định lý 2.*** Giả thiết tồn tại phương án phân luồng đáp ứng nhu cầu đi lại của tất cả các cặp nguồn đích với chi phí B_{max}. Giả thiết hệ số cực đại đồng thời giới hạn $\lambda_{inf} < 1$ và hệ số xấp xỉ $\omega > 0$ thỏa $\lambda_{inf} \le 1/(1+\omega)$. Khi đó thuật toán kết thúc sau một số hữu hạn vòng lặp thực hiện chương trình tìm luồng cực đại đồng thời chi phí cực tiểu. Độ phức tạp thuật toán là

$$O(r.\omega^{-2}.(2k.\log_2 k+m).\log_2 m.n^3).$$

trong đó k là số cặp nguồn đích, m là số cạnh đồ thị, n là số nút đồ thị, $r=\log_{\lambda_{inf}} \frac{B_1}{B_{max}}+1$ và B_1 là chi phí giới hạn đầu vào ở vòng lặp đầu tiên.

Chứng minh. Theo chứng minh bổ đề 2, với

$$\varepsilon \le 1-\sqrt[3]{\frac{1}{1+\omega}}$$ ta có $\beta/\lambda < (1+\omega)$,

suy ra $\lambda > \beta/(1+\omega)$. Ký hiệu B_{max} là chi phí của phương án phân luồng đáp ứng nhu cầu đi lại của tất cả các cặp nguồn đích theo giả thiết. Ký hiệu $\lambda(B,\omega)$ là hệ số cực đại phụ thuộc tham số đầu vào B và ω. Như vậy bài toán luồng cực đại đồng thời chi phí cực tiểu với mọi chi phí $B \geq B_{max}$ sẽ cho hệ số cực đại đồng thời $\lambda(B,\omega) \geq 1/(1+\omega)$. Suy ra $\lambda(B,\omega) \geq \lambda_{inf}$

$\forall B \geq B_{max}$ (*).

Ký hiệu B_i là chi phí đầu vào của vòng lặp thứ *i*, *i*=1,2, …. Giả sử đến vòng lặp thứ q ta vẫn có $B_q < B_{max}$ và $\lambda(B_q,\omega) < \lambda_{inf}$. Theo cách tính của thuật toán, ta có

$$B_q = B_{q-1}/\lambda(B_{q-1},\omega) \geq B_{q-1}/\lambda_{inf} \geq B_{q-2}/(\lambda_{inf})^2 \geq \ldots \geq B_1/(\lambda_{inf})^{q-1}$$

Suy ra $B_1/(\lambda_{inf})^{q-1} < B_{max}$, kéo theo

$q < \log_{\lambda_{inf}} \frac{B_1}{B_{max}} + 1 = r$. Vậy, kết hợp với (*) suy ra với

$q \geq r$ thì $\lambda(B_q,\omega) \geq \lambda_{inf}$ và thuật toán dừng.

Theo định lý 1 độ phức tạp của mỗi vòng lặp là $O(\omega^{-2}.(2k.\log_2 k+m).\log_2 m.n^3)$. Số vòng lặp không quá *r*, từ đó suy ra độ phức tạp thuật toán là

$$O(r.\omega^{-2}.(2k.\log_2 k+m).\log_2 m.n^3).$$

IV. THUẬT TOÁN SONG SONG PHÂN LUỒNG ĐA PHƯƠNG TIỆN TUYẾN TÍNH TỐI ƯU TRÊN MẠNG GIAO THÔNG MỞ RỘNG

Độ phức tạp thuật toán tuần tự là:

$$O(r.\omega^{-2}.(2k.\log_2 k+m).\log_2 m.n^3).$$

Với độ phức tạp như vậy thì thời gian chạy tuần tự cho ứng dụng cụ thể là rất lớn (dù ứng dụng đó là nhỏ). Điều này cũng đã được chứng minh bằng thực tế là chúng tôi đã cho chạy thực nghiệm và Chương trình được sử dụng để phân luồng giao thông tối ưu cho mạng giao thông trung tâm thành phố Đà Nẵng – Việt Nam. Dữ liệu mạng giao thông này bao gồm 120 nút giao thông chính, 211 tuyến giao thông và 999 cặp nút nguồn đích với lưu lượng gần 50000 xe con quy đổi. Thời gian chạy là gần 100 phút.

Với thời gian như vậy đòi hỏi chúng tôi phải xây dựng thuật toán song song để giảm bớt thời gian tính toán và thực hiện được các ứng dụng mà dữ liệu đầu vào lớn mà thuật toán tuần tự không thể thực hiện được.

IV.1. Ý tưởng thuật toán song song

Ta xây dựng thuật toán trên c bộ xử lý $P_1,\ldots,P_c$. Trong c bộ xử lý đó ta chọn bộ xử lý chính P_1 đóng vai trò trung tâm, thực hiện quản lý dữ liệu, phân chia công việc, gửi dữ liệu đến c-1 bộ xử lý phụ $P_2,\ldots,P_c$. Bộ xử lý chính đồng thời thực hiện công việc giống các bộ xử lý phụ.

Bộ xử lý chính P_1 sẽ chia đều k bộ nhu cầu (s_j,t_j,d_j), j=1,…,k cho c bộ xử lý. Tuy nhiên để tận dụng hết khả năng của các bộ xử lý thì ta chia các bộ nhu cầu cho c bộ xử lý sao cho d_j (lượng phương tiện qua (s_j,t_j)) gần bằng nhau giữa các bộ xử lý.

c-1 bộ xử lý phụ nhận các bộ nhu cầu mà bộ xử lý chính gửi đến và thực hiện nhân gấp c lần nhu cầu d_j rồi thực hiện tính toán độc lập trên các bộ nhu cầu đó. Kết quả tính được trên c-1 bộ xử lý phụ sẽ gửi về bộ xử lý chính, bộ xử lý chính sẽ cộng các kết quả này lại và chia cho c và

$$\lambda_{max} = \min\{\lambda_{max\,1}, \lambda_{max\,2}, ..., \lambda_{max\,m}\}$$

Các tiến trình trên các bộ xử lý phụ $P_2,\ldots, P_c$ bắt đầu thực hiện bước 2 chỉ khi nhận được yêu cầu từ P_1. Tiến trình trên P_1 thực hiện bước 3 chỉ khi đã nhận đủ λmax_i từ các bộ xử lý P_i (i=1,…, c).

Trong phần B sau đây chúng tôi thiết kế thuật toán song song trên c bộ xử lý P_1, $P_2,\ldots$ P_c

IV.2. Các *bước thực hiện của thuật toán song song*

- *Đầu vào*:

  Mạng mở rộng $G = (V, E, c_E, c_V, b_E, b_V)$.

  Nhu cầu (s_j, t_j, d_j), j =1,…,*k*.

  Hệ số cực đại đồng thời giới hạn $\lambda_{inf} \approx 1$.

  Hệ số xấp xỉ $\omega > 0$, c bộ xử lý.

- *Đầu ra*:

1) Luồng tối ưu $\{fe_j(a), fv_j(u,e,e')| a \in E, (e,u,e') \in$ Bảng $b_v, j=1,...,k\}$.

2) Chi phí cực tiểu B_{min}.

Bước 1: Bộ xử lý chính P_1 thực hiện công việc sau:

- Nhận dữ liệu đầu vào.
- // *Khởi tạo các giá trị ban đầu*

Chọn hệ số xấp xỉ $\omega > 0$;

Chọn hệ số cực đại đồng thời giới hạn $\lambda_{inf} \approx 1$.

- //Khởi tạo chi phí giới hạn B

```
B = 0;
for j = 1 to k do
    { tìm đường đi ngắn nhất p từ s_j đến t_j
      theo hàm chi phí b(p);
      B = B + d_j*b(p);
    }
```

- Chuyển mạng G, B, ω, λ_{inf} đến c-1 bộ xử lý phụ.
- P_1 chia k nhu cầu (s_j,t_j,d_j), j=1,…,k cho c bộ xử lý $P_1, P_2, \ldots, P_c$

Cách chia như sau:

Đầu tiên ta chia bộ (s_1, t_1, d_1) cho bộ xử lý P_1, (s_2, t_2, d_2) cho $P_2,\ldots,(s_c, t_c, d_c)$ cho P_c

$sumpt_1=d_1$ (biến chứa tổng số phương tiện d_1 của P_1)

$sumpt_2=d_2$ (biến chứa tổng số phương tiện d_2 của P_2)

.
.
.

$sumpt_c=d_c$(biến chứa tổng số phương tiện d_c của P_c)

```
For t:=c+1 to k do
 {
    h:=1; min:=sumpt_h;
    for i:= 2 to c do
        If min> sumpt_i then
            min:=sumpt_i;
    h:=i;
    Đánh dấu để chia bộ nhu cầu (s_t, t_t, d_t)
    cho P_h
    sumpt_h:=sumpt_h+d_t;
 }
```

Bước 2: c bộ xử lý $P_1, P_2,\ldots,P_c$ thực hiện đồng thời các công việc sau đây:

Thực hiện chương trình tìm luồng cực đại đồng thời chi phí cực tiểu với tham số đầu vào B và ω, và cho hệ số cực đại λ_{maxi}, B^i_{ex} .

Chú ý rằng việc thực hiện tìm luồng cực đại đồng thời chi phí cực tiểu chỉ thực hiện trên số bộ nhu cầu mà P_i nhận ở bước 1 (s_{ji},t_{ji},d_{ji}), ji=1,…,k_i, đồng thời các nhu cầu phương tiện d_{ji} phải được nhân lên gấp c lần $(s_{ji},t_{ji},c*d_{ji})$, ji=1,…,k_i

c-1 bộ xử lý phụ gửi λ_{maxi} (i=2,...,c) lên bộ xử lý chính P_1.

Bước 3: Bộ xử lý chính P_1 thực hiện:

$$\lambda_{max}=\min\{\lambda_{max1}, \lambda_{max2},\ldots., \lambda_{maxc});$$

$$B_{ex} = (B^1_{ex}+B^2_{ex}, +,\ldots.+ B^c_{ex},)/c;$$

Bước 4: Bộ xử lý chính P_1 kiểm tra, nếu $\lambda_{max} < \lambda_{inf}$ thì gán $B=B/\lambda_{max}$, gửi B đến các bộ xử lý phụ, quay lại *bước 2*. Ngược lại, sang *bước 5*.

Bước 5: Bộ xử lý chính thực hiện hiệu chỉnh luồng tối ưu và chi phí cực tiểu như trong thuật toán tuần tự, nhưng tất cả giá trị đều phải chia cho c:

```
if (λ_max > 1)
    {   fe_j(a) = fe_j(a) / c*λ_max; ∀a∈E, j=1,...,k
        fv_j(u,e,e') = fv_j(u,e,e') / c*λ_max;
        ∀u∈V,∀(e,u,e')∈ Bảng b_v, j=1,...,k
        B_f=B_ex/c*c_ex;
        B_in = B_f / c*λ_max;// chi phí thực tế
    }
```

Kết thúc.

Trong phần sau đây chúng tôi sẽ diễn giải thuật toán mà c bộ xử lý thực hiện song song ở bước 2 của thuật toán song song ở mục IV.2.

IV.3. Các bước thực hiện thuật toán ở bước 2 trong mục IV.2 của P_i bộ xử lý (i=1,…,c) để tìm $\lambda_{max1},\ldots, \lambda_{maxc}$

```
T_i=1;
B^i_ex =0;
While D_i<1 do
```

{

For ji=1 to k_i do

{

$d^{'}$ =c*d_{ji}; //nhân c lần lượng phương tiện cần chuyển từ s_{ji} đến t_{ji}

While $d^{'}$ >0 do//mức các bước trong giai đoạn

{

Gọi thủ tục tìm đường đi ngắn nhất p^i từ s_{ji} đến t_{ji} theo hàm length công thức: length(p^i) theo công thức tuần tự ở trên.

Tính $f^{'} = \min\{d^{'}, c_E(e), c_V(v) \mid e \in p^i, v \in p^i\}$

$B^{'}$=b(p^i)*$f^{'}$;//b(p)tính theo công thức (1)

If $B^{'}$>B {$f^{'}$=$f^{'}$ *B/$B^{'}$; $B^{'}$=B};

// Hiệu chỉnh luồng

$$fe_{ji}(a) = fe_{ji}(a) + f^{'}; \forall a \in p^i;$$

$$fv_{ji}(u,e,e^{'}) = fv_{ji}(u,e,e^{'}) + f^{'}; \forall (e,u,e^{'}) \in p^i$$

//Hiệu chỉnh các tham số khác

$d^{'}$=$d^{'}$-$f^{'}$; $\varphi = \varphi * (1 + \varepsilon * B^{'} / B);$

$$lv(v) = lv(v) * (1 + \varepsilon * f^{'} / c_V(v)); \forall v \in p^i$$

$$le(e) = le(e) * (1 + \varepsilon * f^{'} / c_E(e)); \forall e \in p^i$$

$$D^i = D^i + \varepsilon * f^{'} * length(p^i);$$

B^i_{ex}= B^i_{ex} +$B^{'}$;

} //end while $d^{'}$>0

}// End for

t_i=t_i+1;

} //End D_i<1

//Hiệu chỉnh luồng thực tế

$$c^{'} = \max\left\{\frac{le(e)}{\delta / c_E(e)}, \frac{lv(v)}{\delta / c_V(v)}; \frac{\varphi}{\delta / B} \mid e \in E, v \in V\right\};$$

$$c_{ex} = \log_{1+\varepsilon} c^{'};$$

$$\lambda_{\max i} = \frac{t_i}{c_{ex}};$$

Ở ví dụ trên, khi thực hiện song song trên 2 bộ xử lý thì bộ xử lý P_1 nhận 1 bộ nhu cầu (1, 5, 15)còn bộ xử lý P_2 nhận 2 bộ nhu cầu (1, 4, 8),); (3, 6, 25) đồng thời 2 bộ xử lý nhân đôi lượng phương tiện rồi tính toán.

• ***Định lí 3.*** Thuật toán song song là đúng.

Chứng minh: Việc chứng minh dựa trên thuật toán tuần tự. Thuật toán song song được thực hiện qua 5 bước. Bước 1, bộ xử lý chính khởi tạo các biến giống như trong thuật toán tuần tự, chia số bộ nhu cầu cho c bộ xử lý và gửi đến các bộ xử lý phụ. Bước 2, c bộ xử lý thực hiện song song để tìm λ_{maxi} (i=1,...,c), sau đó gửi về bộ xử lý chính. Bước 3, bộ xử lý chính sẽ tìm

λ_{max}=min{λ_{max1}, λ_{max2},...., λ_{maxc});

B_{ex} = (B^1_{ex}+B^2_{ex}, +,….+ B^c_{ex},)/c;

Bước 4, kiểm tra tính kết thúc của bài toán. Bước 5, bộ xử lý chính thực hiện hiệu chỉnh luồng tối ưu và chi phí cực tiểu

Trong thuật toán song song điều kiện ràng buộc của bài toán là số bộ nhu cầu k phải lớn hơn c bộ xử lý. Việc chia số lượng phương tiện d_j cho c bộ xử lý phải tương đối bằng nhau để tránh các bộ xử lý quá nhàn rỗi. Ví dụ nếu có 2 bộ xử lý, bộ xử lý thứ nhất nhận 100 lượng phương tiện, còn bộ xử lý 2 nhận 1 lượng phương tiện thì thuật toán song song sẽ không cải tiến nhiều về mặt thời gian, thậm chí thời gian còn nhiều hơn so với thuật toán tuần tự. Theo cách chứng minh độ phức tạp của thuật toán ở định lý 1, ta có thể tăng hoặc giảm β bởi một thừa số *r* bằng cách nhân các *c*(*e*) hoặc các *d*(*j*) lên một thừa số *r* tương ứng (mà việc nhân này sẽ không ảnh hưởng đến kết quả của bài toán, vì sau đó ta có thể giảm hoặc tăng λ bởi thừa số *r*).

Vì vậy việc chia số bộ nhu cầu cho c bộ xử lý và nhân c lần *d(j)* để các bộ xử tính toán độc lập vẫn đảm bảo tính đúng đắn của thuật toán.

Hơn nữa, việc chia k bộ nhu cầu phải thỏa k_1+k_2+…+k_c=k, với k_1 là số bộ nhu cầu của P_1, k_2 là số bộ nhu cầu của P_2, …, k_c là số bộ nhu cầu của P_c. Gọi k_1 là số bộ nhu cầu của bộ xử lý P_1, ta có P_1 sẽ nhận các nhu cầu (s_{j1},t_{j1},d_{j1}), j1=1,…,k_1. Gọi k_2 là số bộ nhu cầu của bộ xử lý P_2, ta có P_2 sẽ nhận các nhu cầu (s_{j2},t_{j2},d_{j2}), j2=k_1+1, k_1+2…, k_1+k_2. Gọi k_c là số bộ

nhu cầu của bộ xử lý P_c nhận, ta có P_c sẽ nhận các nhu cầu (s_{jc},t_{jc},d_{jc}), jc = $k_1+k_2+\ldots+k_{c-1}+1$, $k_1+k_2+\ldots+k_{c-1}+2,\ldots$, $k_1+k_2+\ldots+k_c$. Thỏa mãn $k_1+k_2+\ldots+k_c=k$.

IV.4. Phân tích độ phức tạp

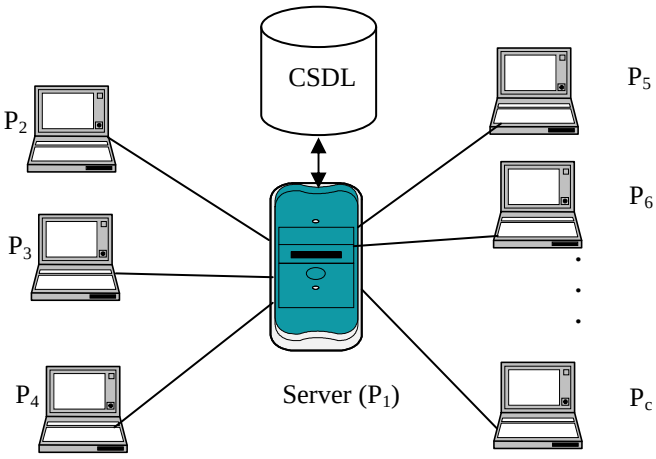
Độ phức tạp tuần tự $T_s=O(r.\omega^{-2}.(2k.\log_2 k+m).\log_2 m.n^3)$. Trong đó m số cạnh, n là số đỉnh. Khi thực hiện song song chia k nhu cầu cho c bộ xử lý nên thời gian của thuật toán song song là T_P

$T_p = O(r.\omega^{-2}.((2k.\log_2 k)/c^2+m).\log_2 m.n^3)+T$

Trong đó T là thời gian truyền thông

IV.5. Mô hình thực nghiệm

Hệ thống thực nghiệm ở đây là mạng LAN, các máy tính có cấu hình thông dụng trên thị trường và chương trình được xây dựng bằng ngôn ngữ Java với hệ quản trị cơ sở dữ liệu My SQL. Mô hình có c bộ xử lý, bộ xử lý trung tâm P_1 được gọi là Server (Hình 5)

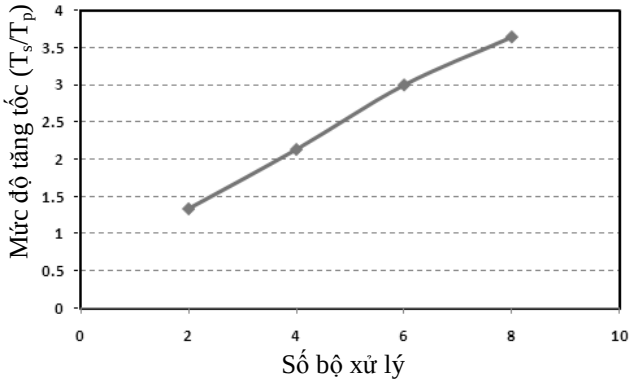


Hình 5. Mô hình phân tán đa bộ xử lý
(Mô hình mạng LAN)

Cho mạng giao thông ở thành phố Đà Nẵng như đã giới thiệu ở mục IV, thì thời gian chạy trên 1, 2, 4, 6, 8 bộ xử lý cho kết quả được biểu diễn ở Hình 6 trên (T_s là thời gian chạy tuần tự, T_p là thời gian chạy song song, T_s/T_p là mức độ tăng tốc (*speedup*)).

IV.6. Kết quả thử nghiệm

Kết quả trên Hình 6 cho thấy mức độ tăng tốc ở 2 bộ xử lý là gần 1.5; 4 bộ xử lý là gần 2.1 và mức độ này tăng lên đáng kể khi ta dùng 6 hoặc 8 bộ xử lý. Điều này chứng tỏ rằng thời gian chạy song song trên 2, 4, 6, 8 bộ xử lý sẽ giảm so với thời gian chạy tuần tự. Bảng 3 biểu diễn thời gian tính toán, hệ số cực đại đồng thời và số vòng lặp trên các bộ xử lý.



Hình 6. Biểu đồ biểu diễn mức độ tăng tốc

Bảng 3. Biểu diễn các tham số thực hiện song song

Số bộ xử lý	Số bước lặp(tỷ)	Giá trị λ_{max}	Thời gian tính (phút)
1	24.8976	1.0120	99.26745
2	19.0123	1.0122	74.37825
4	10.3427	1.0123	46.72799
6	7.32456	1.0132	33.31926
8	5.67601	1.0134	27.51636

Nhìn vào mức độ tăng tốc và Bảng 3 trên, chúng ta thấy rằng việc đồng bộ hóa các tiến trình song song trên nhiều bộ vi xử lý độc lập do việc trao đổi dữ liệu và chờ tất cả các bộ xử lý tìm xong các λ_{maxi}. Điều này làm suy giảm đáng kể hiệu năng của thuật toán song song.

V. KẾT LUẬN

Trong bài báo này chúng tôi đã xây dựng được thuật toán tuần tự và song song phân luồng tối ưu tuyến tính trên mạng giao thông mở rộng. Thuật toán song song chạy trên các bộ xử lý khác nhau để phân luồng cho mạng giao thông ở thành phố Đà Nẵng với dữ liệu mạng giao thông bao gồm 120 nút giao thông chính, 211 tuyến giao thông và 999 cặp nút nguồn

đích với lưu lượng gần 50000 xe con quy đổi. Khi thực hiện thì thời gian đã giảm đi cụ thể là với 1 bộ xử lý thì thời gian thực hiện hơn 99 phút, 2 bộ xử lý thì thời giảm xuống còn hơn 74 phút, 4 bộ xử lý thì thời lại tiếp tục giảm còn hơn 46 phút, 6 bộ xử lý thì thời gian giảm còn hơn 33 phút và 8 bộ xử lý thì thời gian hơn 27 phút (thể hiện ở Bảng 3). Thuật toán song song xây dựng giúp chúng tôi chạy các ứng dụng với quy mô dữ liệu lớn sẽ thực hiện được mà thuật toán tuần tự chờ rất lâu hoặc thậm chí xử lý tuần tự không thực hiện được. Các kết quả cơ bản được hệ thống và chứng minh.

CƠ SỞ THỰC HIỆN/TÀI TRỢ CÔNG TRÌNH NGHIÊN CỨU

Trường ĐH Sư Phạm Đà Nẵng, UBND TP Đà Nẵng, Sở Giao thông Đà Nẵng, Cục đường bộ 3- Bộ giao thông Vận tải.

TÀI LIỆU THAM KHẢO

[1] TRẦN QUỐC CHIẾN, *Thuật toán tìm đường đi ngắn nhất trên đồ thị tổng quát*, Tạp chí Khoa học & Công nghệ, Đại học Đà Nẵng, 12(61)/2012, 16-21.

[2] TRẦN QUỐC CHIẾN, *Ứng dụng thuật toán tìm đường đi ngắn nhất đa nguồn đích tìm luồng cực đại đa hàng hóa đồng thời*. Tạp chí Khoa học & Công nghệ, Đại học Đà nẵng, 4(53)2012, 76-83.

[3] TRẦN QUỐC CHIẾN, *Ứng dụng thuật toán tìm đường đi ngắn nhất đa nguồn đích tìm luồng cực đại đa hàng hóa đồng thời chi phí cực tiểu*. Tạp chí Khoa học & Công nghệ, Đại học Đà nẵng, 5(54)2012, 86-93.

[4] TRẦN QUỐC CHIẾN, TRẦN THỊ MỸ DUNG, *Ứng dụng thuật toán tìm đường đi ngắn nhất tìm luồng cực đại đa hàng hóa*, Tạp chí Khoa học và Công nghệ, Đại học Đà nẵng, số 3 (44), 2011,119-126.

[5] TRẦN QUỐC CHIẾN, *Bài toán mạng giao thông đa phương tiện tuyến tính*, Đề tài NCKH cấp bộ, mã số B2010DN-03-52. 2010

[6] NGUYỄN ĐÌNH LẦU, TRẦN NGỌC VIỆT, *Thuật toán song song tìm đường đi ngắn nhất của nhiều cặp đỉnh nguồn đích trên đồ thị*, Tạp chí Khoa học và Công nghệ, Đại học Đà Nẵng, số 9 (58) quyển I, 2012, 30-34.

[7] NGUYỄN ĐÌNH LẦU, TRẦN NGỌC VIỆT, *Song song hóa thuật toán tìm đường đi ngắn nhất của tất cả các đỉnh trên hệ thống cụm máy tính*, Kỷ yếu hội thảo quốc gia lần thứ 15 những vấn đề chọn lọc của công nghệ thông tin và truyền thông, Viện công nghệ thông tin 02-03/12/2012 .

[8] NGUYỄN ĐÌNH LẦU, TRẦN NGỌC VIỆT, *Song song hóa thuật toán dijkstra tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh*, Tạp chí Khoa học, Đại học Huế, Tập 74B, Số 5, (2012), 81-92.

[9] TRẦN QUỐC CHIẾN, LÊ MẠNH THẠNH, NGUYỄN ĐÌNH LẦU, *Thuật toán tuần tự và song song đẩy luồng trước tìm luồng cực đại*, Chuyên san số đặc biệt các công trình về Điện tử, Viễn thông và Công nghệ thông tin của Viện Hàn Lâm Khoa Học và Công Nghệ Việt Nam & Học Viện Bưu Chính Viễn Thông, số 51(4A)2013, 109-125.

[10] TRẦN QUỐC CHIẾN, LÊ MẠNH THẠNH, NGUYỄN ĐÌNH LẦU, *Thuật toán song song tìm đường đi ngắn nhất trên mạng mở rộng*, báo cáo tại đại hội Toán học lần thứ 8 (Nha Trang 10-14/8/2013).

[11] TRẦN QUỐC CHIẾN, LÊ MẠNH THẠNH, NGUYỄN ĐÌNH LẦU, *Thuật toán tuần tự và song song tìm luồng cực đại bằng phương pháp hỗn hợp đẩy kéo luồng*, Kỷ yếu hội nghị FAIR 2013, Huế 20-21/6/2013.

[12] CHIEN TRAN QUOC, LAU NGUYEN DINH, TRINH NGUYEN THI TU, *Sequential and Parallel Algorithm by Postflow-Pull Methods to Find Maximum Flow*, Proceedings 13th International Conference on Computational Science and Its Applications, ISNB:978-0-7695-5045-9/13 $26.00 © 2013 IEEE, DOI 10.1109/ICCSA.2013.36, 178-181

[13] NAVEEN GARG, JOCHEN KONEMANN: *Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems, SIAM J. Comput*, Canada, 37(2), 2007, pp. 630-652.

Nhận bài ngày: 03/09/2013

SƠ LƯỢC VỀ TÁC GIẢ

NGUYỄN ĐÌNH LẦU

Sinh năm 1978 tại Điện Bàn, Quảng Nam.

Tốt nghiệp đại học ngành Toán –Tin năm 2000 tại trường ĐH Khoa học (Huế), thạc sĩ CNTT năm 2008 tại trường ĐH Bách khoa, ĐH Đà Nẵng.

Hiện công tác tại Trường CĐ Giao thông Vận tải II-Bộ Giao thông Vận tải và đang NCS tại trường ĐH Bách khoa, ĐH Đà Nẵng.

Lĩnh vực nghiên cứu: Toán ứng dụng trong Giao thông vận tải, xử lý song song và phân tán, toán rời rạc, lý thuyết đồ thị, tính toán lưới, lập trình phân tán.

Điện thoại: 05113842266

Email: launhi@gmail.com

TRẦN QUỐC CHIẾN

Sinh năm 1953 tại Điện Bàn, Quảng Nam.

Tốt nghiệp đại học chuyên ngành Toán-Tin; Tiến sĩ khoa học chuyên ngành Toán điều khiển năm 1985 tại Trường ĐH Tổng hợp Sac-lơ, Praha - Tiệp Khắc cũ.

Được phong học hàm Phó Giáo sư năm 1992.

Hiện đang công tác tại trường ĐH Sư phạm Đà Nẵng.

Lĩnh vực nghiên cứu: Toán, tin học

Điện thoại: 0511.3841429

Email: dhsp@dng.vnn.vn

LÊ MẠNH THẠNH

Sinh năm 1953 tại Bố Trạch, Quảng Bình.

Nhận bằng tiến sĩ về khoa học máy tính năm 1994. Được phong học hàm PGS năm 2004.

Hiện đang công tác tại ĐH Huế.

Lĩnh vực nghiên cứu: Cơ sở dữ liệu, thuật toán, chương trình Datalog, ngôn ngữ hình thức và Otômát hữu hạn.

Điện thoại: 0914425355

Email: lmthanh@hueuni.edu.vn