Applying algorithm finding shortest path in the multiple-weighted graphs to find maximal flow in extended linear multicommodity multicost network

Chien Tran Quoc 1, Hung Ho Van 2

1 University of Da Nang, tqchien@dce.udn.vn
2 Quang Nam University, hovanhung@qnamuni.edu.vn

Abstract

The shortest path finding algorithm is used in many problems on graphs and networks. This article will introduce the algorithm to find the shortest path between two vertices on the extended graph. Next, the algorithm finds the shortest path between the pairs of vertices on the extended graph with multiple weights is developed. Then, the shortest path finding algorithms is used to find the maximum flow on the multicommodity multicost extended network is developed in the article [12].

Key word: Graph; Network; Multicommodity Multicost flow; Optimization; Linear Programming.

Received on 12 October 2017, accepted on 7 December 2017, published on 21 December 2017

Copyright © 2017 Chien Tran Quoc and Hung Ho Van et al., licensed to EAI. This is an open access article distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/3.0/), which permits unlimited use, distribution and reproduction in any medium so long as the original work is properly cited.
doi: 10.4108/eai.21-12-2017.153499

2 Corresponding author: Hung Ho Van. Email: hovanhung@qnamuni.edu.vn

1. Introduction

The shortest path finding algorithm is used in many problems on graphs and networks. This article will introduce the algorithm to find the shortest path between two vertices on the extended graph. Next, the algorithm finds the shortest path between the pairs of vertices on the extended graph with multiple weights is developed. Then, the shortest path finding algorithms is used to find the maximum flow on the multicommodity multicost extended network is developed in the article [12].

2. The problem of finding the shortest path in extended graph

Given extended graph $G = (V, E)$ with a set of vertices $V$ and a set of edges $E$, where edges can be directed or undirected. Each edge $e \in E$ is assigned a weight $w(e)$. For each vertex $v \in V$, we denote $E_v$, the set of edges incident vertex $v$. For each vertex $v \in V$ and each of pair of edges $(e, e') \in E_v \times E_v$, $e \neq e'$ is assigned switch weight $w_{v}(v, e, e')$.

The sets $(V, E, w_e, w_v)$ are called extended graph.

Let $p$ be a path from a vertex $u$ to a vertex $v$ through the edges $e_i, i = 1, ..., h+1$, and vertices $u_i, i = 1, ..., h$, as following:

$$p = [u, e_1, u_1, e_2, u_2, ..., e_h, u_h, v] \quad (1)$$

Define the length of the path $p$, denoted $l(p)$, as following:

$$l(p) = \sum_{j=1}^{h+1} w_e(e_j) + \sum_{j=1}^{h} w_v(u_j, e_j, e_{j+1}) \quad (2)$$

- The problem of finding the shortest path

Given extended graph $G = (V, E, w_e, w_v)$ and vertices $s, t \in V$. Find the shortest path from $s$ to $t$.

- Algorithm

  - Input. The extended graph $G = (V, E, w_e, w_v)$ and vertices $s, t \in V$.
  - Output. $l(t)$ is the length of the shortest path from $s$ to $t$, and the shortest path (if $l(t) < +\infty$).
Procedure

The algorithm uses the following notations:

- $S$ is a set of the vertices that found the shortest path starting from $s$.
- $T = V - S$;
- $l(v)$ is the length of the shortest path from $s$ to $v$;
- $l_e(v)$ is the edge that leads to the vertex $v$ on the shortest path from $s$ to $v$;
- $VE = \{(v, e) \mid v \in V - \{s\} \& e \in E, \cup \{(s, \emptyset)\}\}$ is the set of pairs of vertices and incident edges;
- $SE$ is a set of vertex-edge excluded from $VE$;
- $TE = VE - SE$;
- $L(v, e)$ is the label of the vertex-edge pair $(v, e) \in VE$.

\[ P(v, e) = \emptyset; \]

// Initialization

Assign to

- \[ S = \emptyset; T = V; \]
- \[ VE = \{(v, e) \mid v \in V - \{s\} \& e \in E, \cup \{(s, \emptyset)\}\}; \]
- \[ SE = \emptyset; TE = VE; \]
- \[ L(v, e) = +\infty; \forall (v, e) \in VE, L(s, \emptyset) = 0; \]
- \[ for (v, e) \in VE: P(v, e) = \emptyset; \]

do

\{ Calculate $m = \min\{L(v, e) \mid (v, e) \in TE\}.$

if ($m < +\infty$)

\{

Choose $(v_{min}, e_{min}) \in TE$ such that

\[ L(v_{min}, e_{min}) = m; \]

\[ TE = TE - \{(v_{min}, e_{min})\}; SE = SE \cup \{(v_{min}, e_{min})\}; \]

if $(v_{min} \notin S)$

\{

\[ l(v_{min}) = e_{min}, S = S \cup \{v_{min}\}; \]

\[ l(v_{min}) = L(v_{min}, e_{min}) + T = T - \{v_{min}\}; \]

\}

if $(t \not< v_{min})$

\{

for $(v, e) \in TE$ adjacent after $(v_{min}, e_{min})$

\{

if $(v_{min} = s)$

\[ L'(v, e) = L(s, \emptyset) + w(v_{min}, v); \]

else

\[ L'(v, e) = L(v_{min}, e_{min}) + w(v_{min}, v) + w(v_{min}, e_{min}) + w(v_{min}, e); \]

if $(L(v, e) > L'(v, e))$

\[ L(v, e) = L'(v, e); P(v, e) = \{(v_{min}, e_{min})\}; \]

\}

\}

\}

\}

} while $(m < +\infty$ or $t \not< v_{min}$)

if $(m = +\infty)$ 'no path exists from $s$ to $t$';

else // finding the shortest path

\{

Assign to \[ l(t) = L(t, l(e(t))); \] shortest path length from $s$ to $t$.

// Moves from $t$, in reverse direction, to the preceding vertex-edges, we get the shortest path as follows:

\[ k = 1; (v_k, e_k) = P(t, l(e(t))); \]

while $((v_k, e_k) \not< (s, \emptyset))$

\{

\[ k = k + 1; (v_k, e_k) = P(v_{k-1}, e_{k-1}); \]

\}

// Describe the shortest path is

\[ s \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_1 \rightarrow t \]

// End

- \[ Theorem \] 2.1. The algorithm that finds the shortest path in the extended graph is correct and has an algorithmic complexity of \[ O(n^3) \] (n is the number of vertices in the graph).

- \[ Proof \] [7] [8]

3. The problem of finding the shortest path on the multiple-weighted extended graph

Given the extended graph \[ G = (V, E) \] with a set of vertices $V$ and a set of edges $E$, where edges can be directed or undirected. On the graph there are $r$ edge weights $w_{i}$, and switch weights $w_{v_{r}}$, $i = 1...r$

The set \[ (V, E, \{w_{i}, w_{v_{r}} \mid i = 1...r\}) \] is called the multiple-weighted extended graph

Let $p$ be the path from the $u$ to $v$ through the edges $e_{i}$, $i = 1, ..., h+1$, and vertices $u_{i}, i = 1, ..., h$, as follows

\[ p = [u, e_{1}, u_{2}, e_{2}, u_{3}, ..., e_{h}, u_{h}, e_{h+1}, v] \]

Define the length of the path $p$ by edge weight $w_{i}$, and switch weights $w_{v_{r}}$ the symbol \[ l(p), \] $i = 1...r$, using the following formula:

\[ l(p) = \sum_{j=1}^{h+1} w_{i}(e_{j}) + \sum_{j=1}^{h} w_{v_{r}}(u_{j}, e_{j}, e_{j+1}) \]

- The problem of finding the shortest path

Given the multiple-weighted extended graph \[ G = (V, E, \{w_{i}, w_{v_{r}} \mid i = 1...r\}) \]. Assume for each weight $i$, $i = 1...r$, there are $k_{i}$ source-destination pairs $s_{i,j}, t_{i,j}, j = 1...k_{i}$.

The path length from the source node $s_{i,j}$ to the destination node $t_{i,j}$ is given by the function \[ l_{i}, i = 1...r, j = 1...k_{i}. \]
Applying algorithm finding shortest path in the multiple-weighted graphs to find maximal flow in extended linear multicommodity multicommodity network

The problem is to find, among the source-destination pairs \((s_{i,j}, t_{i,j})\), \(i=1..r, j=1..k_i\), the one that has the smallest shortest path length.

**Algorithm**

- **Input**: Multiple-weighted extended graph \(G = (V, E, \{w_e, w_{wv} \| i=1..r\})\). The source-destination pairs \((s_{i,j}, t_{i,j})\), \(i=1..r, j=1..k_i\).
- **Output**: The source-destination pair \((s_{i_min,j_min}, t_{i_min,j_min})\) with the shortest path length. \(l_{min}\) is the shortest path length from \(s_{i_min,j_min}\) to \(t_{i_min,j_min}\) and the shortest path (if \(l_{min} < +\infty\)).

**Procedure**

\(l_{min} = +\infty\):

for \(i=1; i<=r; i++\)

for \(j=1; j <= k_i; j++\)

\[
\begin{align*}
S &= \emptyset; T = V; \\
VE &= \{(v,e) \in V - \{s_{i,j}\} \land e \in E; \} \cup \{(s_{i,j}, \emptyset)\}; \\
SE &= \emptyset; TE = VE; \\
L(v,e) &= +\infty; V(v,e) \in VE, L(s_{i,j}, \emptyset) = 0; \\
&\text{for } (v,e) \in VE: P(v,e) = \emptyset; \\
&\text{do } [ \\
&\text{Calculate } m = \min\{L(v,e) \land (v,e) \in TE\}. \\
&\text{if } (m < l_{min}) \\
&\quad \text{Choose } (v_{min}, e_{min}) \in TE \text{ such that} \\
&\quad L(v_{min}, e_{min}) = m; \\
&\quad TE = TE \setminus \{(v_{min}, e_{min})\}; \\
&\quad SE = SE \cup \{(v_{min}, e_{min})\}; \\
&\quad \text{if } (v_{min} \notin S) \\
&\quad \quad \text{le}(v_{min}) = e_{min}; S = S \cup \{v_{min}\}; \\
&\quad \quad l(v_{min}) = L(v_{min}, e_{min}) ; T = T \setminus \{v_{min}\}; \\
&\quad \text{if } (t_{i,j} \not< v_{min}) \\
&\quad \quad \text{for } (v,e) \in TE \text{ adjacent after } (v_{min}, e_{min}) \\
&\quad \quad \quad \text{if } (v_{min}==s_{i,j}) \\
&\quad \quad \quad \quad L'(v,e) = L(s_{i,j}, \emptyset) + we(v_{min}, v); \\
&\quad \quad \quad \text{else} \\
&\quad \quad \quad \quad L'(v,e) = L(v_{min}, e_{min}) + we(v_{min}, v) + wv(v_{min}, e_{min}, e); \\
&\quad \quad \quad \text{if } (L(v,e) > L'(v,e)) \\
&\quad \quad \quad \quad L(v,e) = L'(v,e); \\
&\quad \quad \quad P(v,e) = (v_{min}, e_{min}); \\
&\text{while (m < l_{min} or t_{i,j} < v_{min})} \\
&\quad \text{if } (L(t_{i,j}, e(t_{i,j})) < l_{min}) \text{ and } (t_{i,j} = v_{min}) \text{ //edge to } (i,j) \\
&\quad \quad \quad \text{imin} = i; j_{min} = j; l_{min} = L(t_{i,j}, e(t_{i,j})); \\
&\quad \quad \quad w_{min} = le(t_{i,j}); // \text{ edge to } t_{i,j} \\
&\quad \quad \quad \text{for } (v,e) \in VE: Pmin(v,e) = P(v,e); \\
&\quad \text{ }// end for...for \\
&\quad \text{ // find the shortest path} \\
&\quad \text{if } (l_{min} < +\infty) \\
&\quad \quad \text{// Moves from } t_{i_min,j_min}, \text{ in reverse direction, to the} \\
&\quad \quad \text{preceding vertex-edges, we get the shortest path as} \\
&\quad \quad \text{follows:} \\
&\quad \quad k=1; (v_k, e_k) = Pmin(t_{i_min,j_min}, e_{min}); \\
&\quad \quad \text{while } ((v_k, e_k) <\rangle (s_{i_min,j_min}, \emptyset)) \\
&\quad \quad \quad \{ \\
&\quad \quad \quad \quad k=k+1; (v_k, e_k) = P(v_{k-1}, e_{k-1}); \\
&\quad \quad \text{ }\\
&\quad \text{// Deduced the shortest path is} \\
&\quad s_{i_min,j_min} \rightarrow v_k \rightarrow v_{k-1} \rightarrow \ldots \rightarrow v_1 \rightarrow t_{i_min,j_min} \\
&\text{// End}\n\end{align*}
\]

**Theorem 3.1.** The algorithm that finds the smallest shortest path between the pairs of vertices on the multiple-weighted extended graph is correct and has an algorithmic complexity \(O(k \cdot n^3)\), where \(n\) is the number of vertices and \(k=k_1+\ldots+k_r\).

**Proof.** The correctness of the algorithm derives from theorem 2.1. The algorithm that finds the shortest path between the source-destination vertices has the complexity \(O(n^3)\), which inferred the algorithm finding the smallest shortest path between the \(k\) of the destination source pair has complexity \(O(k \cdot n^3)\).

**4. The problem of maximum flow on extended Linear multicommodity multicommodity network**

The model of multicommodity multicommodity network was built in the article [12].

Given a multicommodity multicommodity extended graph \(G=(V, E, ce, ze, cv, zv, \{be_{ii}, bv_{ii}, q_i \ | i=1..r\})\). Assume for each commodity \(i, i=1..r\), with \(k_i\) source-destination pairs \((s_{i,j}, t_{i,j})\), \(j=1..k_i\), each of pair assigned a quantity of commodity type \(i\), which needs to be transferred from source node \(s_{i,j}\) to destination node \(t_{i,j}\).

The problem is to find the multicommodity flow such that the flow value is maximal.
Algorithm

Input: Given a multicommodity multicut extended network G=(V,E, ce, ze, cv, zv, {be, bv, qi | i=1..r}). Assume for each of commodity type i, j=1..r, there are ki source-destination pairs (sij, tij), j=1..ki, each pair assigned a quantity of commodity type i which needs to be transferred from source node sij to destination node tij.

ω is the approximation to be achieved.

Output: Maximum flow F represents the set of converged streams at the edges

F = \{x_i(e) | e \in E, i=1..r, j=1..k_i \}

Procedure

// The symbol n=|V|, m=|E|. Calculate ε and δ

ε = 1 - \sqrt{1/(1+ω)} ;

δ = \left(1+\varepsilon^2\right)^{\frac{1}{2}} ;

for e \in E : le(e)=δ, x_i(e)=0 ;

for v \in V : hv(v)=δ ;

do

Using the algorithm to find the source-destination pair (sij, tij), 1≤i≤r and 1≤j≤k_i, with the smallest shortest path from sij to tij with edge weight le(e), \forall e \in E, and switch weights at nodes are hv(v), \forall v \in V.

Symbol

imin and jmin are index pairs of the source-destination nodes has the shortest path.

α is the shortest path length;

p is the shortest path;

c is the smallest capacity of passing edges and vertex of p:

c=\min\{\min\{ce(e),ze(e)\}\in p\},\min\{cv(v),zv(v)\}\in p\};

// Adjust the flow:

∀e ∈ p, x_{imin,p}(e)=x_{imin,p}(e)+c; fv=fv+c ;

le(e)= le(e),(1+εc)(ce(e),ze(e));

∀v \in p, hv(v)= hv(v),(1+εc)(cv(v),zv(v));

while (α < 1)

// Calculating the value resulted from flow F and value of flow fv.

x_i(e)=x_i(e) / \log_{1+ε} \delta , \forall i=1..r, j=1..k_i, \forall e \in E ;

fv = fv / \log_{1+ε} \delta ;

// Calculating the flow on the undirected edge

for (i=1 ; i<r ; i++)

for (j=1 ; j<k; j++)

for e\in E, e\ undirected

if x_i(e)≥x_i(e')/ε' is the opposite edge e

\{ x_i(e)=x_i(e')−x_i(e') ;

Table 1. Node flow capability

| Nodes | cv |
|-------|----|
| 1     | 100 |
| 2     | 100 |
| 3     | 50  |
| 4     | 100 |
| 5     | 50  |
| 6     | 100 |

Table 2. Commodity converting coefficient

| Commodity | q |
|-----------|---|
| 1         | 1 |
| 2         | 2 |
| 3         | 3 |
Applying algorithm finding shortest path in the multiple-weighted graphs to find maximal flow in extended linear multicommodity multicost network

Table 3: Pairs of source-target nodes

| Commodity | \( s_{ij} \) | \( t_{ij} \) |
|-----------|-------------|-------------|
| 1         | 1           | 5           |
| 2         | 2           | 4           |
| 3         | 3           | 6           |

Table 4: Edge capability and cost
Notes: Type 1 is directional, type 0 is undirectional.

| Edge   | Type | \( ce \) | \( be_1 \) | \( be_2 \) | \( be_3 \) |
|--------|------|----------|----------|----------|----------|
| (1,2)  | 1    | 50       | 4        | 5        | 6        |
| (1,3)  | 1    | 50       | 4        | 5        | 6        |
| (2,3)  | 0    | 70       | 4        | 5        | 6        |
| (3,2)  | 0    | 70       | 3        | 4        | 5        |
| (2,5)  | 1    | 50       | \( \infty \) | 5        | 6        |
| (3,4)  | 1    | 50       | 4        | 5        | 6        |
| (3,5)  | 0    | 70       | 4        | 5        | \( \infty \) |
| (5,3)  | 0    | 70       | 3        | 4        | \( \infty \) |
| (4,6)  | 1    | 50       | 4        | \( \infty \) | 6        |
| (4,5)  | 0    | 70       | 3        | 5        | \( \infty \) |
| (5,4)  | 1    | 50       | 4        | 5        | 6        |

Table 5: Switch cost

| Node | Edge 1 | Edge 2 | \( bv_1 \) | \( bv_2 \) | \( bv_3 \) |
|------|--------|--------|----------|----------|----------|
| 2    | (1,2)  | (2,3)  | 1        | 2        | 3        |
| 2    | (1,2)  | (2,5)  | 1        | 2        | 3        |
| 2    | (3,2)  | (2,5)  | 1        | 2        | 3        |
| 3    | (1,3)  | (3,4)  | 1        | 2        | 3        |
| 3    | (1,3)  | (3,5)  | 1        | \( \infty \) | \( \infty \) |
| 3    | (1,3)  | (3,2)  | 1        | 2        | 3        |
| 3    | (5,3)  | (3,2)  | 1        | 2        | 3        |
| 3    | (3,4)  | (4,6)  | 1        | 2        | 3        |
| 3    | (3,4)  | (4,5)  | 1        | 2        | 3        |
| 4    | (3,4)  | (4,6)  | 1        | 2        | 3        |
| 5    | (2,5)  | (5,3)  | 1        | \( \infty \) | \( \infty \) |
| 5    | (2,5)  | (5,4)  | 1        | \( \infty \) | \( \infty \) |
| 5    | (2,5)  | (5,6)  | 1        | 2        | 3        |
| 5    | (3,5)  | (5,4)  | 1        | 2        | 3        |
| 5    | (3,5)  | (5,6)  | 1        | 2        | 3        |
| 5    | (4,5)  | (5,3)  | 1        | 2        | 3        |
| 5    | (4,5)  | (5,6)  | 1        | 2        | 3        |

The algorithm is coded in C++ and gives correct results. Below is the result of the above example.

Coefficient of approximation: 0.070000
Total output: 148.908624
Total cost: 1877.662162

Flow for commodity type 1, which needs to be transferred from source node 1 to destination node 5
1 2 8.396823
1 3 41.500042
2 3 8.396823
3 5 49.896862

Flow for commodity type 2, which needs to be transferred from source node 2 to destination node 4
2 3 0.093091
3 4 0.093091

Flow for commodity type 3, which needs to be transferred from source node 3 to destination node 6
3 2 49.375553
2 5 49.375553
3 4 49.543118
4 6 49.543118
5 6 49.375553

6. Conclusion

The article develops the algorithm finding the shortest path in extended graphs (Section 2), the algorithm finding the shortest path on the multiple-weighted extended graph (Section 3). Based on the duality theory of linear programming, an approximation algorithm with polynomial complexity is developed on the base of the algorithm finding shortest paths in section 2 and 3. This is also the main result of the article. Correctness and algorithm complexity are justified and the algorithm is stored in C++ and given an exact result. The results of this article are the basis for studying the applications of multicomodity multicost flow optimization.

7. References

[1] Naveen Garg, Jochen Könemann: Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems, SIAM J. Comput, Canada, 37 (2), 2007, pp. 630-652.
[2] Xiaolong Ma, Jie Zhou: An Extended Shortest Path Problem with Switch Between Arcs, Proceedings of the International Conference on Engineers and Computer Scientists 2008 Vol IIMECS 2008, 19-21 March, 2008, Hong Kong.
[3] Tran Quoc Chien: Linear multitransport network calculation, ministerial scientific research, code B2010DN-03-52.

[4] Tran Quoc Chien, Tran Thi My Dung: Applying of algorithm finding the shortest path for maximum multicommodity flow. Journal of Science & Technology, University of Danang, 3 (44) 2011.

[5] Tran Quoc Chien: Applying of algorithm finding the shortest path for maximum simultaneous multicommodity flow. Journal of Science & Technology, University of Danang, 4 (53) 2012.

[6] Tran Quoc Chien: Applying of algorithm finding the shortest path for minimum cost maximum simultaneous multicommodity flow. Journal of Science & Technology, University of Danang, 5 (54) 2012.

[7] Tran Quoc Chien: The algorithm finding the shortest path in the general graph, Journal of Science & Technology, University of Da Nang, 12 (61) / 2012, 16-21.

[8] Tran Quoc Chien, Nguyen Mau Tue, Tran Ngoc Viet: The algorithm finding the shortest path on the extension graph. Proceeding of the 6th National Conference on Fundamental and Applied Information Technology (FAIR), Proceedings of the Sixth National Conference on Scientific Research and Applying, Hue, 20-21 June 2013. Natural Science and Technology Publishing House. Hanoi 2013. p.522-527.

[9] Tran Quoc Chien: Applying of algorithm finding the fastest path for maximum multi-transport linearity of minimum cost on extension transportation networks, Journal of Science & Technology, University of Da Nang . 10 (71) 2013, 85-91.

[10] Tran Ngoc Viet, Tran Quoc Chien, Nguyen Mau Tue: Optimized Linear Multiplexing Algorithm on Extension Transportation Networks, Journal of Science & Technology, University of Da Nang. 3 (76) 2014, 121-124.

[11] Tran Ngoc Viet, Tran Quoc Chien, Nguyen Mau Tue: The calculation of linear multitransport on transportation network. Proceedings of the 7th National Conference on Fundamental and Applied Information Technology Research (FAIR7), ISBN: 978-604-913-300-8, Proceedings of the 7th National Scientific Conference on "Fundamental and Applied Research IT", Thai Nguyen, 19-20 / 6/2014. NXB Natural Science and Technology. Hanoi 2014. p.31-39.

[12] Tran Quoc Chien, Ho Van Hung: The multicommodity multi flow expansion network and the algorithm used to find the maximum flow FAIR-2017, August, 2017.