

# Implement and Test Algorithm Finding Maximal Flow Limited Cost in Extended Multi commodity Multicost Network

Ho Van Hung<sup>1</sup> and Tran Quoc Chien<sup>2</sup>

<sup>1</sup> Quangnam University, Tamky, Vietnam

<sup>2</sup>The University of Education, University of Danang, Danang, Vietnam

## Abstract:

Graph is a powerful mathematical tool applied in many fields as transportation, communication, informatics, economy,... In ordinary graph the weights of edges and vertexes are considered independently where the length of a path is the sum of weights of the edges and the vertexes on this path. However, in many practical problems, weights at a vertex are not the same for all paths passing this vertex, but depend on coming and leaving edges. Furthermore, on a network, many commodities share capacities of edges and vertexes with different costs. So it is necessary to study network with multiple weights. The presented paper introduces the algorithm to find the shortest path between two vertices on extended graph with multiple weights. Then, the shortest path finding algorithms is used to implement the general algorithm finding the maximum flow limited cost on the extended multicommodity multicost network developed in the article [14].

**Keywords:** Graph, Network, Multicommodity Multicost Flow, Optimization, Linear Programming.

## 1. Introduction

Network and its flow is a powerful mathematical tool applied in many fields as transportation, communications, informatics, economics, and so on. So far, most of applications in graphs just solely consider to the weights of edges and nodes independently, in which the path length merely is the sum of weights of edges and nodes along the path. However, in many practical problems, the weight at a node is not the same for all paths passing through that node, but also depends on coming and leaving edges. For example, the transit time on the transport network depends on the direction of transportation: turn right, turn left or go straight, even some directions are forbidden. Paper [2] propose switching cost only for directed graphs. Multicommodity flow in traditional network problems have been studied in the papers [1,3,4,5,6]. Multicommodity flow in extended network problems with extended transport networks were studied in the papers [7-11]. Furthermore, on a network, many commodities share capacities of edges and vertexes with different costs. So it is necessary to study network with multiple weights. The paper [12] and [13] study maximal flow problems and the paper [14] studies maximal flow limited cost problems on extended multicommodity multicost networks.

The presented paper introduces the algorithm to find the shortest path between two vertices on extended graph with multiple weights. Then, the shortest path finding algorithms is used to implement the general algorithm finding the maximum flow limited cost on the extended multicommodity multicost network developed in the

article [14]. The content of the paper is as follows. The maximal flow limited cost problems on extended multicommodity multicost networks is introduced in section 2. In section 3, the shortest path finding algorithms is used to implement the general algorithm finding the maximum flow limited cost on the extended multicommodity multicost network developed in the article [14]. The algorithm is coded in the programming language C and tested in section 4.

## 2. Maximal Flow Limited Cost Problems In Extended Multicommodity Multicost Network

Given mixed graph  $G = (V, E)$  with node set  $V$  and edge set  $E$ . The edges may be undirected or directed. The symbol  $E_v$  is the set of edges incident vertex  $v \in V$ . There are many kinds of goods circulating on the network. Commodities share the capacities of the edges, but have different costs. The undirected edges represent the two-way edge, in which the goods on the same edge, but reverse directions share the capacity of the edge.

The symbol  $r$  is the commodity number,  $q_i > 0$  is the coefficient of conversion of commodity  $i$ ,  $i = 1..r$ .

Given the following functions:

*Edge passing capacity function*  $ce: E \rightarrow R^*$ , where  $ce(e)$  is the passing capability of the edge  $e \in E$ .

*Edge service coefficient function*  $ze: E \rightarrow R^*$ , where  $ze(e)$  is the passing ratio of the edge  $e \in E$  (the real capacity of the edge  $e$  is  $ze(e).ce(e)$ ).

*Node passing capability function*  $cv: V \rightarrow R^*$ , where  $cv(u)$  is the passing capability of the node  $u \in V$ .

Node service coefficient function  $z_v:V \rightarrow R^*$ , where  $z_v(u)$  is the passing ratio of the node  $v \in V$  (the real capacity of the node  $v$  is  $z_v(v).cv(v)$ ).

The tuples  $(V, E, ce, ze, cv, z_v)$  are called *extended networks*.

Edge cost function  $i, i=1..r, be_i: E \rightarrow R^*$ , where  $be_i(e)$  is the cost of passing  $e$  a converted unit of commodity of type  $i$ . Note that with 2-way paths, the costs of each way may vary.

Node switch cost function  $i, i=1..r, bv_i: V \times E_v \times E_v \rightarrow R^*$ , where  $bv_i(u, e, e')$  is the cost of transferring a converted unit of commodity of type  $i$  from edge  $e$  through  $u$  to edge  $e'$ .

The sets  $((V, E, ce, ze, cv, z_v, \{be_i, bv_i, q_i | i=1..r\}))$  are called *the extended linear multicommodity multicommodity network*.

◇Note:

If  $be_i(e)=\infty$ , commodity of type  $i$  is prohibited from circulation on path  $e$ . If  $bv_i(u, e, e') = \infty$ , commodity of type  $i$  is banned from path  $e$  through  $u$  to path  $e'$ .

Let  $p$  be the path from node  $u$  to node  $v$  through edges  $e_j, j=1..(h+1)$ , and nodes  $u_j, j=1..h$  has follows

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

The cost of circulating a converted unit of commodity of type  $i, i=1..r$ , passing the path  $p$ , is denoted by the symbol  $b_i(p)$ , and defined by the following formula:

$$b_i(p) = \sum_{j=1}^{h+1} be_i(e_j) + \sum_{j=1}^h bv_i(u_j, e_j, e_{j+1}) \quad (2)$$

Given a multicommodity network  $G=(V, E, ce, ze, cv, z_v, \{be_i, bv_i, q_i | i=1..r\})$ . Assume, for each commodity of type  $i, i=1..r$ , there are  $k_i$  source-target pairs  $(s_{i,j}, t_{i,j}), j=1..k_i$ , each pair assigned a quantity of commodity of type  $i$ , that is necessary to move from source node  $s_{i,j}$  to target node  $t_{i,j}$ .

Denote  $P_{i,j}$  is the set of paths from node  $s_{i,j}$  to node  $t_{i,j}$  in  $G$ , which commodity of type  $i$  can be passed,  $i=1..r, j=1..k_i$ . Set

$$P_i = \bigcup_{j=1}^{k_i} P_{i,j} \quad (3)$$

For each path  $p \in P_{i,j}, i=1..r, j=1..k_i$ , denote  $x_{i,j}(p)$  the flow of converted commodity of type  $i$  from the source nodes  $s_{i,j}$  to the destination nodes  $t_{i,j}$  along the path  $p, i=1..r, j=1..k_i$ .

Denote  $P_{i,e}$  the set of paths in  $P_i$  passing through the edge  $e, \forall e \in E$ .

Denote  $P_{i,v}$  the set of paths in  $P_i$  passing through the node  $v, \forall v \in V$ .

A set

$$F = \{x_{i,j}(p) | p \in P_{i,j}, i=1..r, j=1..k_i\} \quad (4)$$

is called *a multicommodity flow* on the linear extended multicommodity multicommodity network, if it satisfies the following *edge and node capacity* constraints:

$$\sum_{i=1}^r \sum_{j=1}^{k_i} \sum_{p \in P_{i,e}} x_{i,j}(p) \leq ce(e).ze(e), \forall e \in E \quad (5)$$

$$\sum_{i=1}^r \sum_{j=1}^{k_i} \sum_{p \in P_{i,v}} x_{i,j}(p) \leq cv(v).z_v(v), \forall v \in V \quad (6)$$

The expressions

$$fv_{i,j} = \sum_{p \in P_{i,j}} x_{i,j}(p), i=1..r, j=1..k_i \quad (7)$$

is called *the flow value of commodity of type  $i$  of the source-target pair  $(s_{i,j}, t_{i,j})$  of  $F$* .

The expressions

$$fv_i = \sum_{j=1}^{k_i} fv_{i,j}, i=1..r \quad (8)$$

is called *the flow value of commodity of type  $i$  of  $F$* .

The expression

$$fv = \sum_{i=1}^r fv_i \quad (9)$$

is called *the flow value of  $F$* .

Given an extended linear multicommodity multicommodity network  $G=(V, E, ce, ze, cv, z_v, \{be_i, bv_i, q_i | i=1..r\})$ . Assume, for each commodity of type  $i, i=1..r$ , there are  $k_i$  source-target pairs  $(s_{i,j}, t_{i,j}), j=1..k_i$ , each pair assigned a quantity of commodity of type  $i$ , that is necessary to move from source node  $s_{i,j}$  to target node  $t_{i,j}$ . Given a limit cost  $B$ .

The task of the problem is to find the multicommodity flow such that the value of the flow  $fv$  is maximal. At the same time, the total cost of the flow does not exceed  $B$ .

The problem is expressed by the an implicit linear programming model  $(P)$  as follows:

$$\left. \begin{aligned} &fv = \sum_{i=1}^r \sum_{j=1}^{k_i} \sum_{p \in P_{i,j}} x_{i,j}(p) \rightarrow \max \\ &\text{satisfies} \\ &\sum_{i=1}^r \sum_{j=1}^{k_i} \sum_{p \in P_{i,e}} x_{i,j}(p) \leq ce(e).ze(e), \forall e \in E \\ &\sum_{i=1}^r \sum_{j=1}^{k_i} \sum_{p \in P_{i,v}} x_{i,j}(p) \leq cv(v).z_v(v), \forall v \in V \\ &(P) \\ &\sum_{i=1}^r \sum_{j=1}^{k_i} \sum_{p \in P_{i,j}} x_{i,j}(p).b_i(p) \leq B \\ &x_{i,j}(p) \geq 0, \forall i=1..r, j=1..k_i, \forall p \in P_{i,j} \end{aligned} \right\}$$

### 3. Implementing of Algorithm

The general algorithm with polynomial complexity is proved in [14]. In this paper we integrate the algorithm finding shortest path [7,8] in order to implement the mentioned algorithm [14].

### 3.1. The problem of finding the shortest path in extended graph

Given extended mixed 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  $we(e)$ . 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  $wv(v, e, e')$ .

The sets  $(V, E, we, wv)$  are called *extended graph*.

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

$$p = [u, e_1, u_1, e_2, u_2, \dots, e_h, u_h, e_{h+1}, v] \quad (10)$$

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

$$l(p) = \sum_{j=1}^{h+1} we(e_j) + \sum_{j=1}^h wv(u_j, e_j, e_{j+1}) \quad (11)$$

#### •The problem of finding the shortest path

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

The following function find the shortest path  $p$  from  $s$  to  $t$  and return the length of  $p$ . This function 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  $v$ ;

$le(v)$  is the edge that leads to the vertex  $v$  on the shortest path from  $s$  to  $v$ ;

$VE = \{(v, e) \mid v \in V - \{s\} \ \& \ e \in E_v\} \cup \{(s, \emptyset)\}$  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  $(v, e) \in VE$

$P(v, e)$  is the vertex-edge adjacent before  $(v, e) \in VE$ .

```
float ShortestPath(V, E, we, wv, s, t, p)
{
    // Initialization
    S = ∅ ; T = V ;
    VE = { (v, e) | v ∈ V - {s} & e ∈ E_v } ∪ { (s, ∅) };
    SE = ∅ ; TE = VE;
```

```
L(v, e) = +∞; ∀ (v, e) ∈ VE, L(s, ∅) = 0;
for (v, e) ∈ VE: P(v, e) = ∅;
// main algorithm body
do
{
    m = min{L(v, e) | (v, e) ∈ TE}.
    if (m < +∞)
    {
        Choose (v_min, e_min) ∈ TE such that
        L(v_min, e_min) == m;
        TE = TE - { (v_min, e_min) };
        SE = SE ∪ { (v_min, e_min) };
        if (v_min ∉ S)
        {
            le(v_min) = e_min;
            S = S ∪ {v_min};
            l(v_min) = L(v_min, e_min);
            T = T - {v_min};
        }
        if (t <> v_min)
        {
            for (v, e) ∈ TE adjacent
            after (v_min, e_min)
            {
                if (v_min == s)
                L'(v, e) = L(s, ∅)
                + we(v_min, v);
                else
                L'(v, e) = L(v_min, e_min) + w
                e(v_min, v) + wv(v_min, e_min,
                e);
                If (L(v, e) > L'(v, e))
                {
                    L(v, e) = L'(v, e);
                    P(v, e) = (v_min, e_min);
                }
            }
        }
    }
} while (m < +∞ or t <> v_min)
if (m == +∞) return +∞; // no
path exists from s to t
// finding the shortest path
k=1; (v_k, e_k) = P(t, le(t));
while ((v_k, e_k) <> (s, ∅))
{
    k=k+1; (v_k, e_k) = P(v_{k-1}, e_{k-1});
}
p = s → v_k → v_{k-1} → ... → v_1 → t
return L(t, le(t)); // shortest
path length from s to t.
} // end of ShortestPath
```

### 3.2. Algorithm of finding maximal flow limited cost

◇**Input:** Extended multicommodity network  $G=(V,E, ce, ze, cv, zv, \{be_i, bv_i, q_i | i=1..r\})$ ,  $n=|V|$ ,  $m=|E|$ . Assume, for each commodity of type  $i$ ,  $i=1..r$ , there are  $k_i$  source-target pairs  $(s_{i,j}, t_{i,j})$ ,  $j=1..k_i$ , each pair assigned a quantity of commodity of type  $i$ , that is necessary to move from source node  $s_{i,j}$  to target node  $t_{i,j}$ . Given a limited cost  $B$ , an approximation ratio  $\omega$ .

◇**Output:** Maximal flow  $F$  represents a set of converged flows at the edges

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

with total cost  $B_f$  not over the limit cost  $B$ .

◇**Procedure**

// Calculate  $b_{min}$ , the smallest cost in the paths from the source  $s_{i,j}$  to the destination  $t_{i,j}$ .

For  $(i=1..r, j=1..k_i) b_{min} = \min\{b_i(p) | i=1..r,$

$j=1..k_i, p \in P_{i,j}\}$ .

$b_{min} = \infty;$

for  $(i=1; i \leq r; i++)$

for  $(j=1; j \leq k_i; j++)$

{

$l = \text{ShortestPath}(V, E, be_i, bv_i, s_{i,j}, t_{i,j}, p)$

if  $(b_{min} > l)$

$b_{min} = l;$

}

// Choose  $b_{max} \geq \max\{b_i(p) | i=1..r, j=1..k_i, p \in P_{i,j}\}$ .

$c1 = 0;$

for  $(i=1; i \leq r; i++)$

for  $(e \in E)$

if  $(be_i(e) > c1)$

$c1 = be_i(e); c2 = 0;$

for  $(i=1; i \leq r; i++)$

for  $(v \in E)$

for  $(e, e') \in E_v \times E_v$

if  $(bv_i(v, e, e') > c2)$

$c2 = bv_i(v, e, e');$

$b_{max} = n * (c1 + c2);$

// Choose  $\varepsilon, \delta$  and initialization

$\varepsilon = 1 - \sqrt[1/\omega]{1 + \omega};$

$\delta = (1 + \varepsilon)$

$1$

$$\left[ (1 + \varepsilon)^2 (n + n + b_{max} / b_{min}) \right]^{1/\varepsilon};$$

for  $(e \in E) l_e(e) = \delta;$

for  $(v \in V) l_v(v) = \delta;$

$\varphi = \delta / b_{min}; f_v = 0;$

$B_f = 0;$

for  $(i=1; i \leq r; i++)$

for  $(j=1; j \leq k_i; j++)$

for  $(e \in E)$

$x_{i,j}(e) = 0;$

// Note

$i_{min}$  and  $j_{min}$  the indexes of source-destination pairs with the shortest path;

$\alpha$  the shortest path length;

$p$  the shortest path;

$c$  the minimal edge and node capacity on the path  $p$ , i.e.

// main algorithm body

do

{

$\alpha = \infty; i_{min} = 1; j_{min} = 1; p = \emptyset;$

for  $(i=1; i \leq r; i++)$

for  $(j=1; j \leq k_i; j++)$

{

$l =$

$\text{ShortestPath}(V, E, l_e + \varphi.be_i, l_v + \varphi.bv_i,$

$s_{i,j}, t_{i,j}, p_{temp})$

if  $(\alpha > l)$

{

$\alpha = l; i_{min} = i;$

$j_{min} = j;$

$p = p_{temp};$

}

}

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

$B' = c.b_{i_{min}}(p);$

if  $(B' > B) \{ c = c.B / B'; B' = B;$

}

// Flow adjustments:

for  $(e \in$

$p) \{ x_{i,j}(e) = x_{i,j}(e) + c; l_e(e) = l_e(e) \cdot (1 + \varepsilon.c / (ce(e).ze(e))); \}$

for

$(v \in p) l_v(v) = l_v(v) \cdot (1 + \varepsilon.c / (cv(v).zv(v)))$

;

$\varphi = \varphi \cdot (1 + \varepsilon.B' / B);$

$B_f = B_f + B';$

$f_v = f_v + c;$

} while  $(\alpha < 1)$

// Modifying the resulting flows  $F$  and flow value  $f_v$ .

for  $(i=1; i \leq r; i++)$

for  $(j=1; j \leq k_i; j++)$

for  $(e \in E)$

$x_{i,j}(e) = x_{i,j}(e) / (1 - \log_{1+\varepsilon}\delta);$

$f_v = f_v / (1 - \log_{1+\varepsilon}\delta);$

$B_f = B_f / (1 - \log_{1+\varepsilon}\delta);$

// Modifying flows on scalar edge

for  $(i=1; i \leq r; i++)$

for  $(j=1; j \leq k_i; j++)$

for  $(e \in E \&\& e \text{ scalar})$

if  $x_{i,j}(e) \geq x_{i,j}(e') // e'$  is the opposite of the direction  $e$

{

$x_{i,j}(e) = x_{i,j}(e) - x_{i,j}(e');$

$x_{i,j}(e') = 0$

};

else

$\{ x_{i,j}(e') = x_{i,j}(e') - x_{i,j}(e);$

$x_{i,j}(e) = 0 \};$

/\*\* end of algorithm \*/

4. TEST

4.1. Example

The following example is inspired by the beautiful DaNang City of VietNam, where the world leaders were welcomed to take part in the APEC 2017. The traffic network is in Figure 1. The database contains the table 1, table 2, table 3, table 4 and table 5.

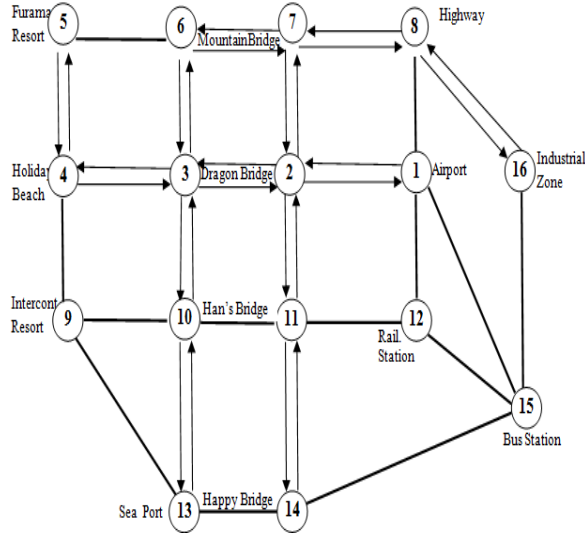


Figure 1. DaNang City

Table 1. Node flow capability

Nodes	$cv$	$zv$
1	1000	0.7
2	1000	0.8
3	1000	0.8
4	500	0.9
5	500	0.9
6	1000	0.8
7	1000	0.8
8	1500	0.8
9	500	0.9
10	1000	0.8
11	500	0.8
12	1000	0.7
13	1500	0.7
14	1000	0.8
15	1200	0.8
16	1500	0.7

Table 2. Commodity converting coefficient

Commodity	Vehicle	$q$
1	Motor car	1
2	Light truck	5

Table 4: Edge capacity and cost

No	Edge	Type	$ce$	$ze$	$be_1$	$be_2$	$be_3$	$be_4$
1	(1,2)	1	500	0.9	3	4	$\infty$	$\infty$
2	(2,1)	1	500	0.9	3	4	$\infty$	$\infty$
3	(2,3)	1	500	0.9	3	4	$\infty$	$\infty$
4	(3,2)	1	500	0.9	3	4	$\infty$	$\infty$
5	(3,4)	1	500	0.9	3	4	$\infty$	$\infty$
6	(4,3)	1	500	0.9	3	4	$\infty$	$\infty$
7	(4,5)	1	500	0.9	3	4	$\infty$	$\infty$
8	(5,4)	1	500	0.9	3	4	$\infty$	$\infty$
9	(5,6)	0	700	0.8	3	4	$\infty$	$\infty$
10	(6,5)	0	700	0.8	3	4	$\infty$	$\infty$
11	(6,7)	1	700	0.9	3	4	6	8
12	(7,6)	1	700	0.9	3	4	6	8
13	(7,8)	1	700	0.9	3	4	6	8
14	(8,7)	1	700	0.9	3	4	6	8
15	(8,16)	1	700	0.9	3	4	6	8
16	(16,8)	1	700	0.9	3	4	6	8
17	(3,6)	1	700	0.9	3	4	6	8
18	(6,3)	1	700	0.9	3	4	6	8
19	(2,7)	1	500	0.9	3	4	$\infty$	$\infty$
20	(7,2)	1	500	0.9	3	4	$\infty$	$\infty$
21	(1,8)	0	800	0.8	3	4	6	$\infty$
22	(8,1)	0	800	0.8	3	4	6	$\infty$
23	(4,9)	0	600	0.8	4	5	$\infty$	$\infty$
24	(9,4)	0	600	0.8	3	4	$\infty$	$\infty$
25	(10,9)	0	700	0.8	4	5	$\infty$	$\infty$
26	(9,10)	0	700	0.8	3	4	$\infty$	$\infty$
27	(9,13)	0	500	0.8	3	4	$\infty$	$\infty$
28	(13,9)	0	500	0.8	4	5	$\infty$	$\infty$
29	(3,10)	1	700	0.9	3	4.5	6	8
30	(10,3)	1	700	0.9	3	4.5	6	8

31	(13,10)	1	700	0.9	3	4.5	6	8
32	(10,13)	1	700	0.9	3	4.5	6	8
33	(10,11)	0	500	0.8	3	4	∞	∞
34	(11,10)	0	500	0.8	3	4	∞	∞
35	(2,11)	1	500	0.9	3	4	∞	∞
36	(11,2)	1	500	0.9	3	4	∞	∞
37	(11,14)	1	500	0.9	3	4	∞	∞
38	(14,11)	1	500	0.9	3	4	∞	∞
39	(11,12)	0	600	0.8	3	4	∞	∞
40	(12,11)	0	600	0.8	3	4	∞	∞
41	(1,12)	0	800	0.8	3	4	6	∞
42	(12,1)	0	800	0.8	3	4	6	∞
43	(12,15)	0	800	0.8	3	4	6	∞
44	(15,12)	0	800	0.8	3	4	6	∞
45	(1,15)	0	800	0.8	4	6	8	∞
46	(15,1)	0	800	0.8	4	6	8	∞
47	(15,16)	0	800	0.8	4	6	8	∞
48	(16,15)	0	800	0.8	4	6	8	∞
49	(13,14)	0	500	0.8	4	5	∞	∞
50	(14,13)	0	500	0.8	4	5	∞	∞
51	(14,15)	0	800	0.8	4	6	8	∞
52	(15,14)	0	800	0.8	4	6	8	∞

Notes: Type 1 is directional, type 0 is undirectional.

**Table 5. Switch cost**

No	Node	Edge 1	Edge 2	$bv_1$	$bv_2$	$bv_3$	$bv_4$
1	1	(2,1)	(1,8)	1.5	2.5	∞	∞
2	1	(2,1)	(1,12)	1	2	∞	∞
3	1	(2,1)	(1,15)	1	2	∞	∞
4	1	(8,1)	(1,2)	1.5	2	∞	∞
5	1	(8,1)	(1,12)	2	2.5	3	∞
6	1	(8,1)	(1,15)	2	2.5	3	∞
7	1	(12,1)	(1,2)	2	3	∞	∞
8	1	(12,1)	(1,8)	1.5	2	3	∞
9	1	(15,1)	(1,2)	2	3	∞	∞
10	1	(15,1)	(1,8)	1.5	2	3	∞
11	2	(1,2)	(2,7)	1	2	∞	∞
12	2	(1,2)	(2,3)	1.5	2.5	∞	∞
13	2	(1,2)	(2,11)	2	3	∞	∞
14	2	(7,2)	(2,3)	1	2	∞	∞
15	2	(7,2)	(2,1)	1.5	2.5	∞	∞
16	2	(7,2)	(2,11)	2	3	∞	∞
17	2	(3,2)	(2,11)	1	2	∞	∞
18	2	(3,2)	(2,1)	1.5	2.5	∞	∞
19	2	(3,2)	(2,7)	2	3	∞	∞
20	2	(11,2)	(2,1)	1	2	∞	∞
21	2	(11,2)	(2,7)	1.5	2.5	∞	∞
22	2	(11,2)	(2,3)	2	3	∞	∞
23	3	(2,3)	(3,6)	1	2	∞	∞
24	3	(2,3)	(3,4)	1.5	2.5	∞	∞
25	3	(2,3)	(3,10)	2	3	∞	∞
26	3	(4,3)	(3,10)	1	2	∞	∞
27	3	(4,3)	(3,2)	1.5	2.5	∞	∞
28	3	(4,3)	(3,6)	2	3	∞	∞

29	3	(6,3)	(3,4)	1	2	∞	∞
30	3	(6,3)	(3,10)	1.5	2.5	3	4
31	3	(6,3)	(3,2)	2	3	∞	∞
32	3	(10,3)	(3,2)	1	2	∞	∞
33	3	(10,3)	(3,6)	1.5	2.5	3	4
34	3	(10,3)	(3,4)	2	3	∞	∞
35	4	(3,4)	(4,5)	1	2	∞	∞
36	4	(3,4)	(4,9)	1.5	2.5	∞	∞
37	4	(5,4)	(4,9)	1	2	∞	∞
38	4	(5,4)	(4,3)	1.5	2.5	∞	∞
39	4	(9,4)	(4,3)	1	2	∞	∞
40	4	(9,4)	(4,5)	1.5	2.5	∞	∞
41	5	(4,5)	(5,6)	1	2	∞	∞
42	5	(6,5)	(5,4)	1.5	2.5	∞	∞
43	6	(5,6)	(6,3)	1	2	∞	∞
44	6	(5,6)	(6,7)	1.5	2.5	∞	∞
45	6	(3,6)	(6,5)	1	2	∞	∞
46	6	(3,6)	(6,7)	1.5	2.5	3	4
47	6	(7,6)	(6,5)	1	2	∞	∞
48	6	(7,6)	(6,3)	1.5	2.5	3	4
49	7	(2,7)	(7,8)	1	2	∞	∞
50	7	(2,7)	(7,6)	1.5	2.5	∞	∞
51	7	(6,7)	(7,2)	1	2	∞	∞
52	7	(6,7)	(7,8)	1.5	2.5	3	4
53	7	(8,7)	(7,6)	1	2	3	4
54	7	(8,7)	(7,2)	1.5	2.5	∞	∞
55	8	(1,8)	(8,16)	1	2	3	∞
56	8	(1,8)	(8,7)	2	3	4	∞
57	8	(7,8)	(8,1)	1	2	3	∞
58	8	(7,8)	(8,16)	1.5	2.5	3.5	4.5
59	9	(4,9)	(9,13)	1	2	∞	∞
60	9	(4,9)	(9,10)	1.5	2.5	∞	∞
61	9	(13,9)	(9,10)	1	2	∞	∞
62	9	(13,9)	(9,4)	1.5	2.5	∞	∞
63	9	(10,9)	(9,4)	1	2	∞	∞
64	9	(10,9)	(9,13)	1.5	2.5	∞	∞
65	10	(3,10)	(10,9)	1	2	∞	∞
66	10	(3,10)	(10,11)	2	3	∞	∞
67	10	(3,10)	(10,13)	1.5	2.5	3	4
68	10	(13,10)	(10,9)	1	2	∞	∞
69	10	(13,10)	(10,11)	2	3	∞	∞
70	10	(13,10)	(10,3)	1.5	2.5	3	4
71	10	(9,10)	(10,13)	1	2	∞	∞
72	10	(9,10)	(10,11)	2	3	∞	∞
73	10	(9,10)	(10,3)	1.5	2.5	∞	∞
74	10	(11,10)	(10,3)	1	2	∞	∞
75	10	(11,10)	(10,9)	2	3	∞	∞
76	10	(11,10)	(10,13)	1.5	2.5	∞	∞
77	11	(2,11)	(11,14)	1	2	∞	∞
78	11	(14,11)	(11,12)	1	2	∞	∞
79	11	(14,11)	(11,2)	1	2	∞	∞
80	11	(14,11)	(11,10)	2	3	∞	∞
81	11	(10,11)	(11,12)	1	2	∞	∞
82	11	(10,11)	(11,2)	1.5	2.5	∞	∞

83	11	(12,11)	(11,2)	1	2	∞	∞
84	11	(12,11)	(11,10)	1.5	2.5	∞	∞
85	12	(1,12)	(12,11)	1	2	∞	∞
86	12	(1,12)	(12,15)	1.5	2.5	3.5	∞
87	12	(11,12)	(12,15)	1	2	∞	∞
88	12	(11,12)	(12,1)	1.5	2.5	∞	∞
89	12	(15,12)	(12,1)	1	2	3	∞
90	12	(15,12)	(12,11)	1.5	2.5	∞	∞
91	13	(9,13)	(13,14)	1	2	∞	∞
92	13	(9,13)	(13,10)	1.5	2.5	∞	∞
93	13	(10,13)	(13,9)	1	2	∞	∞
94	13	(10,13)	(13,14)	1.5	2.5	∞	∞
95	13	(14,13)	(13,10)	1	2	∞	∞
96	13	(14,13)	(13,9)	1.5	2.5	∞	∞
97	14	(13,14)	(14,15)	1	2	∞	∞
98	14	(13,14)	(14,11)	1.5	2.5	∞	∞
99	14	(11,14)	(14,13)	1	2	∞	∞
100	14	(11,14)	(14,15)	1.5	2.5	∞	∞
101	14	(15,14)	(14,11)	1	2	∞	∞
102	14	(15,14)	(14,13)	1.5	2.5	∞	∞
103	15	(14,15)	(15,16)	1	2	3	∞
104	15	(14,15)	(15,1)	1.5	2.5	3.5	∞
105	15	(14,15)	(15,12)	2	3.5	4.5	∞
106	15	(12,15)	(15,14)	1	2	3	∞
107	15	(12,15)	(15,16)	1.5	2.5	3.5	∞
108	15	(12,15)	(15,1)	2	3.5	4.5	∞
109	15	(1,15)	(15,12)	1	2	3	∞
110	15	(1,15)	(15,14)	1.5	2.5	3.5	∞
111	15	(1,15)	(15,16)	2	3.5	4.5	∞
112	15	(16,15)	(15,1)	1	2	3	∞
113	15	(16,15)	(15,12)	1.5	2.5	3.5	∞
114	15	(16,15)	(15,14)	2	3.5	4.5	∞
115	16	(8,16)	(16,15)	1	2	3	∞
116	16	(15,16)	(16,8)	1	2	3	∞

```

Edge (12,11): conv.flow 0.626,
real flow 0.626
Edge ( 1,12): conv.flow 0.626,
real flow 0.626
Source: 1, Target: 5,
conv.flow:376.565, real flow:376.565
Edge ( 1, 2): conv.flow 3.967,
real flow 3.967
Edge ( 2, 3): conv.flow 1.088,
real flow 1.088
Edge ( 6, 5): conv.flow 376.565,
real flow 376.565
Edge ( 7, 6): conv.flow 375.477,
real flow 375.477
Edge ( 8, 7): conv.flow 372.257,
real flow 372.257
Edge ( 3, 6): conv.flow 1.088,
real flow 1.088
Edge ( 2, 7): conv.flow 3.221,
real flow 3.221
Edge ( 1, 8): conv.flow 372.257,
real flow 372.257
Edge (11, 2): conv.flow 0.341,
real flow 0.341
Edge (12,11): conv.flow 0.341,
real flow 0.341
Edge ( 1,12): conv.flow 0.341,
real flow 0.341
Source: 1, Target: 9,
conv.flow:679.106, real flow:679.106
Edge (10, 9): conv.flow 362.815,
real flow 362.815
Edge (13, 9): conv.flow 316.290,
real flow 316.290
Edge (13,10): conv.flow 0.796,
real flow 0.796
Edge (11,10): conv.flow 362.019,
real flow 362.019
Edge (12,11): conv.flow 362.019,
real flow 362.019
Edge ( 1,12): conv.flow 362.019,
real flow 362.019
Edge ( 1,15): conv.flow 317.087,
real flow 317.087
Edge (14,13): conv.flow 317.087,
real flow 317.087
Edge (15,14): conv.flow 317.087,
real flow 317.087
* Commodity type: 2
Source: 12, Target: 4,
conv.flow:0.000, real flow:0.000
Source: 12, Target: 5,
conv.flow:0.000, real flow:0.000
Source: 12, Target: 9,
conv.flow:36.060, real flow:7.212
Edge (10, 9): conv.flow 36.060,
real flow 7.212
Edge (11,10): conv.flow 36.060,
real flow 7.212

```

#### 4.2. Test

The algorithm is coded in programming language C and gives reliable results, what is verified by the following test.

```

Limited Cost : 60000.000
Approx. ratio : 0.100
Total flow : 2753.695
Total cost : 58703.329

```

```

* Commodity type: 1
Source: 1, Target: 4,
conv.flow:445.969, real flow:445.969
Edge ( 1, 2): conv.flow 445.343,
real flow 445.343
Edge ( 2, 3): conv.flow 445.969,
real flow 445.969
Edge ( 3, 4): conv.flow 445.969,
real flow 445.969
Edge (11, 2): conv.flow 0.626,
real flow 0.626

```

```
Edge (12,11): conv.flow 36.060,  
real flow 7.212  
* Commodity type: 3  
Source: 12, Target:13,  
conv.flow:0.000, real flow:0.000  
Source: 12, Target:16,  
conv.flow:898.074, real flow:89.807  
Edge ( 8,16): conv.flow 265.517,  
real flow 26.552  
Edge ( 1, 8): conv.flow 265.517,  
real flow 26.552  
Edge (12, 1): conv.flow 265.517,  
real flow 26.552  
Edge (12,15): conv.flow 632.558,  
real flow 63.256  
Edge (15,16): conv.flow 632.558,  
real flow 63.256  
Source: 13, Target:16,  
conv.flow:317.921, real flow:31.792  
Edge ( 6, 7): conv.flow 317.921,  
real flow 31.792  
Edge ( 7, 8): conv.flow 317.921,  
real flow 31.792  
Edge ( 8,16): conv.flow 317.921,  
real flow 31.792  
Edge ( 3, 6): conv.flow 317.921,  
real flow 31.792  
Edge (10, 3): conv.flow 317.921,  
real flow 31.792  
Edge (13,10): conv.flow 317.921,  
real flow 31.792  
* Commodity type: 4  
Source: 13, Target:16,  
conv.flow:0.000, real flow:0.000
```

## 5. CONCLUSIONS

The presented paper introduces the algorithm to find the shortest path between two vertices on extended graph with multiple weights. Then, the shortest path finding algorithm is used to implement the general algorithm finding the maximum flow limited cost on the extended multicommodity multicut network developed in the article [14]. The algorithm is coded in the programming language C and tested in section 4. The results of this paper are the basis for studying the further multicommodity multicut flow optimization problems.

## 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 Cost Between Arcs*, Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol IIMECS 2008, 19-21 March, 2008, Hong Kong.
- [3] Tran Quoc Chien: *Linear multi-channel traffic network*, Ministry of Science and Technology, code B2010DN-03-52.
- [4] Tran Quoc Chien, Tran Thi My Dung: *Application of the shortest path finding algorithm to find the maximum flow of goods*. *Journal of Science & Technology*, University of Danang, 3 (44) 2011.
- [5] Tran Quoc Chien: *Application of the shortest multi-path finding algorithm to find the maximum simultaneous flow of goods simultaneously*. *Journal of Science & Technology*, University of Danang, 4 (53) 2012.
- [6] Tran Quoc Chien: *Application of the shortest multi-path finding algorithm to find the maximal simultaneous flow of goods simultaneously the minimum cost*. *Journal of Science & Technology*, Da Nang University, 5 (54) 2012.
- [7] Tran Quoc Chien: *The algorithm finds 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 finds the shortest path on the extended graph*. Proceedings of the 6th National Conference on Fundamental and Applied Information Technology (FAIR), Proceedings of the Sixth National Conference on Scientific Research and Application, Hue, 20-21 June 2013. Publisher of Natural Science and Technology. Hanoi 2013. p.522-527.
- [9] Tran Quoc Chien: *Applying the algorithm to find the fastest way to find the maximum linear and simultaneous minimum cost on an extended transportation network*, *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 Expanded Transport 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 problem of linear multi-channel traffic flow in traffic network*. Proceedings of the 7th National Conference on Fundamental and Applied Information Technology Research (FAIR'7), ISBN: 978-604-913-300-8, p.31-39. Publisher of Natural Science and Technology. Hanoi 2014.
- [12] Tran Quoc Chien, Ho Van Hung: *Extended linear multicommodity multicut network and maximal flow finding problem*. Proceedings of the 7th National Conference on Fundamental and Applied Information Technology Research (FAIR'10), ISBN: 978-604-913-614-6, p.385-395. Publisher of Natural Science and Technology. Hanoi 2017.
- [13] Tran Quoc Chien, Ho Van Hung: *Applying algorithm finding shortest path in the multiple-*



*weighted graphs to find maximal flow in extended linear multicommodity multicommodity network*, EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, 12.2017, Volume 4, Issue 11, p 1-6.

- [14] Tran Quoc Chien, Ho Van Hung: *Extended Linear Multi-Commodity Multi-Cost Network and Maximal Flow Limited Cost Problems*, The International Journal of Computer Networks & Communications (IJCNC), Volue 10, No. 1, January 2018, p 79-93.

## **Authors**

**First Author:** M.Si. Ho Van Hung (<http://qnamuni.edu.vn/viewLLKH.asp?MaGV=134>). Born in 1977 in Thang Binh, Quang Nam, Vietnam. He graduated from Faculty of Information Technology – College of Sciences – Hue University in 2000. He got master of science (IT) at Danang university of technology. His main major: Applicable mathematics in transport, maximum flow, parallel and distributed process, graph theory and distributed programming.

**Second Author:** Ass. Prof. DrSc. Tran Quoc Chien ([http://scv.ued.udn.vn/ly\\_lich/chi\\_tiet/275](http://scv.ued.udn.vn/ly_lich/chi_tiet/275)). He has 14 papers in SCIE of Journal (<http://www.kybernetika.cz>). Born in 1953 in Dien Ban, Quang Nam, Vietnam. He graduated from Maths\_IT faculty. He got Ph.D Degree of maths in 1985 in Charles university of Prague, Czech Republic and hold Doctor of Science in Charles university of Prague, Czech Republic in 1991. He received the title of Ass. Pro in 1992. He work for university of Danang, Vietnam. His main major: Maths and computing, applicable mathematics in transport, maximum flow, parallel and distributed process, discrete mathematics, graph theory, grid Computing, distributed programming.