IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 22, Issue 2, Ser. II (Mar - Apr 2020), PP 34-44
www.iosrjournals.org

Implement Algorithm Finding Maximal Concurent Limited Cost Flow on Extended Multi-commodity Multi-cost Network

Ho Van Hung[1], Tran Quoc Chien[2]
[1] *Faculty of Information Technology, Quangnam University, Tamky, Vietnam*
[2] *The University of Education, University of Danang, Danang, Vietnam*

Abstract: *Graph is a powerful mathematical tool applied in many fields as transportation, economy,communication, informatics, …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 node are not the same for all paths passing this vertex, but depend on coming and leavingedges. Furthermore, on a network, many commodities share capacities of edges and nodes with different costs. So it is necessary to study network with mutiple weights.In the presented paper, the algorithm finding shortest path in network with multiple weights is applied to imstall the general algorithm finding the maximum concurent limited cost flow on the multi-cost multi-commodity extended network.*

Keywords: *Graph,Network,Multi-commodityMulti-cost Flow,Optimization,Linear Programming.*

Date of Submission: 16-04-2020 Date of Acceptance: 01-05-2020

I. Introduction

A network and a flow network is a useful device to solve many problems in many fields in reality. However, most of the network applications in traditional graphs have only considered the weights of edges and vertexes independently, in which the length of a path is the sum of weights of the edges and the vertexes on the path. However, in many practical problems, weights at a vertex are not the same for all paths passing the vertex, but depend on the edges coming to and leaving the vertex. For example, the transit time on the transport network depends on the direction of transportation: turn left turn right, or go straight, even some directions arebanned. The idea of using duality theory of linear optimization to solve these problems is motivated by the work [1]. The paper [2] propose switching cost only for directed graphs. Multi-commodity flows in traditional network have been studied in the works [1,3,4,5,6]. Multi-commodity flow in extended network problems with extended transport networks were studied in the works [7, 8, 9, 10, 11,12, 14].Furthermore, on a network, many commodities share capacities of edges and nodes with different costs. So it is needed to study networks with mutiple weights. The contributions [15] and [16] study maximal flow problems on extended multi-costmulti-commodity networks.The works [17] and [18] studymaximal flow limited cost problems on extended multi-costmulti-commoditynetworks. The works [19] and [20] study maximal concurent flow problems on extended multi-commoditymulti-cost networks.And finallythe paper [21] studies maximal concurent limited cost flow problems on extended multi-costmulti-commodity networks and developed a polynomial method to find maximal concurent limited cost flows.

In the presented paper, the algorithm finding shortest path in network with multiple weights is applied to imstall the general algorithm finding the maximum concurent limited cost flow on the multi-costmulti-commodity extended network developed in the article [21]. The content of the paper is as follows. The maximal concurent limited cost flow problems on extended multi-commoditymulti-cost networks is introduced in section 2. In section 3, the shortest path finding algorithms is used to implement the general algorithm finding the maximal concurentlimited cost flow on the extended multi-commoditymulti-costnetwork developed in the article [21]. The algorithm is coded in the programming language C and tested in section 4.

II. Maximal Concurent Limited Cost Flow Problems In Extended Multi-commodity Multi-cost Network

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

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

The following functions are defined on the graph G:

Edge passing capacity function $ce:E\to R^*$, where $ce(e)$is the passing capabilityof theedge $e \in E$.

Edge service coefficientfunction $ze:E\to R^*$, *where* $ze(e)$is the passingratio of the edge $e\in E$ (the real capacity of the edge e is $ze(e).ce(e)$).

Node passing capacity functioncv:$V\to R^*$, where$cv(u)$ is the passingcapacity of the node$u\in V$.

Node service coefficient function zv:$V\to R^*$, where$zv(u)$ is the passingratio of the node $v\in V$ (the real capacity of the node v is $zv(v).cv(v)$).

The tuples (V,E, ce, ze, cv, zv) arecalled*extended networks.*

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

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

The sets $((V, E, ce, ze, cv, zv,\{be_i,bv_i, q_i|i=1, \dots, r\})$ are called*the extended multi-commoditymulti-cost network*.

Note: If $be_i(e)=\infty$, commodity of type $i$is prohibited from passing on the edge$e$. If $bv_i(u,e,e') = \infty$,commodity of type $i$is banned from passing on the edge e through u to the edgee'.

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

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

The cost of circulating a converted unit of commodity of type i, i = 1, ..., r, through the path p, is denoted by the symbol $b_i(p)$, and definedby 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 multi-costmulti-commoditynetwork$G=(V,E,ce, ze, cv, zv, \{be_i, bv_i, q_i|i=1, \dots, r\})$. Assume, for each commodityof kind$i$, 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 kind$i$, that is necessary to move from source node $s_{i,j}$ to target vertex$t_{i,j}$.

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 through,i=1, ..., r,j=1, ..., k_i.

Set

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

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 node$s_{i,j}$to the destination node$t_{i,j}$ along the path$p$.

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

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

is called a*multi-commodity flow* on the extended multi-commoditymulti-cost network, if it satisfies the following *edge and node capacity* constraints:

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

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

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 kind i of the source-destination pair* $(s_{i,j}, t_{i,j})$ *of F*.

The expresstions

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

arecalled *the flow value of commodity of type i of F*.

The expresstion

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

is called the flow value of F.

Given an extendedmulti-costmulti-commoditynetwork $G=(V,E,ce, ze,cv,zv,\{be_i,bv_i, q_i|i=1,\ldots, r\})$. Assume, for each commodity of kind i, $i=1, \ldots, r$, there are k_i source-destination pairs $(s_{i,j}, t_{i,j})$, $j=1, \ldots, k_i$, each pair assigned a quantity $D_{i,j}$ of commodity of type i, that is necessary to move from source node $s_{i,j}$ to destination node $t_{i,j}$. A limited cost B is given.

The task of the problem is to find the maximal ratio λ such that there exists a flow converting $\lambda.D_{i,j}$ commodity type i, $\forall i=1, \ldots, r$, from source node $s_{i,j}$ to destination node $t_{i,j}$, $\forall j=1, \ldots, k_i$, and the total cost doex not exceed the limited cost B.

Put $d_{i,j} = q_i.D_{i,j}$, $\forall i=1, \ldots, r$, $\forall j=1, \ldots, k_i$

The problem is expressed byan implicit linear programming modelas follows:

$$\lambda \rightarrow \quad max$$

satisfies

$$\left.\begin{array}{l}
\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{p \in P_{i,e}} x_{i,j}(p) \le ce(e).ze(e), \; \forall e \in E \\
\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{p \in P_{i,v}} x_{i,j}(p) \le cv(v).zv(v), \; \forall v \in V \\
\sum_{p \in P_{i,j}} x_{i,j}(p) \ge \lambda.d_{i,j}, \; \forall i=1, \ldots, r, \; \forall j = 1, \ldots, k_i \\
\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{p \in P_{i,j}} x_{i,j}(p).b_i(p) \le B \\
\lambda \ge 0, \; x_{i,j}(p) \ge 0, \; \forall i=1, \ldots, r, \; \forall j = 1, \ldots, k_i, \; \forall p \in P_{i,j}
\end{array}\right\} (P)$$

III. Installing of Algorithm

The generalmethod with polynomial complexity is proved in [21]. In this contribution we integrate the algorithm finding shortest path [7,8] in order to install the mentioned method [21].

◊***Input:*** Extended multi-costmulti-commoditynetwork $G=(V,E, ce, ze, cv, zv, \{be_i, bv_i, q_i|i=1. \ldots, r\})$, $n=|V|$, $m=|E|$. Assume, for each commodity of type i, $i=1, \ldots, r$, there are k_i source-destination pairs $(s_{i,j}, t_{i,j})$, $j=1, \ldots, k_i$, each pair assigned a quantity $D_{i,j}$ of commodity of kind i, that is needed to move from source node $s_{i,j}$ to destination node $t_{i,j}$. Given a limited cost B and an approximation ratio ω.

◊***Output***:

Maximal multi-commodity concurent flow F represents a set of converted commodity flows at edges

$$F = \{f_{i,j}(e) \mid e \in E, i=1, \ldots, r, j=1, \ldots, k_i\}$$

Maximal multi-commodity concurent flow rF represents a set of real commodity flows at edges

$$rF = \{rf_{i,j}(e) \mid e \in E, i=1, \ldots, r, j=1, \ldots, k_i\}$$

Total cost $B_f \le B$ and maximal concurent coefficient λ.

◊***Procedure***

```
//Choose ε, δ and initialization
ε = 1 − ∛(1/(1+ω)) ;
δ = ((m+n+1)/(1−ε))^(−1/ε) ;
for(i=1; i <= r; i++)
  for(j=1; i <= k_i; j++)
        d_i,j = D_i,j*q_i;
for (e∈E) le(e)=δ/(ce(e)ze(e));
for (v∈V) lv(v)=δ/(cv(v)zv(v));
φ = δ/B ;
for(i=1; i <= r; i++)
  for(j=1; i <= k_i; j++)
        for (e∈E)
        x_i,j(e)=0 ;
B_f = 0; D1 = (m+n+1)*δ; t = 0 ;
// Note
dist  the shortest path length;
p  the shortest path;
c  the minimal edge and node capacity on the path p.
be_i(p) the cost of commodity type i on the path p, i=1, ..., r.
// main algorithm body
do
{
 for(i=1; i <= r; i++)
  for(j=1; i <= k_i; j++)
  {
    d' = d_i,j;
  do
        {
            Find the shortest pathpfrom the source nodes_i,jtothe target node t_i,jwith
        thelengthfunctionlength(.)[21]. The pathpmust be suited to thecommodity of type i, i.e. it
        does not contain edges with cost ∞ and vertex with switch cost ∞ for thecommodity of
        type i.
        Setdist= length(p);
          c=min{min{ce(e)*ze(e)|e∈p},min{cv(v)*zv(v)|v∈p},d'}
        //Flow adjustments:
  for(e∈p)x_i,j(e)=x_i,j(e)+c;
        for(e∈p) le(e)=le(e)*(1+ε*c/(ce(e)*ze(e)));
        for(v∈p) lv(v)=lv(v)*(1+ε*c/(cv(v)*zv(v)));
                D1 = D1 + ε*c*dist;d' = d'-c;
                B_f = B_f + c*be_i(p);
  } while (d'>0)
  }
if (D1 < 1)
    for (i=1; i<= r; i++)
        for (j=1; j<= k_i; j++)
                for (e∈E)
                f_i,j(e) = x_i,j(e);
  t++ ;
} while (D1<1)
// Modifying the resulting flows F and flowcost.
for(i=1; i <= r; i++)
```

```
      for(j=1; i <= ki; j++)
       for (e∈E)
              fi,j(e)= fi,j(e)/(-log1+εδ);
              Bf = Bf/(-log1+εδ);
    // Modifyingflows on scalar edge
    for(i=1; i <= r; i++)
      for(j=1; i <= ki; j++)
       for (e∈E&& e scalar)
         if fi,j(e)>=fi,j(e')//e' is the opposite of the direction e
              {
                   Bf=Bf-fi,j(e')(bei(e)+bei(e')) ;fi,j(e)=fi,j(e)–fi,j(e') ;
                   fi,j(e')=0 ;
              }
         else
              {
                   Bf = Bf - fi,j(e)(bei(e)+bei(e')) ;
                   fi,j(e')=fi,j(e')–fi,j(e) ;
                   fi,j(e)=0 ;
              }
      //Convert the flow fi,j(e) to the actual flow rfi,j(e) by dividing the conversion flow by the
conversion coefficient
    for(i=1; i <= r; i++)
      for(j=1; i <= ki; j++)
    for (e∈E)
              rfi,j(e) = fi,j(e)/qi ;
    // maximal concurent ratio
    λ = t/(-log1+εδ);
    /*** end of program ***/
```

IV. 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.

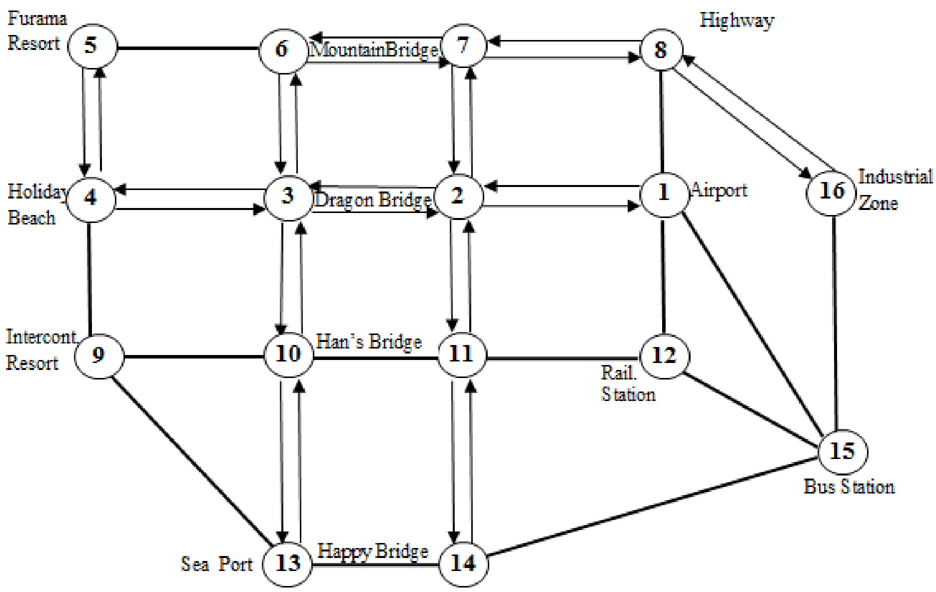

Figure. DaNang City

Table 1. Node capability and service coefficient

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 conversion coefficient

Commodity	Vehicle	*q*
1	Motor car	1
2	Light truck	5
3	Heavy truck	10
4	Container truck	20

Table 3: Pairs of commodity source-target nodes and $D_{i,j}$

No	Commodity	$s_{i,j}$	$t_{i,j}$	$D_{i,j}$
1	1	1	4	200
2	1	1	5	150
3	1	1	9	300
4	2	12	4	50
5	2	12	5	50
6	2	12	9	25
7	3	12	13	25
8	3	12	16	25
9	3	13	16	25
10	4	13	16	10

Table 4: Edge capacity, service coeffitient and cost

No	Edge	Type	*ce*	*ze*	be_1	be_2	be_3	be_4
1	(1,2)	1	500	0.9	3	4	∞	∞
2	(2,1)	1	500	0.9	3	4	∞	∞
3	(2,3)	1	500	0.9	3	4	∞	∞
4	(3,2)	1	500	0.9	3	4	∞	∞
5	(3,4)	1	500	0.9	3	4	∞	∞
6	(4,3)	1	500	0.9	3	4	∞	∞
7	(4,5)	1	500	0.9	3	4	∞	∞
8	(5,4)	1	500	0.9	3	4	∞	∞
9	(5,6)	0	700	0.8	3	4	∞	∞
10	(6,5)	0	700	0.8	3	4	∞	∞
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	∞	∞
20	(7,2)	1	500	0.9	3	4	∞	∞
21	(1,8)	0	800	0.8	3	4	6	∞
22	(8,1)	0	800	0.8	3	4	6	∞
23	(4,9)	0	600	0.8	4	5	∞	∞
24	(9,4)	0	600	0.8	3	4	∞	∞
25	(10,9)	0	700	0.8	4	5	∞	∞
26	(9,10)	0	700	0.8	3	4	∞	∞
27	(9,13)	0	500	0.8	3	4	∞	∞
28	(13,9)	0	500	0.8	4	5	∞	∞
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	∞

4.2. Test

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

Limited cost : 50000.000
Approximation ratio : 0.050
Maximal Concurent ratio: 0.689
Total cost : 49647.969
* Commodity type: 1
Source: 1, Target: 4, conv.flow: 137.851, real flow: 137.851
Edge (1, 2): conv.flow 137.839, real flow 137.839
Edge (2, 3): conv.flow 137.839, real flow 137.839
Edge (3, 4): conv.flow 137.839, real flow 137.839
Source: 1, Target: 5, conv.flow: 103.389, real flow: 103.389
Edge (1, 2): conv.flow 102.915, real flow 102.915
Edge (2, 3): conv.flow 56.541, real flow 56.541
Edge (3, 4): conv.flow 6.942, real flow 6.942
Edge (4, 5): conv.flow 6.942, real flow 6.942
Edge (6, 5): conv.flow 96.437, real flow 96.437
Edge (7, 6): conv.flow 46.838, real flow 46.838
Edge (8, 7): conv.flow 0.465, real flow 0.465
Edge (3, 6): conv.flow 49.599, real flow 49.599
Edge (2, 7): conv.flow 46.374, real flow 46.374
Edge (1, 8): conv.flow 0.465, real flow 0.465
Source: 1, Target: 9, conv.flow: 206.777, real flow: 206.777
Edge (10, 9): conv.flow 145.098, real flow 145.098
Edge (13, 9): conv.flow 61.661, real flow 61.661
Edge (11,10): conv.flow 145.098, real flow 145.098
Edge (12,11): conv.flow 145.098, real flow 145.098

Edge (1,12): conv.flow 145.098, real flow 145.098
Edge (1,15): conv.flow 61.661, real flow 61.661
Edge (14,13): conv.flow 61.661, real flow 61.661
Edge (15,14): conv.flow 61.661, real flow 61.661
* Commodity type: 2
Source:12, Target: 4, conv.flow: 172.314, real flow: 34.463
Edge (1, 2): conv.flow 82.194, real flow 16.439
Edge (2, 3): conv.flow 98.882, real flow 19.776
Edge (3, 4): conv.flow 98.882, real flow 19.776
Edge (9, 4): conv.flow 73.417, real flow 14.683
Edge (10, 9): conv.flow 73.417, real flow 14.683
Edge (11,10): conv.flow 73.417, real flow 14.683
Edge (11, 2): conv.flow 16.688, real flow 3.338
Edge (12,11): conv.flow 90.105, real flow 18.021
Edge (12, 1): conv.flow 82.194, real flow 16.439
Source:12, Target: 5, conv.flow: 172.314, real flow: 34.463
Edge (1, 2): conv.flow 56.487, real flow 11.297
Edge (2, 3): conv.flow 10.933, real flow 2.187
Edge (3, 4): conv.flow 0.972, real flow 0.194
Edge (4, 5): conv.flow 3.371, real flow 0.674
Edge (6, 5): conv.flow 168.928, real flow 33.786
Edge (7, 6): conv.flow 158.967, real flow 31.793
Edge (8, 7): conv.flow 91.198, real flow 18.240
Edge (3, 6): conv.flow 9.961, real flow 1.992
Edge (2, 7): conv.flow 67.769, real flow 13.554
Edge (1, 8): conv.flow 91.198, real flow 18.240
Edge (9, 4): conv.flow 2.399, real flow 0.480
Edge (10, 9): conv.flow 2.399, real flow 0.480
Edge (11,10): conv.flow 2.399, real flow 0.480
Edge (11,2): conv.flow 22.215, real flow 4.443
Edge (12,11): conv.flow 24.614, real flow 4.923
Edge (12,1): conv.flow 147.685, real flow 29.537
Source:12, Target: 9, conv.flow: 86.157, real flow: 17.231
Edge (10,9): conv.flow 82.513, real flow 16.503
Edge (13, 9): conv.flow 3.637, real flow 0.727
Edge (11,10): conv.flow 82.513, real flow 16.503
Edge (12,11): conv.flow 82.513, real flow 16.503
Edge (12,15): conv.flow 3.637, real flow 0.727
Edge (14,13): conv.flow 3.637, real flow 0.727
Edge (15,14): conv.flow 3.637, real flow 0.727
* Commodity type: 3
Source:12, Target:13, conv.flow: 172.314, real flow: 17.231
Edge (7, 6): conv.flow 172.299, real flow 17.230
Edge (8, 7): conv.flow 172.299, real flow 17.230
Edge (6, 3): conv.flow 172.299, real flow 17.230
Edge (1,8): conv.flow 172.299, real flow 17.230
Edge (3,10): conv.flow 172.299, real flow 17.230
Edge (10,13): conv.flow 172.299, real flow 17.230
Edge (12,1): conv.flow 172.299, real flow 17.230
Source:12, Target:16, conv.flow: 172.314, real flow: 17.231
Edge (12,15): conv.flow 172.299, real flow 17.230
Edge (15,16): conv.flow 172.299, real flow 17.230
Source:13, Target:16, conv.flow: 172.314, real flow: 17.231
Edge (6,7): conv.flow 172.299, real flow 17.230
Edge (7,8): conv.flow 172.299, real flow 17.230
Edge (8,16): conv.flow 172.299, real flow 17.230
Edge (3, 6): conv.flow 172.299, real flow 17.230
Edge (10, 3): conv.flow 172.299, real flow 17.230

Edge (13,10): conv.flow 172.299, real flow 17.230
* Commodity type: 4
Source:13, Target:16, conv.flow: 137.851, real flow: 6.893
Edge (6, 7): conv.flow 137.839, real flow 6.892
Edge (7, 8): conv.flow 137.839, real flow 6.892
Edge (8,16): conv.flow 137.839, real flow 6.892
Edge (3,6): conv.flow 137.839, real flow 6.892
Edge (10, 3): conv.flow 137.839, real flow 6.892
Edge (13,10): conv.flow 137.839, real flow 6.892

V. Conclusions

The presented contribution use the algorithm finding shortest path in multi-weight graphs to install the general method determining the maximal concurent limited cost flows on multi-costmulti-commodity extended networks developed in the work [21]. The program was installed in the language C and has given reliable tests.

References

[1]. Naveen Garg, Jochen Könemann, "Faster and Simpler Algorithms for Multi-commodity 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, Hong Kong, **(2008)**.

[3]. Tran Quoc Chien, "Linear multi-channel traffic networ", Ministry of Science and Technology, code B2010DN-03-52, **(2010).**

[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", 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 with minimal 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)**, p.p 16-21.

[8]. Tran Quoc Chien, Nguyen Mau Tue, Tran Ngoc Viet "The algorithm finds the shortest path on the extended graph." Proceeding of the 6th National Conference on Fundamental and Applied Information Technology (FAIR), 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)**, pp.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)**, pp.121-124.

[11]. Tran Ngoc Viet, Tran Quoc Chien, Le Manh Thanh, "The Revised Ford-Fulkerson Algorithm Finding Maximal Flows on Extended Networks," International Journal of Computer Technology and Applications, vol. 5, no. 4, **(2014)**, pp. 1438-1442.

[12]. Viet Tran Ngoc, Chien Tran Quoc, Tau Nguyen Van, "Improving Computing Performance for Algorithm Finding Maximal Flows on Extended Mixed Networks," Journal of Information and Computing Science, England, UK, vol. 10, no. 1, **(2015)**, pp. 075-080.

[13]. Xiangming Yao, Baomin Han, Baomin Han, Hui Ren, "Simulation-Based Dynamic Passenger Flow Assignment Modelling for a Schedule-Based Transit Network;" Discrete Dynamics in Nature and Society- Hindawi, **(2017)**.

[14]. 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, Publisher of Natural Science and Technology. Hanoi, **(2014)**, pp.31-39

[15]. Tran Quoc Chien, Ho Van Hung, " Extended linear multi-commoditymulti-cost 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, Publisher of Natural Science and Technology. Hanoi, **(2017)**, pp.385-395

[16]. Tran Quoc Chien, Ho Van Hung ,"Applying algorithm finding shortest path in the multiple-weighted graphs to find maximal flow in extended linear multicomodity multi-cost network", EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, , Volume 4, Issue 11, **(2017)**, pp 1-6.

[17]. 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, **(2018)**, pp 79-93. (SCOPUS)

[18]. Ho Van Hung, Tran Quoc Chien, "Implement and Test Algorithm finding Maximal Flow Limited Cost in extended multicomodity multi-cost network", The International Journal of Computer Techniques(IJCT), Volume 6 Issue 3, **(2019)**, p.p 1-9.

[19]. Ho Van Hung, Tran Quoc Chien, "Extended Linear Multi-Commodity Multi-Cost Network and Maximal Concurent Flow Problems", he International Journal of Mobile Netwwork Communications & Telematics, Vol.9, No.1, **(2019)**, pp 1-14.

[20]. Ho Van Hung, Tran Quoc Chien, "Installing Algorithm to find Maximal Concurent Flow in Multi-costMulti-commodity Extended", International Journal of Innovative Science and Research Technology (IJISRT), Volue 4, Issue 12, **(2019)**, pp 1110-1119.

[21]. Ho Van HUng, Tran Quoc Chien, "Maximal Concurent Limited Cost Flow Problems on Extended Linear Multi-Commodity Multi-Cost Networks", American Journal of Applied Mathematics , Accepted and to be published, **(2020)**.

Ho Van Hung,etal. "Implement Algorithm Finding Maximal Concurent Limited Cost Flow on Extended Multi-commodity Multi-cost Network." *IOSR Journal of Computer Engineering (IOSR-JCE)*, 22.2 (2020), pp. 34-44.