# Installing Algorithm to Find Maximal Concurent Flow in Multicost Multicommodity Extended Network

Ho Van Hung
Quangnam University
Tamky City – Vietnam

Tran Quoc Chien
Danang University
Danang City – Vietnam

**Abstract:- Graph is an excelent mathematical means used in many applications as communication, transportation, economy, informatics, … In traditional graph the weights of vertexes and edges are independent, where the distance of a path is the sum of weights of the vertexes and the edges on that path. On the other side, in many practical applications, weights at vertexes are not equal for all paths going through these vertexex, but are depending on leaving and coming edges. Furthermore, capacities of vertexes and edges of a network are shared by many commodities with different costs. Hence, studying networks of mutiple weights is needed. In the presented paper, the algorithm finding shortest path in network with multiple weights is applied to implement the general algorithm finding the maximum concurent flow on the multicost multicommodity extended network.**

***Keywords:-*** *Network, Graph, Multicost Multicommodity Flow, Libear Optimization, Aproximation.*

## I. INTRODUCTION

Flows on networks are excelent mathematical means used in many applications as communication, transportation, economy, informatics, …. So far, many applications in networks assume the weights of nodes and edges are independent, where the distance of a path is the sum of weights of the vertexes and the edges on that path. On the other side, in many practical applications, weights at vertexes are not equal for all paths going through these vertexex, but are depending on leaving and coming edges. For example, the time passing a node on the traffic network depends on the direction of vehicles: go straight, turn left or turn right. Even, in some situations, some directions are prohibited. Switching costs for directed networks are introduced in the article [2]. The articles [1,3,4,5,6] studied multicommodity flows on ordinary networks. The articles [7,8,9,10,11] studied multicommodity flows of extended networks. Furthermore, capacities of vertexes and edges of a network are shared by many commodities with different costs. Hence, studying networks of mutiple weights is needed. Maximal flow problems on multicost multicommodity extended networks were studied in the articles [12,13]. Maximal flow limited cost problems on networks were studied in the article [14,15]. The article [16] studies maximal concurent flow problems on extended multicommodity multicost networks and developed a polynomial algorithm to find maximal concurent flow.

The presented work develops an algorithm finding the shortest path between nodes on extended graph of mutiple weights. Then, this mentioned algorithm is applied to install the general method finding the maximum concurent flow on the multicost multicommodity extended network introduced in the work [16]. The content of this work is as following. The section 2 defines the maximal concurent flow problems on multicost multicommodity extended networks. In the section 3, the algorithm finding shortest path is used to install the general method finding maximum concurent flows on the multicost multicommodity extended network studied in the paper [16]. The algorithm is installed in the language C and tested in the section 4. The last section is the conclusion of the paper.

## II. MAXIMAL CONCURENT FLOW PROBLEMS IN MULTICOST MULTICOMMODITY EXTENDED NETWORK

Let $G = (N, A)$ be a mixed graph, where $N$ is the node set and $A$ is the edge set. The edges of the graph may be directed or undirected. For all nodes $v \in N$ we denote symbol $A_v$ the set of edges incident node $v$. There are some kinds of commodities transferring on the network. The nodes and the edges of the graph are shared by commodities with different costs. The commodities transferring on undirected edges in both reverse directions directions share the capacities of the edges

Let $r$ denote the number of commodities, $q_i > 0$ denote the conversion coefficient of commodity $j$, $j = 1, \dots, r$.

- We define the functions:

  *Edge circulating capacity function* $ca{:}A \to R$, where $ca(a)$ is the circulating capability of the edge $a \in A$.

  *Edge circulating ratio function* $za{:}A \to R$, *where* $za(a)$ is the circulating ratio of the edge $a \in A$ (the real capacity of the edge $a$ is $za(a).ca(a)$).

  *Node circulating capability function* $cn{:}N \to R$, where $cn(n)$ is the circulating capability of the vertice $n \in N$.

*Node circulating ratio function* $zn:N\to R$, where $zn(n)$ is the circulating ratio of the vertex $n\in N$ (the real capacity of the vertex $n$ is $zn(n).cn(n)$).

The tuple ($N, A, ca, za, cn, zn$) is called an *extended network.*

*Edge cost function* $j$, $j=1, ..., r$, $ba_j:A\to R$, where $ba_j(a)$ is the cost of circulating $a$ a converted unit of commodity of kind $j$. Note that with undirected edges, the costs of each directions may vary.

*Node switch cost function* $j$, $j=1, ..., r$, $bn_j:N\times A_v\times A_v\to R$, *where* $bn_j(n,a,a')$ is the cost of passing a converted unit of commodity of kind $j$ from edge $a$ through node $n$ to edge $a'$.

The sets ($N, A, ca, za, cn, zn,\{ba_j, bn_j, q_j| j=1, …, r\}$) are called *the multicost multicommodity extended network*.
◊ *Note*: If $ba_j(a)=\infty$, commodity of kinf $j$ is forbidden from passing on edge $a$. If $bn_j(n,a,a') = \infty$, comodity of kind $j$ is forbidden from edge $a$ through node $n$ to edge $a'$.

Let $q$ be a path from vertex $m$ to vertex $n$ through edges $a_j$, $j=1, …, (h+1)$, and vertices $m_j$, $j=1, …, h$ as follows

$$q = [m, a_1, m_1, a_2, m_2, …, a_h, m_h, a_{h+1}, n] \qquad (1)$$

The cost of transferring a converted unit of commodity of kind $j$, $j = 1, …, r$, on the path $q$, is denoted by the symbol $b_j(q)$, and calculated as following:

$$b_j(q) = \sum_{i=1}^{h+1} ba_j(a_i) + \sum_{i=1}^{h} bn_j(m_i, a_i, a_{i+1}) \qquad (2)$$

Given an extended multicommodity multicost network $G=(N, A, ca, za, cn, zn,\{ba_j, bn_j, q_j| j=1, …, r\})$. Assume, for each commodity of 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 $D_{i,j}$ of commodity of kind $i$, that is required to transferred from source vertex $s_{i,j}$ to target vertex $t_{i,j}$.

Let $Q_{i,j}$ denote the set of paths from vertex $s_{i,j}$ to vertex $t_{i,j}$ in $G$, which commodity of kind $i$ can be circulated, $i=1, ..., r$, $j=1, …, k_i$. Let

$$Q_i = \bigcup_{j=1}^{k_i} Q_{i,j}, \ \forall\ i=1, ..., r \qquad (3)$$

For each path $q\in Q_{i,j}$, $i=1, …, r$, $j=1, …, k_i$, denote $x_{i,j}(q)$ the flow of converted commodity of kind $i$ from the source vertex $s_{i,j}$ to the target vertex $t_{i,j}$ along the path $q$.

Let $Q_{i,a}$ denote the set of paths in $Q_i$ passing through the edge $a$, $\forall a\in A$.

Let $Q_{i,n}$ denote the set of paths in $Q_i$ passing through the vertex $n$, $\forall n\in N$.

A set

$$X = \{x_{i,j}(q) \mid q\in Q_{i,j}, i=1, …, r, j=1, …, k_i\} \qquad (4)$$

Is called a *multicommodity flow* on the multicost multicommodity extended network, if the following *node and edge capacity* constraints are satisfied

$$\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{q\in Q_{i,a}} x_{i,j}(q) \le ca(a).za(a),\ \forall a\in A \qquad (5)$$

$$\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{q\in Q_{i,n}} x_{i,j}(q) \le cn(n).zn(n),\ \forall n\in N \qquad (6)$$

The expressions

$$fv_{i,j} = \sum_{q\in Q_{i,j}} x_{i,j}(q),\ i=1, …, r, j=1, …, k_i \qquad (7)$$

Are called *the flow values of commodity of kind i of the pair* $(s_{i,j}, t_{i,j})$ *of the multicommodity flow X.*

The expresstions

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

Are called *the flow values of commodity of kind i of the multicommodity flow X.*

The expresstion

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

is called *the flow value of the multicommodity flow X*.

The mission of the problem is to find the maximal ratio $\lambda$ so that there exists a flow converting $\lambda.D_{i,j}$ commodity type $i$, $\forall i=1..r$, from source vertex $s_{i,j}$ to target vertex $t_{i,j}$, $\forall j = 1, …, k_i$. Put

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

The problem is expressed by the implicit linear programming model as follows:

$$\lambda \to \quad max$$

satisfies

$$\left.\begin{array}{l}
\displaystyle\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{q\in Q_{i,a}} x_{i,j}(q) \le ca(a).za(a),\ \forall a\in A \\
\displaystyle\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{q\in Q_{i,n}} x_{i,j}(q) \le cn(n).zn(n),\ \forall n\in N \\
\displaystyle\sum_{q\in Q_{i,j}} x_{i,j}(q) \ge \lambda.d_{i,j},\ \forall i=1, …, r,\ \forall j = 1, …, k_i \\
x_{i,j}(q) \ge 0,\ \forall i=1, …, r, j=1, …, k_i,\ \forall q\in Q_{i,j}
\end{array}\right\}$$

## III. INSTALLATION OF ALGORITHM OF FINDING MAXIMAL CONCURENT FLOW

The general method finding maximum concurent flows on the multicost multicommodity extended network with polynomial complexity is presented in [16]. In this paper we use the method finding shortest path introduced in [7,8] to install the general method in [16].

➢ ***Input***:

Multicommodity multicost extended network $G=(N, A, ca, za, cn, zn,\{ba_j, bn_j, q_j| j=1, \dots, r\})$, $n=|N|$, $m=|A|$. Assume, for each commodity of kind $i$, $i=1. \dots, r$, there are $k_i$ source-target pairs $(s_{i,j}, t_{i,j})$, $j=1, \dots, k_i$, each pair assigned a quantity $D_{i,j}$ of commodity of kind $i$, that is needed to transfer from source vertex $s_{i,j}$ to target vertex $t_{i,j}$. Let $\omega$ be the required approximation ratio.

➢ ***Output***: Maximal flow $X$

$X = \{x_{i,j}(a) \mid a\in A, i=1, \dots, r, j=1, \dots, k_i\}$,

where $x_{i,j}(a)$ is a converted flow at the edge $a$, and the total cost $B_f$.

➢ ***Procedure***

// Initiolization: compute $\varepsilon$ and $\delta$

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

for(i=1; i <= r; i++)
for(j=1; i <= $k_i$; j++)
$d_{i,j} = D_{i,j}.q_i$;
for ($a\in A$) $la(a)=\delta/(ca(a)*za(a))$;
for ($v\in N$) $ln(v)=\delta/(cn(v)*zn(v))$;
for(i=1; i <= r; i++)
for(j=1; j <= $k_i$; j++)
for ($a\in A$)
$x_{i,j}(a)=0$ ;
$B_f = 0$; $D1 = (m+n)*\delta$ ; $t = 0$ ;

// The following notations shall be used
*dist* is the shortest path length;
*q* is the shortest path;
*c* is the minimal node and edge capacity on the path *q*.
$ba_i(q)$ is the cost of commodity kind $i$ on the path $q$, $i=1, \dots, r$.
// *main algorithm body*
**do**
{
for(i=1; i <= r; i++)
for(j=1; j <= $k_i$; j++)
{
d' = $d_{i,j}$;
**do**
{
Call the procedure find the shortest path $q$ from $s_{i,j}$ to $t_{i,j}$ with the length function *length*(.) [16]. The path *p* must be suited to the commodity of type i, i.e. it does not contain edges with cost ∞ and node with switch cost ∞ for the commodity of type i;
*dist* = length(q);
$c=\min\{\min\{ca(a)*za(a)|a\in q\}, \min\{cn(v).zn(v)|v\in q\},d'\}$
// adjustments of flows:
for($a\in q$)$x_{i,j}(a)=x_{i,j}(a)+c$;
for($a\in q$) $la(a)=le(e)*(1+\varepsilon.c/(ca(a)*za(a)))$;
for($v\in q$) $ln(v) = ln(v)*(1+\varepsilon.c/(cn(v)*zn(v)))$;
D1 = D1 + ε.c.dist;
d' = d'-c;
$B_f = B_f + c.ba_i(q)$;
} while (d'>0)
}
if ($D1 < 1$)
for ($i$=1; $i$ <= $r$; $i$++)
for ($j$=1; $j$ <= $k_i$; $j$++)
for ($a\in A$)
$x_{i,j}(a) = x_{i,j}(a)$;
t++ ;
} **while** ($D1 < 1$)
// Modifying flows $x_{i,j}(a)$ and the flow cost $B_f$.
for(i=1; i <= r; i++)
for(j=1; j <= $k_i$; j++)
for ($a\in A$)
$x_{i,j}(a)=a_{i,j}(a)/(-\log_{1+\varepsilon}\delta)$;
$B_f = B_f/(-\log_{1+\varepsilon}\delta)$;
// Modifying flows on undirected edge
for(i=1; i <= r; i++)
for(j=1; j <= $k_i$; j++)
for ($a\in A$ && a is undirected)
if $x_{i,j}(a)>= x_{i,j}(a')$
//$a'$ is the opposite of the direction $a$
{
$B_f=B_f-x_{i,j}(a')(ba_i(a)+ba_i(a'))$;
$x_{i,j}(a) = x_{i,j}(a)-x_{i,j}(a')$ ;
$x_{i,j}(a')=0$ ;
}
else
{
$B_f=B_f-x_{i,j}(a)(ba_i(a)+ba_i(a'))$;
$x_{i,j}(a') = x_{i,j}(a')-x_{i,j}(a)$;
$x_{i,j}(a)=0$ ;
}
// maximal concurent ratio
$\lambda = t/(-\log_{1+\varepsilon}\delta)$;
// the end

## IV. TEST

➢ *Example*

DaNang is the most dynamic city of Vietnam, where the leaders all over the world were welcomed to take part in the APEC 2017.

The scheme 1 prensents the traffic network of a part of DaNang City. The tables 1, 2, 3, 4 and 5 describe the database of the problem.

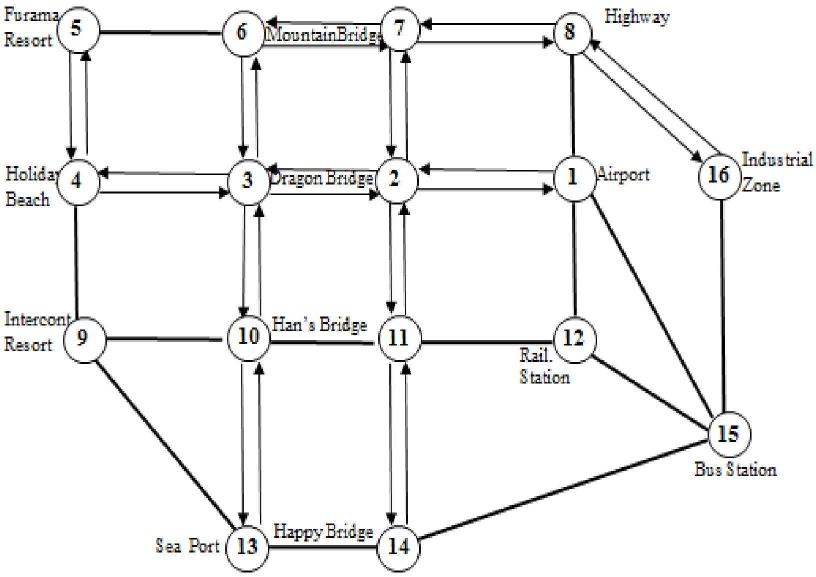


Fig 1:- DaNang City

| Nodes | *cn* | *zn* |
|---|---|---|
| 1 | 1000.00 | 0.70 |
| 2 | 1000.00 | 0.80 |
| 3 | 1000.00 | 0.80 |
| 4 | 500.00 | 0.90 |
| 5 | 500.00 | 0.90 |
| 6 | 1000.00 | 0.80 |
| 7 | 1000.00 | 0.80 |
| 8 | 1500.00 | 0.80 |
| 9 | 500.00 | 0.90 |
| 10 | 1000.00 | 0.80 |
| 11 | 500.00 | 0.80 |
| 12 | 1000.00 | 0.70 |
| 13 | 1500.00 | 0.70 |
| 14 | 1000.00 | 0.80 |
| 15 | 1200.00 | 0.80 |
| 16 | 1500.00 | 0.70 |

Table 1:- Node capability

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

Table 2:- Coefficient of commodity converting

ISSN No:-2456-2165

| 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 3:- Pairs of commodity-source-target nodes

| No | Edge | Kind | $ca$ | $za$ | $ba_1$ | $ba_2$ | $ba_3$ | $ba_4$ |
|---|---|---|---|---|---|---|---|---|
| 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 | ∞ |

Table 4:- Capacity and cost of edges

Notes: Kind 0 resp. 1 means undirectional resp. directional edge.

| No | Node | Edge 1 | Edge 2 | $bn_1$ | $bn_2$ | $bn_3$ | $bn_4$ |
|---|---|---|---|---|---|---|---|
| 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 | ∞ | ∞ |

ISSN No:-2456-2165

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 | ∞ | ∞ |

ISSN No:-2456-2165

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 | ∞ |

Table 5:- Switch Cost

➢ *Test*

The programming language C is used to install the algorithm. The following test gives reliable results.

Approximation ratio : 0.050
Maximal Concurent ratio: 0.772
Total cost : 59392.301
*** Commodity type: 1
 * Source: 1, Target: 4, conv.flow: 154.392, real flow: 154.392

Edge conv.flow real flow
( 1, 2): 42.066, 42.066
( 2, 3): 36.057, 36.057
( 3, 4): 36.057, 36.057
( 5, 4): 90.568, 90.568
( 6, 5): 90.568, 90.568
( 7, 6): 90.568, 90.568
( 8, 7): 84.559, 84.559
( 2, 7): 6.009, 6.009
( 1, 8): 84.559, 84.559
( 9, 4): 27.756, 27.756
(10, 9): 27.756, 27.756
(11,10): 27.756, 27.756
(14,11): 10.203, 10.203
(12,11): 17.553, 17.553
( 1,12): 11.946, 11.946
(15,12): 5.607, 5.607
( 1,15): 15.810, 15.810
(15,14): 10.203, 10.203
 * Source: 1, Target: 5, conv.flow: 115.794, real flow: 115.794
Edge conv.flow real flow
( 1, 2): 48.033, 48.033
( 2, 3): 45.820, 45.820
( 3, 4): 45.820, 45.820
( 4, 5): 89.474, 89.474
( 6, 5): 26.311, 26.311
( 7, 6): 26.311, 26.311
( 8, 7): 24.099, 24.099
( 2, 7): 2.212, 2.212
( 1, 8): 24.099, 24.099
( 9, 4): 43.654, 43.654
(10, 9): 43.654, 43.654
(11,10): 43.654, 43.654
(14,11): 7.625, 7.625
(12,11): 36.029, 36.029
( 1,12): 3.712, 3.712
(15,12): 32.317, 32.317
( 1,15): 39.942, 39.942
(15,14): 7.625, 7.625
 * Source: 1, Target: 9, conv.flow: 231.589, real flow: 231.589
 Edge conv.flow real flow
(13, 9): 231.570, 231.570
( 1,15): 231.570, 231.570
(14,13): 231.570, 231.570
(15,14): 231.570, 231.570
*** Commodity type: 2
 * Source: 12, Target: 4, conv.flow: 192.990, real flow: 38.598
Edge conv.flow real flow
( 1, 2): 91.467, 18.293
( 2, 3): 91.528, 18.306
( 3, 4): 91.528, 18.306
( 9, 4): 101.447, 20.289
(10, 9): 101.447, 20.289
(11,10): 101.447, 20.289
(11, 2): 0.061, 0.012
(12,11): 101.508, 20.302

(12, 1): 91.406, 18.281
(12,15): 0.061, 0.012
(15, 1): 0.061, 0.012
* Source: 12, Target: 5, conv.flow: 192.990, real flow: 38.598

| Edge | conv.flow | real flow |
|---|---|---|
| ( 1, 2): | 78.851, | 15.770 |
| ( 2, 3): | 78.470, | 15.694 |
| ( 3, 4): | 78.470, | 15.694 |
| ( 4, 5): | 157.138, | 31.428 |
| ( 6, 5): | 35.837, | 7.167 |
| ( 7, 6): | 35.837, | 7.167 |
| ( 8, 7): | 35.304, | 7.061 |
| ( 2, 7): | 0.533, | 0.107 |
| ( 1, 8): | 35.304, | 7.061 |
| ( 9, 4): | 78.668, | 15.734 |
| (10, 9): | 78.668, | 15.734 |
| (11,10): | 78.668, | 15.734 |
| (11, 2): | 0.152, | 0.030 |
| (12,11): | 78.820, | 15.764 |
| (12, 1): | 109.538, | 21.908 |
| (12,15): | 4.617, | 0.923 |
| (15, 1): | 4.617, | 0.923 |

* Source: 12, Target: 9, conv.flow: 96.495, real flow: 19.299

| Edge | conv.flow | real flow |
|---|---|---|
| (10, 9): | 0.366, | 0.073 |
| (13, 9): | 96.122, | 19.224 |
| (11,10): | 0.366, | 0.073 |
| (12,11): | 0.366, | 0.073 |
| (12,15): | 96.122, | 19.224 |
| (14,13): | 96.122, | 19.224 |
| (15,14): | 96.122, | 19.224 |

*** Commodity type: 3
* Source: 12, Target:13, conv.flow: 192.990, real flow: 19.299

| Edge | conv.flow | real flow |
|---|---|---|
| ( 7, 6): | 192.975, | 19.298 |
| ( 8, 7): | 192.975, | 19.298 |
| ( 6, 3): | 192.975, | 19.298 |
| ( 1, 8): | 192.975, | 19.298 |
| ( 3,10): | 192.975, | 19.298 |
| (10,13): | 192.975, | 19.298 |
| (12, 1): | 166.402, | 16.640 |
| (12,15): | 26.573, | 2.657 |
| (15, 1): | 26.573, | 2.657 |

* Source: 12, Target:16, conv.flow: 192.990, real flow: 19.299

| Edge | conv.flow | real flow |
|---|---|---|
| ( 8,16): | 12.936, | 1.294 |
| ( 1, 8): | 12.936, | 1.294 |
| (12, 1): | 12.936, | 1.294 |
| (12,15): | 180.039, | 18.004 |
| (15,16): | 180.039, | 18.004 |

* Source: 13, Target:16, conv.flow: 192.990, real flow: 19.299

| Edge | conv.flow | real flow |
|---|---|---|
| ( 6, 7): | 192.975, | 19.298 |
| ( 7, 8): | 192.975, | 19.298 |
| ( 8,16): | 192.975, | 19.298 |
| ( 3, 6): | 192.975, | 19.298 |
| (10, 3): | 192.975, | 19.298 |
| (13,10): | 192.975, | 19.298 |

*** Commodity type: 4
* Source: 13, Target:16, conv.flow: 154.392, real flow: 7.720

| Edge | conv.flow | real flow |
|---|---|---|
| ( 6, 7): | 154.380, | 7.719 |
| ( 7, 8): | 154.380, | 7.719 |
| ( 8,16): | 154.380, | 7.719 |
| ( 3, 6): | 154.380, | 7.719 |
| (10, 3): | 154.380, | 7.719 |
| (13,10): | 154.380, | 7.719 |

## V. CONCLUSIONS

The contribution use the algorithm finding shortest path to install the general method determining the maximal concuret flow on the multicost multicommodity extended network developed in the work [16]. The algorithm was installed in the language C and has given reliable tests. On the basis of results of this work, further problems as maximal concurent limited flows or optimal maximal concurent flows will be studied.

## 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.* 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, 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 multicost 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 multicomodity multicost network,* EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, 12.2017, Volume 4, Issue 11, pp 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, pp 79-93.

[15]. Ho Van Hung, Tran Quoc Chien, *Implement and Test Algorithm finding Maximal Flow Limited Cost in extended multicomodity multicost network,* The International Journal of Computer Techniques(IJCT), Volume 6 Issue 3, May – June 2019, pp 1-9.

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