American Journal of Applied Mathematics
2020; 8(3): 74-82
http://www.sciencepublishinggroup.com/j/ajam
doi: 10.11648/j.ajam.20200803.11
ISSN: 2330-0043 (Print); ISSN: 2330-006X (Online)

Maximal Concurent Limited Cost Flow Problems 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

Email address:
hovanhung@qnamuni.edu.vn (Ho Van Hung), hungqcc@gmail.com (Ho Van Hung)

[*]Corresponding author

To cite this article:
Ho Van Hung, Tran Quoc Chien. Maximal Concurent Limited Cost Flow Problems on Extended Multi-commodity Multi-cost Network. *American Journal of Applied Mathematics*. Vol. 8, No. 3, 2020, pp. 74-82. doi: 10.11648/j.ajam.20200803.11

Received: April 1, 2020; **Accepted**: April 20, 2020; **Published**: April 29, 2020

Abstract: Graphs are excellent mathematical tools applied in many fields such as transportation, communication, informatics, economy,…. 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 right, turn left or go straight, even some directions are forbidden. Furthermore, on a network, there are many types of commodities, each of which are at different costs. Types of commodities share the capacity of edges and vertexes. Therefore, it is necessary to study a network with multiple commodities at multiple costs. The article builds a model of extended multi-commodity multi-cost network in order to modelise practical problems more exactly and effectively. The maximal concurent multi-commodity multi-cost flow limited cost problems, that are modelized by implicit linear programming problems. On the basis of duality theory in linear programming, an effective polynomial approximation algorithm is developed.

Keywords: Network, Graph, Multi-cost Multi-commodity Flow, Linear Optimization, Approximation

1. Introduction

Flows on networks are excelent mathematical means used in many applications as communication, transportation, economy, informatics, ….. So far, 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, the weight at one 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. The idea of using duality theory of linear programming to solve these problems is motivated by the work [1]. Paper [2] proposes switching cost only for directed graphs. Therefore, it is necessary to build an extended mixed network model in order to apply more accurate and effective modeling of practical problems. Multi-commodity flow in traditional network problems have been studied in the papers [3-11]. Multi-commodity singlecost flow problems in extended transport networks are studied in the papers [12-22]. The papers [23, 24] study maximal multi-commodity multi-cost flow problems. The papers [25, 26] study maximal multi-commodity multi-cost flow limited cost problems. The papers [27] and [28] study maximal concurent flow problems on extended multi-commodity multi-cost networks.

2. Multi-commodity Flows in Extended Multi-commodity Multi-cost Network

Given mixed graph $G = (V, E)$ with node set V and edge set E. The edges may be directed or undirected. The symbol E_v is

the set of edges incident the node $v \in V$. There are many types of commodities 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 commodities on the same edge, but reverse directions, share the capacity of the edge.

The symbol r is the number of commodity types, $q_i > 0$ is the coefficient of conversion of commodity type i, $i = 1, 2, \ldots, r$.

We define the following functions:

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

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

Node circulating capacity function $cv: V \to R^*$, where $cv(u)$ is the circulating capacity of the vertex $u \in V$.

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

The tuple (V, E, ce, ze, cv, zv) is called an *extended network.*

Edge cost function i, $i = 1, 2, \ldots, r$, $be_i: E \to R^*$, where $be_i(e)$ is the cost of circulating the edge $e \in E$ a converted unit of commodity of type i. Note that with 2-way paths, the cost of each direction may be different.

Node switch cost function i, $i = 1, 2, \ldots, r$, $bv_i: V \times E \times E \to R^*$, *where* $bv_i(v,e,e')$ is the cost of transferring a converted unit of commodity of type i from edge $e \in E_v$ through $v \in V$ to edge $e' \in E_v$.

The set $(V, E, ce, ze, cv, zv, \{be_i, bv_i, q_i | i = 1..r\})$ is called *an extended multi-cost multi-commodity network*.

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

Let p be the path from vertex u to vertex v through edges e_j, $j = 1, 2, \ldots, (h+1)$, and vertices u_j, $j = 1, 2, \ldots, h$ as follows

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

The cost of passing a converted unit of commodity of kind i, $i = 1, 2, \ldots, r$, through the path p, is denoted by the symbol $b_i(p)$, and calculated by the following formula:

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

Given a multi-cost multi-commodity network $G=(V, E, ce, ze, cv, zv, \{be_i, bv_i, q_i | i = 1, 2, .., r\})$. Assume, for each commodity type i, $i = 1, 2, \ldots, r$, there are k_i source-target pairs $(s_{i,j}, t_{i,j})$, $j = 1, 2, \ldots, k_i$, each pair assigned a quantity of comodity of type i, that is necessary to move from source node $s_{i,j}$ to target node $t_{i,j}$.

Denote $P_{i,j}$ 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, $i = 1, 2, \ldots, r$, $j = 1, 2, \ldots, k_i$. Set

$$P_i = \bigcup_{j=1}^{k_i} P_{i,j}, \ \forall i = 1, \ldots, r \quad (2)$$

For each path $p \in P_{i,j}$, $i = 1, 2, \ldots, r, j = 1, 2, \ldots, k_i$, denote $x_{i,j}(p)$ the flow of converted commodity of type i from the source node $s_{i,j}$ to the target node $t_{i,j}$ along the path p.

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

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

A set

$$F = \{x_{i,j}(p) \mid p \in P_{i,j}, i = 1, 2, \ldots, r, j = 1, 2, \ldots, k_i\} \quad (3)$$

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

$$\sum_{i=1}^{r} \sum_{j=1}^{k_i} \sum_{p \in P_{i,e}} x_{i,j}(p) \le ce(e).ze(e), \ \forall e \in E \quad (4)$$

and the *vertex capacity* constraints:

$$\sum_{i=1}^{r} \sum_{j=1}^{k_i} \sum_{p \in P_{i,v}} x_{i,j}(p) \le cv(v).zv(v), \ \forall v \in V \quad (5)$$

The expressions

$$fv_{i,j} = \sum_{p \in P_{i,j}} x_{i,j}(p), \ i = 1, 2, \ldots, r, j = 1, 2, \ldots, k_i \quad (6)$$

are called the flow value of commodity kind i of the source-target pair $(s_{i,j}, t_{i,j})$ of the multi-commodity flow F.

The expresstions

$$fv_i = \sum_{j=1}^{k_i} fv_{i,j}, \ i = 1..r \quad (7)$$

are called the flow value of commodity kind i of the multi-commodity flow F.

The expresstion

$$fv = \sum_{i=1}^{r} fv_i \quad (8)$$

is called the flow value of the multi-commodity flow F.

3. Maximal Concurent Limited Cost Multi-commodity Multi-cost Flow Problems

Given an extended linear multi-commodity multi-cost network $G=(V, E, ce, ze, cv, zv, \{be_i, bv_i, q_i | i = 1, 2, \ldots, r\})$. Assume, for each commodity type i, $i = 1, 2, \ldots, r$, there are k_i

source-target pairs $(s_{i,j}, t_{i,j})$, j=1, 2, …, 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 $t_{i,j}$. Given a limited cost B.

The mission of the problem is to find a maximum number λ such that there exists a flow converting $\lambda.D_{i,j}$ unit of commodity kind i, i=1, 2, …, r, from source vertex $s_{i,j}$ to target vertex $t_{i,j}$, $\forall j$ = 1, 2, …, k_i, and the total cost doex not exceed the limited cost B.

Set

$$\left.\begin{aligned}
&\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{p\in P_{i,v}} x_{i,j}(p) \le \mathrm{ce}(e).ze(e), \forall e\in E\\
&\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{p\in P_{i,v}} x_{i,j}(p) \le \mathrm{cv}(v).zv(v), \forall v\in V\\
&\sum_{p\in P_{i,j}} x_{i,j}(p) \ge \lambda.d_{i,j}, \forall j=1,2,...,k_i\\
&\sum_{i=1}^{r}\sum_{j=1}^{k_i}\sum_{p\in P_{i,j}} x_{i,j}(p).b_i(p) \le B, \lambda\ge 0, x_{i,j}(p)\ge 0, \forall j=1,2,...,r, \forall j=1,2,...,k_i, \forall p\in P_{i,j}
\end{aligned}\right\}(P)$$

The dual linear programming problem of (P), called (D), is constructed as follows: each edge $e\in E$ is assigned a dual variable $le(e)$, each vetex $v\in V$ is assigned a dual variable $lv(v)$, each requirement d_{ij} is assigned a dual variable z_{ij}, $\forall i$=1, 2, …, r, $\forall j$=1, 2,..., k_i, and the cost constraint is assigned a dual variable φ. The problem (D) states as following:

$$\left.\begin{aligned}
&D(le,lv)=\sum_{e\in E} ce(e).ze(e).le(e)+\sum_{e\in V} cv(v).zv(v).lv(v)+B.\varphi\to\min\\
&\sum_{e\in p} le(e)+\sum_{v\in p} lv(v)\ge z_{i,j}, \forall i=1,2,...,r,\ \ \forall j=1,2,...,k_i, \forall i=P_{i,j}\\
&\sum_{i=1}^{r}\sum_{j=1}^{k_i} d_{i,j}.z_{i,j}\ge 1, le(e)\ge 0, \forall e\in E, lv(v)\ge 0, \forall v\in V, z_{i,j}\ge 0, \forall i=1,2,...,r,\ \ \forall j=1,2,...,k_i,\ \ \varphi\ge 0
\end{aligned}\right\}(\mathrm{D})$$

Now, given $p\in P$ a path from vertex u to vertex v through edges e_i, i=1, 2, …, (h+1), and vertex u_i, i=1, 2, …, h, as follows

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

We define the path length of p, denoted by $length_i(p)$, depending on the variables $le(e)$, $lv(v)$ by the following formula:

$$length_i(p) = \sum_{j=1}^{h+1} le(e_j) + \sum_{j=1}^{h} lv(u_j) + b_i(p).\varphi \quad (9)$$

Denote $dist_{i,j}(le,lv,\varphi)$ the shortest path length from $s_{i,j}$ to $t_{i,j}$ calculated by function $length_i(p)$, $\forall i$=1, 2, …, r, $\forall j$=1, 2,..., k_i. Set

$$\alpha(le,lv,\varphi) = \sum_{i=1}^{r}\sum_{j=1}^{k_i} d_{i,j}.dist_{i,j}(le,lv,\varphi). \quad (10)$$

Consider the problem (D_α):

$$d_{i,j} = q_i.D_{i,j}, \forall i=1, 2, \ldots, r, \forall j=1, 2,..., k_i$$

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

$$\lambda \to \max$$

Satisfies

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

Lemma 3.1. The problem (D) is equivalent to the problem (D_α) such that their optimal value are equal and the optimal solution of one problem derives the optimal solution of the other problem and vice versa.

Prove

Denote $min(D)$ and $min(D_\alpha)$, respectively, the optimal values of the problem (D) and the problem (D_α). Given functions $le: E\to R^*$, $lv: V\to R^*$. Set

$le'(e) = le(e)/\alpha(le,lv,\varphi)\ \forall e\in E$, $lv'(v) = lv(v)/\alpha(le,lv,\varphi)\ \forall v\in V$,

$\varphi' = \varphi/\alpha(le,lv,\varphi)$ and $z'_{i,j} = dist_{i,j}(le',lv',\varphi')$

$= dist_{i,j}(le,lv,\varphi)/\alpha(le,lv,\varphi)$, $\forall i$=1, 2, …, r, $\forall j$=1, 2,..., k_i.

We have

$$\sum_{e\in p} le'(e) + \sum_{v\in p} lv'(v) + b_i(p).\varphi' \geq z'_{i,j},$$

$$\forall i=1, 2, \ldots, r, \forall j=1, 2,..., k_i, \forall p\in P_{i,j}$$

and

$$\sum_{i=1}^{r}\sum_{j=1}^{k_i} d_{i,j}.z'_{i,j} = \frac{1}{\alpha(le,lv,\varphi)}$$

$$\sum_{i=1}^{r}\sum_{j=1}^{k_i} d_{i,j}.dist_{i,j}(le,lv,\varphi) = 1$$

So $(le',lv',\varphi',z'_{i,j})$ is an accepted solution of (D) and $D(le',lv',\varphi') = \dfrac{D(le,lv,\varphi)}{\alpha(le,lv,\varphi)}$. Hence,

$$min(D) \leq min(D_\alpha) \quad (11)$$

On the contrary, let $(le,lv,\varphi, z_{i,j})$ be an admitted solution of (D). Then, we have:

$$z_{i,j} \leq dist_{i,j}(le,lv,\varphi), \forall i=1, 2, \ldots, r, \forall j=1, 2,..., k_i$$

$$\Rightarrow \alpha(le,lv,\varphi) = \sum_{i=1}^{r}\sum_{j=1}^{k_i} d_{i,j}.dist_{i,j}(le,lv,\varphi) \geq \sum_{i=1}^{r}\sum_{j=1}^{k_i} d_{i,j}.z_{i,j} \geq 1$$

It follows $\dfrac{D(le,lv,\varphi)}{\alpha(le,lv,\varphi)} \leq D(le,lv,\varphi)$. Hence,

$$min(D) \geq min(D_\alpha) \quad (12)$$

From (11) and (12) it follows $min(D) = min(D_\alpha)$.

Next, if $(le,lv,\varphi, z_{i,j})$ is an optimal solution of the problem (D_α), then $(le',lv',\varphi',z'_{i,j})$ where

$le'(e) = le(e) / \alpha(le,lv,\varphi)\ \forall e\in E,$

$lv'(v) = lv(v) / \alpha(le,lv,\varphi)\ \forall v\in V,$

$\varphi' = \varphi / \alpha(le,lv,\varphi)$ and $z'_{i,j} = dist_{i,j}(le',lv',\varphi')$

$= dist_{i,j}(le,lv,\varphi) / \alpha(le,lv,\varphi), \forall i=1, 2, \ldots, r, \forall j=1, 2,..., k_i,$

is an optimal solution of problem (D).

Conversely, if $(le,lv,\varphi, z_{i,j})$ is an optimal solution of the problem (D), then $(le,lv,\varphi, z_{i,j})$ is an optimal solution of the problem (D_α).

4. Algorithm

Ideas

Algorithm is implemented through several phases. Each phase consists of k loops, $k=k_1+k_2+ \ldots +k_r$. At the loop $[i,j]$, $\forall i=1, 2, \ldots, r, \forall j=1, 2,..., k_i$, of a phase t we move $d_{i,j}$ converted units of commodity of kind i from source vertex $s_{i,j}$ to target vertex $t_{i,j}$. This move is implemented in several steps.

Algorithm

◊ *Input*: Extended multi-cost multi-commodity network $G=(V,E, ce, ze, cv, zv, \{be_i, bv_i, q_i | i=1, 2, \ldots, r\})$. Assume, for each commodity of kind i, $i=1, 2, \ldots, r$, there are k_i source-target pairs $(s_{i,j}, t_{i,j})$, $j=1, 2, \ldots, k_i$, each pair assigned a quantity of commodity D_{ij} of kind i, that is necessary to move from source vertex $s_{i,j}$ to target vertex $t_{i,j}$. Given a limited cost B and an approximation ratio ω. Let denote $n=|V|$, $m=|E|$.

◊ *Output*:

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

$$F = \{f_{i,j}(e) \mid e\in E, i=1, 2, \ldots, r, j=1, 2, \ldots, k_i\}$$

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

$$rF = \{rf_{i,j}(e) \mid e\in E, i=1, 2, \ldots, r, j=1, 2, \ldots, k_i\}$$

Total cost $B_f \leq B$, maximal concurent coefficient λ.

◊ *Procedure*

//Initialization: Calculate ε and δ

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

$d_{i,j} = D_{i,j}.q_i, \forall i=1, 2, \ldots, r, j=1, 2, \ldots, k_i;$

$le(e) = \delta/(ce(e)ze(e)), \forall e\in E;$

$lv(v) = \delta/(cv(v)zv(v)); \forall v\in V;$

$x_{i,j}(e) = 0, \forall i=1, 2, \ldots, r, j=1, 2, \ldots, k_i, \forall e\in E;$

$\varphi = \delta/B;$

$$D(le,lv,\varphi) = \sum_{e\in E} ce(e).ze(e).le(e) + \sum_{v\in V} cv(v).zv(v).lv(v) + B.\varphi = (m+n+1)\delta;$$

$t = 0$; // phase counts in iteration……while $(D(t) < 1)$

//Denote le_t, lv_t, $D(t)$, $\alpha(t)$ the corresponding quantities after phase t.

$D(0) = (m+n+1)\delta$; $le_0(e) = le(e); \forall e\in E$, $lv_0(v) = lv(v); \forall v\in V$, $\varphi_0 = \varphi$;

do // phases

{

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

for (j=1; j<=k_i; j++) // loops

{

$d'_{i,j} = d_{i,j}$;

do // steps

{

Find the shortest path p from $s_{i,j}$ to $t_{i,j}$ calculated by function $length_i(.)$. Note that the path p must be valid for commodity of type i, i.e., not containing the edge with edge cost ∞ or the node with the switch cost ∞, $dist_{i,j}(le,lv,\varphi)$ is the shortest path p from $s_{i,j}$ to $t_{i,j}$ calculated by function $length_i(p)$, $be_i(p)$ is the cost of a converted unit of commodity type i on the path p. Set $c = \min\{\min\{ce(e).ze(e) | e\in p\}, \min\{cv(v).zv(v) | v\in p\}, d'_{i,j}\}$.

$B' = c.b_i(p)$;

f (B'>B)

{
$c = c.B / B'$; $B' = B$;
}
//Flow adjustments:
$\forall e \in p$, $x_{i,j}(e) = x_{i,j}(e) + c$;
$le(e) = le(e).(1+\varepsilon.c/(ce(e).ze(e)))$;
$\forall v \in p$, $lv(v) = lv(v).(1+\varepsilon.c/(cv(v).zv(v)))$;
$\varphi = \varphi.(1+\varepsilon.B'/B)$;
$B_f = B_f + B'$;
$d'_{i,j} = d'_{i,j} - c$;
$D(le,lv,\varphi) = D(le,lv,\varphi) + \varepsilon.c.dist_{i,j}(le,lv,\varphi)$;
} while ($d'_{i,j} > 0$);
}// for … for …
$t = t + 1$;
$D(t) = D(le,lv,\varphi)$;
$le_t(e) = le(e)$, $\forall e \in E$;
$lv_t(v) = lv(v)$, $\forall v \in V$;
if ($D(t) < 1$)
for (i=1; $i <= r$; i++)
for (j=1; $j <= k_i$; j++)
for ($e \in E$)
$f_{i,j}(e) = x_{i,j}(e)$;// $f_{i,j}(e)$ denote optimal flow
} while ($D(t) < 1$)
//Modifying the resulting flows F
$B_f = B_f / \log_{1+\varepsilon} \frac{1}{\delta}$;
for (i=1; $i <= r$; i++)
for (j=1; $j <= k_i$; j++)
for ($e \in E$)
$f_{i,j}(e) = f_{i,j}(e) / \log_{1+\varepsilon} \frac{1}{\delta}$;
//Modifying flows on scalar edge
for (i=1; $i<=r$;i++)
for (j=1; $j<=k_i$;j++)
for $e \in E$, e scalar
if $f_{i,j}(e) >= f_{i,j}(e')$// e' is the opposite of the direction e
{
$B_f = B_f - f_{i,j}(e')(be_i(e)+be_i(e'))$;
$f_{i,j}(e) = f_{i,j}(e) - f_{i,j}(e')$;
$f_{i,j}(e') = 0$;
}
else
{
$B_f = B_f - f_{i,j}(e)(be_i(e)+be_i(e'))$;
$f_{i,j}(e') = f_{i,j}(e') - f_{i,j}(e)$; $f_{i,j}(e) = 0$;
}
//Convert the flow $f_{i,j}(e)$ to the actual flow $rf_{i,j}(e)$ by dividing the conversion flow by the conversion coefficient
$rf_{i,j}(e) = f_{i,j}(e)/q_i$, $\forall e \in E$, $\forall i$=1, 2, …, r, j=1, 2, …, k_i
// Maximum approximation coefficient
$\lambda = (t-1) / \log_{1+\varepsilon} \frac{1}{\delta}$;
//The End.

Proof of algorithm

◊ *Remarks:* In (t−1) phases of implementation of the above algorithm, $\forall i$=1, 2, …, r, j=1, 2, …, k_i, we have transferred $(t-1).d_{i,j}$ units of converting commodity type i from $s_{i,j}$ to $t_{i,j}$. However, the transferred flow may exceed the throughout capacity of the edges.

The following lemma resolves the above problem

Lemma 4.1. $\lambda > \dfrac{t-1}{\log_{1+\varepsilon} \dfrac{1}{\delta}}$. (13)

Proof. Consider any edge e. Initiationly,

$le(e)=\delta/(ce(e)ze(e))$, $\forall e \in E$, $lv(v)=\delta/(cv(v)zv(v))$, $\forall v \in V$,

$\varphi = \delta/B$.

After (t−1) phases implemented, we have $D(t-1) < 1$, it means

$$\sum_{e \in E} ce(e).ze(e).le_{t-1}(e) + \sum_{v \in V} cv(v).zv(v).lv_{t-1}(v) + B.\varphi_{t-1} < 1,$$

that follows

$le_{t-1}(e) < 1/(c(e).ze(e))$, $\forall e \in E$, $lv_{t-1}(v) < 1/(cv(v)zv(v))$, $\forall v \in V$.

Let $fe(e)$ be the sum of the converted units of commodities passing $e \in E$ and $fv(v)$ is the sum of the converted units of commodities passing $v \in V$ in (t−1) phases.

Consider $e \in E$. Suppose in the process of constructing fe(e) there is $ce(e)ze(e)$ units of the flow passing e through q steps, each step transfers g_s converted units of commodity, i.e. $\sum_s g_s = ce(e)ze(e)$.

Through each step, le(e) is increased by the factor $(1+\varepsilon.g_s/(ce(e)ze(e)))$. So, through q steps, le(e) is increased by the factor

$$\prod_s \left(1 + \varepsilon.g_s/(ce(e)ze(e)\right) > (1 + \varepsilon.\sum_s g_s/(ce(e)ze(e))) = (1+\varepsilon)$$

We see, for every edge $e \in E$, for each transfer of $ce(e)ze(e)$ converted units of commodities through e, $le(e)$ increases by at least one factor $(1+\varepsilon)$.

Similarly, for every node $v \in V$, for every $cv(v)zv(v)$ converted units of commodity passed v, $lv(v)$ increases by at least one factor $(1+\varepsilon)$.

On the other hand, the number of times to send $ce(e).ze(e)$ converted unit of commodity over each edge $e \in E$ is at least $fe(e)/(ce(e).ze(e))$ and the number of times to send $cv(v)zv(v)$ converted unit of commodity through each node $v \in V$ is at least $fv(v)/(cv(v)zv(v))$.

At this point, the edge and node functions will satisfy the following inequality:

$$le_{t-1}(e) \geq le_0(e).(1+\varepsilon)^{fe(e)/(ce(e).ze(e))}, \forall e \in E$$

and

$$lv_{t-1}(v) \geq lv_0(v).(1+\varepsilon)^{fv(v)/(ve(v).zv(v))}, \forall v \in V$$

Hence inferred

$$fe(e) \le ce(e).ze(e).\ \log_{1+\varepsilon}\frac{le_{t-1}(e)}{le_0(e)} <$$

$$ce(e).ze(e).\ \log_{1+\varepsilon}\frac{1/(ce(e).ze(e))}{\delta/(ce(e).ze(e))} =$$

$$ce(e).ze(e).\ \log_{1+\varepsilon}\frac{1}{\delta},\ \forall e \in E$$

and

$$fv(v) \le cv(v).zv(v).\ \log_{1+\varepsilon}\frac{lv_{t-1}(v)}{lv_0(v)} <$$

$$cv(v).zv(v).\ \log_{1+\varepsilon}\frac{1/(cv(v).zv(v))}{\delta/(cv(v).zv(v))} =$$

$$cv(v).zv(v).\ \log_{1+\varepsilon}\frac{1}{\delta},\ \forall v \in V.$$

Thus, divide fe(e) by $\log_{1+\varepsilon}\frac{1}{\delta}$, $\forall$e $\in$ E, that follows fv(v) is divided by $\log_{1+\varepsilon}\frac{1}{\delta}$, we receive accepted flows.

The above analysis shows that after (t–1) of phases, $\forall i$=1, 2, …, r, j=1, 2, …, k_i, we have moved (t–1).$d_{i,j}$ converted units from $s_{i,j}$ to $t_{i,j}$. However, in order for the flow to be accepted, we must divide the flow by $\log_{1+\varepsilon}\frac{1}{\delta}$. So, $\forall i$=1, 2, …, r, j=1, 2, …, k_i, we move $\frac{t-1}{\log_{1+\varepsilon}\frac{1}{\delta}}$. $d_{i,j}$ converted units of commodities from $s_{i,j}$ to $t_{i,j}$. Then, flow through edge e is no greater than $ce(e).ze(e)$, $\forall e \in E$, and flow through v is not greater than $cv(v).zv(v)$, $\forall v \in V$. So, we have

$$\lambda > \frac{t-1}{\log_{1+\varepsilon}\frac{1}{\delta}}. \qquad (14)$$

Lemma 4.2

Assume $\beta \ge 1$. The algorithm's found flow, after being divided by $\log_{1+\varepsilon}\frac{1}{\delta}$, is the concurent maximal flow, where $\frac{t-1}{\log_{1+\varepsilon}\frac{1}{\delta}}$ is the maximal coefficient with the approximation ratio $(1+\omega)$.

Proof

Since le and lv functions are incremental after each update,

$$D(q) \le D(q-1) + \varepsilon.\alpha(q), \forall q = 1, 2,..., t. \qquad (15)$$

Next, we have

$$\frac{D(q)}{\alpha(q)} \ge \beta \Rightarrow \alpha(q) \le \frac{D(q)}{\beta},\ \forall q = 1, 2,..., t. \qquad (16)$$

From (15) and (16) we receive:

$$D(q) \le D(q-1)+\varepsilon.D(q)/\beta,\ \forall q = 1, 2,..., t,$$

$$\Rightarrow D(q) \le \frac{D(q-1)}{1-\varepsilon/\beta} \le \frac{D(q-2)}{(1-\varepsilon/\beta)^2} \le .... \le \frac{D(0)}{(1-\varepsilon/\beta)^q} = \frac{(m+n+1).\delta}{(1-\varepsilon/\beta)^q},\ \forall q = 1, 2,..., t.$$

For $\beta \ge 1$, we have

$$D(q) \le \frac{(m+n+1).\delta}{(1-\varepsilon/\beta)}\left(1+\frac{\varepsilon}{\beta-\varepsilon}\right)^{q-1}\frac{(m+n+1).\delta}{(1-\varepsilon/\beta)}e^{\frac{\varepsilon(q-1)}{\beta-\varepsilon}}$$

$$\le \frac{(m+n+1).\delta}{(1-\varepsilon)}e^{\frac{\varepsilon(q-1)}{\beta-\varepsilon}},\ \forall q = 1, 2,..., t.$$

With regard to stop condition of algorithm $D(t) \ge 1$, we have

$$1 \le D(t) \le \frac{(m+n+1).\delta}{(1-\varepsilon)}e^{\frac{\varepsilon(t-1)}{\beta-\varepsilon}}$$

Hence

$$\frac{\beta}{t-1} \le \frac{\varepsilon}{(1-\varepsilon)\ln\frac{1-\varepsilon}{(m+n+1)\delta}} \qquad (17)$$

Set $\gamma=\frac{\beta}{t-1}\ \log_{1+\varepsilon}\frac{1}{\delta}$ and entail $\frac{\beta}{t-1}$ from (17), we have

$$\gamma < \frac{\varepsilon.\log_{1+\varepsilon}\frac{1}{\delta}}{(1-\varepsilon)\ln\frac{1-\varepsilon}{(m+n+1).\delta}} = \frac{\varepsilon}{(1-\varepsilon)\ln(1+\varepsilon)}\frac{\ln\frac{1}{\delta}}{\ln\frac{1-\varepsilon}{(m+n+1).\delta}}$$

For $\delta=\left(\frac{m+n+1}{1-\varepsilon}\right)^{-\frac{1}{\varepsilon}}$, we have $\frac{\ln\frac{1}{\delta}}{\ln\frac{1-\varepsilon}{(m+n+1).\delta}} = (1-\varepsilon)^{-1}$, and so

$$\gamma < \frac{\varepsilon}{(1-\varepsilon)^2\ln(1+\varepsilon)} \le \frac{\varepsilon}{(1-\varepsilon)^2(\varepsilon-\varepsilon^2/2)} \le (1-\varepsilon)^{-3}$$

On the other hand, by duality, we have $\gamma \ge 1$.

For $\varepsilon = 1-\sqrt[3]{\frac{1}{1+\omega}}$, we have $\gamma < (1+\omega)$. Then, the value $\lambda = \frac{t-1}{\log_{1+\varepsilon}\frac{1}{\delta}}$ is the maximal concurent coefficient with the aproximation ratio $(1+\omega)$.

Lemma 4.3

Assume $\beta<1$. Let $l>1$ such that $l.\beta>1$. Apply the algorithm to the requirements

$$d'_{i,j} = \frac{1}{l} d_{i,j}, \forall i=1, 2, \ldots, r, j=1, 2, \ldots, k_i.$$

The algorithm's found flow, after divide $\frac{t-1}{\log_{1+\varepsilon}\frac{1}{\delta}}$, is the maximal concurent flow of the original problem, where the maximal coefficient is $\frac{1}{l}.\frac{t-1}{\log_{1+\varepsilon}\frac{1}{\delta}}$ with the approximation ratio $(1+\omega)$.

Proof

According to Lemma 4.2, the algorithm's found flow, after dividing by $\log_{1+\varepsilon}\frac{1}{\delta}$, is the maximal concurent flow of the problem to the requirement $d'_{i,j} = \frac{1}{l} d_{i,j}, \forall i=1..r, j=1..k_i.$ and $\lambda = \frac{t-1}{\log_{1+\varepsilon}\frac{1}{\delta}}$ is the maximal coefficient with the aproximation ratio $(1+\omega)$.

Hence, $\frac{1}{l}\lambda = \frac{1}{l}\frac{t-1}{\log_{1+\varepsilon}\frac{1}{\delta}}$ is the maximal coefficient with the approximation ratio $(1+\omega)$ of the original problem.

Lemma 4.4. The total cost after $(t-1)$ phases does not exceed $B.\log_{1+\varepsilon}\frac{1}{\delta}$. That means, that after dividing the flows by $\log_{1+\varepsilon}\frac{1}{\delta}$, the total cost after does not exceed B.

Proof

We have $\varphi_0 = \delta/B$. After (t–1) phases we get $D(t-1) < 1$, t.e.

$$\sum_{e\in E} ce(e).ze(e).le_{t-1}(e) + \sum_{v\in V} cv(v).zv(v).lv_{t-1}(v) + B.\varphi_{t-1} < 1.$$

It follows $\varphi_{t-1} < 1/B$. Furthermore, for each transfer of flow such that the total cost is augmented by an amount B, φ increases by at least one factor $(1+\varepsilon)$.

Therefore, denoting x the times of incresing the total cost by B, after $(t-1)$ phases, we have $\varphi_0.(1+\varepsilon)^x \le \varphi_{t-1} \le 1/B$, suy ra $x \le \log_{1+\varepsilon}\frac{1}{\delta}$.

So, the total cost is $B.\log_{1+\varepsilon}\frac{1}{\delta}$. Hence, after dividing the flows by $\log_{1+\varepsilon}\frac{1}{\delta}$, the total cost after does not exceed B.

5. Algorithm Complexity

Theorem 5.1.

The algorithm's complexity is

$$O(\omega^{-2}.(cemax/dmax).(\chi+k).m.n^3.\log_2(m+n+1)),$$

where m is the number of edges, n is the number of vertices of the network, $k = k_1+\ldots+k_r$, $cemax = \max\{ce(e).ze(e) \mid e\in E\}$, $dmax = \max\{d_{i,j} \mid i=1,..,r, j=1,..., k_i\}$, and $\chi = \sum_{i=1}^{r}\sum_{j=1}^{k_i} d_{i,j}/cmin$, with $cmin=\min\{cemin, cvmin\}$, $cemin=\min\{ce(e).ze(e) \mid e\in E\}$ and $cvmin=\min\{cv(v).zv(v) \mid v\in V\}$.

Proof

First, we find the number of phases the algorithm has taken. According to the proof of lemma 4.2 above and for $\beta = \lambda$, we have

$$1 \le \gamma = \frac{\beta}{t-1}\log_{1+\varepsilon}\frac{1}{\delta}$$

$$\Rightarrow t \le 1+\beta.\log_{1+\varepsilon}\frac{1}{\delta} \Rightarrow t = \log_{1+\varepsilon}\frac{1}{\delta}.O(\beta),$$

where ε and δ depend on ω. Besides, t depends on β.

Further, denote *imax*, *jmax* indexes satisfying

$$dmax = \max\{d_{i,j} \mid i=1, 2, \ldots, r, j=1, 2, \ldots, k_i\} = d_{imax,jmax}.$$

From the constraint of the problem (P)

$$\sum_{p\in P_{i,j}} x_{i,j}(p) \ge \lambda.d_{ij}, \forall i=1,2,..,r, j=1,2,...,k_i$$

we have

$$\lambda \le \sum_{p\in P_{i\max,j\max}} x_{i\max,j\max}(p) / d_{imax,jmax} \le \sum_{e\in E_{s_{i\max,j\max}}} c(e).z(e) / dmax$$

$$\le \left|E_{s_{i\max,j\max}}\right|.cemax / dmax \le m.cemax/dmax$$

that implies

$$\beta = \lambda \le m.cemax/dmax$$

$$\Rightarrow t = \log_{1+\varepsilon}\frac{1}{\delta}.O(m.cemax/dmax),$$

Replacing $\delta = \left(\frac{m+n+1}{1-\varepsilon}\right)^{-\frac{1}{\varepsilon}}$ to the above expresion, we have

$$t = \frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m+n+1}{1-\varepsilon}.O(m.cemax/dmax). \quad (18)$$

On the other hand, each phase implements k loop, so the loop number is $k.t$. Consider the loop transfering $d_{i,j}$ converted units of commodities from $s_{i,j}$ to $t_{i,j}$, $i=1, 2, \ldots, r, j=1, 2, \ldots, k_i$. Since *cmin* is the the minimal capacity of edges and nodes, the

number of steps required to execute the loop is not exceeded ($d_{i,j}/cmin$+1). The main procedure in each step, finding the shortest path from $s_{i,j}$ to $t_{i,j}$, has a complexity of n^3 [8]. So the complexity of the loop is ($d_{i,j}/cmin$+1).n^3. Hence the complexity of each phase is:

$$\sum_{i=1}^{r}\sum_{j=1}^{k_i}(d_{i,j}/cmin+1).n^3=(\sum_{i=1}^{r}\sum_{j=1}^{k_i}d_{i,j}/cmin+1).n^3=(\chi+k).n^3.$$

So the algorithm's complexity is

$$t.(\chi+k).n^3=\frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m+n+1}{1-\varepsilon}.O(m.cemax/dmax).(\chi+k).n^3$$

$$= O(\frac{1}{\varepsilon}\log_{1+\varepsilon}\frac{m+n+1}{1-\varepsilon}.m.(cemax/dmax).(\chi+k).n^3)$$

$$= O(\omega^{-2}.(cemax/dmax).(\chi+k).m.n^3.\log_2(m+n+1)),$$

$$\text{for } \varepsilon = 1-\sqrt[3]{\frac{1}{1+\omega}} = O(\omega) \text{ và } \log_2(1+\varepsilon)\approx\varepsilon.$$

6. Conclusions

The contribution defines the maximal concurent limited cost flow problems on extended multi-commodity multi-cost networks, that can be more exactly and effectively applied to model many practical problems. The maximal concurent limited cost flow problems are modeled as implicit linear optimization problems. On the base of dual theory in linear optimization, an effective polynomial approximate algorithm is developed. Correctness and algorithm complexity are proved.

References

[1] Naveen Garg and 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 and 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] Ellis L. Johnson, George L. Nemhauser; Joel S. Sokol, and Pamela H. Vance, "Shortest Paths and Multi-Commodity Network Flows," A Thesis Presented to the Academic Faculty, pp. 41-73, 2003.

[4] Xiaolong Ma and Jie Zhou, "An extended shorted path problem with switch cost between arcs," Proceedings of the international multiconference of engineers and computer scientists, Hong Kong, IMECS, 2008.

[5] L. K. Fleischer, "Approximating fractional Multi-Commodity flow independent of the number of commodities," SIAM J. Discrete Math., vol. 13, no. 4, 2000.

[6] G. Karakostas, "Faster approximation schemes for fractional Multi-Commodity flow problems," In Proceedings, ACMSIAM Symposium on Discrete Algorithms, vol. 4, no. 1, 2002.

[7] Aleksander, "Faster Approximation Schemes for Fractional Multi-Commodity Flow Problems via Dynamic Graph Algorithms," Massachusetts Institute of Technology, 2009.

[8] Tran Quoc Chien, "Linear multi-channel traffic network", Ministry of Science and Technology, code B2010DN-03-52.

[9] Tran Quoc Chien and 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.

[10] 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.

[11] 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.

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

[13] Tran Quoc Chien; Nguyen Mau Tue; and 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), Viet Nam, pp. 522-527.

[14] 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.

[15] Tran Ngoc Viet; Tran Quoc Chien; and 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.

[16] Viet Tran Ngoc; Chien Tran Quoc; and 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.

[17] 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.

[18] Ziliaskopoulos, A. K, "A linear rogramming model for the single destination system optimum dynamic traffic assignment problem," Transportation Science, vol. 34, no. 1, 2000.

[19] Nagarney A, "Mathematical Models of Transportation and Networks," Mathematical Models in Economics, 2007.

[20] Schulz, A. S., N. E. Stier-Moses, "On the performance of user equilibria in traffic networks," Proc. 14th Annual ACM-SIAM Sympos on Discrete Algorithms (SODA). SIAM, Philadelphia, PA, 2003.

[21] 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.

[22] 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, pp. 31-39.

[23] Tran Quoc Chien, Ho Van Hung, "Extended linear Multi-Commodity multi-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, pp. 385-395.

[24] 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, 12.2017, Volume 4, Issue 11, pp. 1-6.

[25] 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. (SCOPUS).

[26] 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, May – June 2019, pp. 1-9.

[27] 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 (IJMNCT), Vol. 9, No. 1, February 2019, pp 1-14.

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