A COMBINATORIAL ALGORITHM FOR THE MULTI-COMMODITY FLOW PROBLEM

Pengfei Liu
Building No.1, Zhonguancun Road
Haidian District
Beijing, China
pengfeil89@foxmail.com

June 3, 2020

ABSTRACT

This paper researches combinatorial algorithm for the multi-commodity flow problem. We relax the capacity constraints and introduce a *penalty function* h for each arc. If the flow exceeds the capacity on arc a, arc a would have a penalty cost. Based on the *penalty function* h, a new conception , *equilibrium pseudo-flow*, is introduced. If the equilibrium pseudo-flow is a nonzero-equilibrium pseudo-flow, there exists no feasible solution for the multi-commodity flow problem; if the equilibrium pseudo-flow is a zero-equilibrium pseudo-flow, there exists feasible solution for the multi-commodity flow problem and the zero-equilibrium pseudo-flow is the feasible solution. Then a *non-linear* description of the multi-commodity flow problem is given, whose solution is equilibrium pseudo-flow. And a combinatorial algorithm(Frank-Wolfe algorithm) is designed to obtain equilibrium pseudo-flow. Besides, the content in this paper can be easily generalized to minimum cost multi-commodity flow problem.

Keywords combinatorial algorithm · multi-commodity flow

1 Introduction

The *multi-commodity flow problem* (MFP) is the problem of designing flow of several different commodities through a common network with arc capacities. Given a directed graph $G(V, A)$, a capacity function $u : A \to Q^+$, K origin-destination pairs of nodes , defined by $K_k = (s_k, t_k, d_k)$ where s_k and t_k are the origin and destination of commodity k, and d_k is the demand. The flow of commodity k along arc (i, j) is f^k_{ij}. The objective is to obtain an assignment of flow which satisfies the demand for each commodity without violating the capacity constraints. The constraints can be summarized as follows:

$$\begin{aligned} &\sum_{k \in K} f^k_{ij} \leq u_{ij}, \forall (i,j) \in A \\ &\sum_{j \in \delta^+(i)} f^k_{ij} - \sum_{j \in \delta^-(i)} f^k_{ji} = \begin{cases} d_k, \text{ if } i = s_k \\ -d_k, \text{ if } i = t_k \\ 0, \text{ if } i \in V - \{s_k, t_k\} \end{cases} \\ &f^k_{ij} \geq 0, \forall k \in K, (i,j) \in A \end{aligned} \tag{1}$$

where $\delta^+(i) = \{j|(i,j) \in A\}, \delta^-(i) = \{j|(j,i) \in A\}$. In this paper, we assume $s_k \neq t_k$. The first expression is capacity constraint. The second is flow conservation constraint and the last is non-negative constraint.

Multicommodity flow problems have attracted great attention since the publication of the works of [Ford and Fulkerson(1962)] and [Hu(1963)]. [Assad(1978)] gives a comprehensive survey, which includes decomposition, partitioning, compact inverse methods, and primal-dual algorithms. Although there are many combinatorial algorithms for single-commodity flow models like Ford-Fulkerson algorithm([Ford and Fulkerson(1962)]), Edmonds-Karp algorithm([Edmonds and Karp(1972)]), Dinic's algorithm ([Dinic(1970)]) and push-relabel algorithm([Goldberg and Tarjan(1988)]), there is no known combinatorial algorithm for multi-commodity flow problem. It is well known that MFP can be solved in polynomial time using linear programming. However, up to date, there is no other way to solve the problem precisely without using linear programming. In this paper, we would *give the first combinatorial algorithm* for the multi-commodity flow problem.

1.1 Our contribution

A new conception, *equilibrium pseudo-flow*, is introduced and a *non-linear* description of the multi-commodity flow problem is given whose solution is equilibrium pseudo-flow. Besides, we give a combinatorial algorithm to obtain equilibrium pseudo-flow for the multi-commodity flow problem. To the best of our knowledge, this is the first algorithm to obtain the solution of the multi-commodity flow problem without using linear programming.

2 Equilibrium Pseudo-flow

Unlike other methods, our combinatorial algorithm *does not maintain the capacity constraints* throughout the execution. The algorithm, however, maintains a *pseudo-flow*, which is a function $\mathbf{f} : K \times V \times V \to \mathbb{R}^+$ that just satisfies the flow conservation on every node. That is, a pseudo-flow $\mathbf{f}$ is a feasible solution of Expression (2).

$$\begin{aligned}&\sum_{j\in\delta^+(i)} f_{ij}^k - \sum_{j\in\delta^-(i)} f_{ji}^k = \begin{cases} d_k, \text{ if } i = s_k \\ -d_k, \text{ if } i = t_k \\ 0, \text{ if } i \in V - \{s_k, t_k\}\end{cases} \\ &f_{ij}^k \geq 0, \forall k \in K, (i,j) \in A\end{aligned} \tag{2}$$

We introduce a *penalty function* h for each arc $(i, j) \in A$, which is defined as

$$h(f_{ij}) = \begin{cases} 0 & if\ f_{ij} \leq u_{ij} \\ f_{ij} - u_{ij} & if\ f_{ij} > u_{ij}\end{cases} \tag{3}$$

If the flow on an arc (i, j) is less than the capacity, the penalty of arc (i, j) is zero. Otherwise, the penalty of arc (i, j) is the amount by which the flow exceeds the capacity.

Intuitively, the greater the $h(f_{ij})$ is, the more 'congested' the arc (i, j) is. By using $\{h(f_{ij}), \forall(i, j) \in A\}$ as weights for the arcs, the longer the path $p_{s_k t_k}$ is, the more 'congested' the path $p_{s_k t_k}$ is, where $p_{s_k t_k}$ is a path connecting s_k and t_k. In fact, for a pair(s_k, t_k), our algorithm iteratively adjusts the flow to the shortest paths until all the used paths have equal length.

We introduce the concept of *equilibrium pseudo-flow* here, which is the key to the combinatorial algorithm.

Definition 1 *By using \{ $h(f_{ij}), \forall(i, j) \in A$\} as weights for all the arcs, a pseudo-flow* $\mathbf{f}$ *is called an* equilibrium pseudo-flow *if it satisfies the following conditions:*

(i) for any given pair (s_k, t_k), all used paths connecting s_k and t_k have equal and minimum length;

(ii) for any given pair (s_k, t_k), all unused paths connecting s_k and t_k have greater or equal length;

where a path p connecting s_k and t_k is called *used* if there exists $s_k - t_k$ flow on path p, otherwise it is called *unused*. The conditions above are also called *equilibrium conditions*. Note that the conception above is similar to 'user equilibrium'([Wardrop (1953)]), which is a sound and simple behavioral principle to describe the spreading of trips.

Definition 2 *An equilibrium pseudo-flow* $\mathbf{f}$ *is called zero-equilibrium pseudo-flow if \{ $h(f_{ij}) = 0, \forall(i, j) \in A$\}. Otherwise, it is called nonzero-equilibrium pseudo-flow.*

Obviously, by the definition above, a zero-equilibrium pseudo-flow is a feasible flow that satisfies Expression (1). Therefore, we have the following theorem:

Theorem 1 *Given $\{(s_k, t_k, d_k) : k \in K\}$ and capacity reservation $\{u_{ij} : (i, j) \in A\}$, the feasible region of Expression (1) is not empty if and only if there exists a zero-equilibrium pseudo-flow.*

In fact, if there exists a nonzero-equilibrium pseudo-flow, there is no feasible solution for Expression (1). Before proving this conclusion, we need the following lemma, which was originally given by [Onaga and Kakusho(1971)] and [Iri(1971)], and subsequently observed by [Matula and Shahrokhi(1986)].

Lemma 1 *Given $\{(s_k, t_k, d_k) : k \in K\}$ and capacity reservation $\{u_{ij} : (i,j) \in A\}$, the feasible region of Expression (1) is not empty if and only if:*

$$\sum_{k \in K} l^{\mu}_{s_k,t_k} d_k \leq \sum_{(i,j)\in A} \mu_{ij} u_{ij}, \forall \mu : A \to \mathbb{Z}^+ \cup \{0\} \tag{4}$$

where $l^{\mu}_{s_k,t_k}$ is the length of the shortest path from s_k to t_k using μ as weights for the arcs.

Theorem 2 *Given $\{(s_k, t_k, d_k) : k \in K\}$ and capacity reservation $\{u_{ij} : (i,j) \in A\}$, the feasible region of Expression (1) is empty if there exists a nonzero-equilibrium pseudo-flow.*

Proof: Let f^p_k be the flow on path p connecting s_k and t_k and $\delta^k_{a,p}$ indicator variable where

$$\delta^k_{a,p} = \begin{cases} 1 & \text{if arc } a \text{ is on path } p \text{ connecting } s_k \text{ and } t_k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Let P_k be the set of all the used paths connecting s_k and t_k , we have

$$\sum_{p \in P_k} f^p_k = d_k \quad \forall k \in K \tag{6}$$

Let l_{s_k,t_k} be the length of the shortest path from s_k to t_k and $l^p_{s_k,t_k}$ the length of the path p connecting s_k and t_k using the penalty function $\{h(f_a) : \forall a \in A\}$ as weights for the arcs. The following formulation shows the relationship between $l^p_{s_k,t_k}$ and $\{h(f_a) : \forall a \in A\}$.

$$l^p_{s_k,t_k} = \sum_{a \in A} h(f_a)\delta^k_{a,p} \tag{7}$$

Based on the relationship between arc flows and path flows, the following equation holds:

$$f_a = \sum_{k \in K}\sum_{p \in P_k} \delta^k_{a,p} f^k_p \tag{8}$$

According to the definition of the equilibrium pseudo-flow, all used paths connecting s_k and t_k have equal and minimum length, that is,

$$\begin{cases} l^p_{s_k,t_k} = l_{s_k,t_k} \text{ if } f^p_k > 0 \\ l^p_{s_k,t_k} \geq l_{s_k,t_k} \text{ if } f^p_k = 0 \end{cases} \tag{9}$$

Then we have

$$\begin{aligned} \sum_{k \in K} l_{s_k,t_k} d_k &= \sum_{k \in K} l_{s_k,t_k} (\sum_{p \in P_k} f^k_p) && \backslash\backslash \text{ by Expression (6)} \\ &= \sum_{k \in K} (\sum_{p \in P_k} l_{s_k,t_k} f^k_p) \\ &= \sum_{k \in K} (\sum_{p \in P_k} l^p_{s_k,t_k} f^k_p) && \backslash\backslash \text{ by Expression (9)} \\ &= \sum_{k \in K} (\sum_{p \in P_k} (\sum_{a \in A} h(f_a)\delta^k_{a,p}) f^k_p) && \backslash\backslash \text{ by Expression (7)} \\ &= \sum_{a \in A} h(f_a) (\sum_{k \in K}\sum_{p \in P_k} \delta^k_{a,p} f^k_p) \\ &= \sum_{a \in A} h(f_a) f_a && \backslash\backslash \text{ by Expression (8)} \end{aligned} \tag{10}$$

According to the definition of the nonzero-equilibrium pseudo-flow, there exists at least an arc a that satisfies $f_a > u_a$. Since $h(f_a)$ = 0 if $f_a \leq u_a$ and $h(f_a) > 0$ if $f_a > u_a$, $\sum_{a\in A} h(f_a)f_a > \sum_{a\in A} h(f_a)u_a$. That is,

$$\begin{aligned}\sum_{k\in K} l_{s_k,t_k} d_k &= \sum_{a\in A} h(f_a)f_a \qquad \backslash\backslash \text{ by Formulation (10)} \\ &> \sum_{a\in A} h(f_a)u_a\end{aligned} \tag{11}$$

By Lemma 1(viewing h as μ),the feasible region of Expression (1) is empty. □

Remark 1 *In fact, Theorem 2 is a necessary and sufficient condition. We would see that in Section 3.*

Assume we have an algorithm to get the equilibrium pseudo-flow. Based on Theorem 1 and Theorem 2, we have the following conclusion:

Theorem 3 *If the equilibrium pseudo-flow is a nonzero-equilibrium pseudo-flow, there exists no feasible solution for Expression (1); if the equilibrium pseudo-flow is a zero-equilibrium pseudo-flow, there exists feasible solution for Expression (1) and the zero-equilibrium pseudo-flow is a feasible solution.*

So what we need to do is only to design an algorithm to obtain the equilibrium pseudo-flow.

3 The Formulation of MFP

The multi-commodity flow problem (MFP) is always regarded as a linear programming problem. However, in this part, we will give a non-linear programming formulation of MFP, whose solution is an equilibrium pseudo-flow.

3.1 The Basic Formulation

Let f_a be the sum of the flow of all pairs on arc a and $h(f_a)$ be penalty function on arc a.

$$\begin{aligned}\mathbf{min} \quad & z = \sum_a \int_0^{f_a} h(\omega)d\omega \\ \mathbf{s.t} \quad & \sum_{j\in\delta^+(i)} f_{ij}^k - \sum_{j\in\delta^-(i)} f_{ji}^k = \begin{cases} d_k, \text{ if } i = s_k \\ -d_k, \text{ if } i = t_k \\ 0, \text{ if } i \in V - \{s_k, t_k\} \end{cases} \\ & f_{ij}^k \geq 0, \forall k \in K, (i,j) \in A\end{aligned} \tag{12}$$

In the program above, the objective function is the sum of the integrals of the arc penalty function. The first constraint is flow conservation constraint and the second is non-negative constraint. Note that there is no capacity constraint here. According to the definition of the penalty function h, if the feasible region of Expression (1) is not empty, *the minimum value of the objective function is zero; otherwise it is greater than zero.*

The formulation above is similar to Beckmann Formulation ([Beckmann et al.(1956)]), whose solution is called User Equilibrium ([Wardrop (1953)]). However, [Beckmann et al.(1956)] didn't give an reasonable interpretation of the objective function. It is just viewed strictly as a mathematical construct that is utilized to solve User Equilibrium problems. In this paper we give an economic interpretation of the objective function.

Let's look at a simple example. Assume there are ten cars queueing up to cross an intersection. The intersection allows a car to pass at one time and each car will take 1 unit time to go through the intersection. Obviously, after 10 units time all the cars would go through the intersection. Now let's look this phenomenon from another perspective. The time the ith car spends to go through the intersection is i units time because it needs to wait until the cars in front go through the intersection. Therefore, the sum of the time of every car to go through the intersection is $1 + 2 + 3 + \cdots + 10 = 55$. That is, the sum of the time of every car to go through the intersection is 55 and the time of the last car to go through the intersection is 10. Now let's look at the objective function. The penalty of that the ith unit flow passes through the arc a is $h(i)$. The integration $\int_0^{f_a} h(\omega)d\omega$ means the sum of the penalty of every unit flow to pass through the arc a. *For an arc a, what the objective function minimizes is the sum of the penalty of every unit flow to pass through the arc a, not the penalty of the last unit flow to pass through the arc a.*

3.2 Equivalence

To demonstrate the equivalence between the equilibrium pseudo-flow and Program (12), it has to be shown that any flow pattern that solves Program (12) satisfies the equilibrium conditions. This equivalency is demonstrated in this part by proving that the Karush-Kuhn-Tucker conditions for Program (12) are identical to the equilibrium conditions.

Lemma 2 $t(f_a) = \int_0^{f_a} h(\omega)d\omega$ *is a convex function.*

Proof: Proof. The derivative of $t(f_a)$ is $h(f_a)$, which is monotone nondecreasing function. So $t(f_a) = \int_0^{f_a} h(\omega)d\omega$ is a convex function. □

Lemma 3 *Let* $\mathbf{f}^*$ *be a solution of Program (12).* $\mathbf{f}^*$ *is the optimal solution of Program (12) if and only if* $\mathbf{f}^*$ *satisfies the Karush-Kuhn-Tucker conditions of Program (12).*

Proof: Proof. By Lemma 2, $t(f_a) = \int_0^{f_a} h(\omega)d\omega$ is a convex function. Therefore, the objective function $z = \sum_a \int_0^{f_a} h(\omega)d\omega$ is a convex function. Besides, the inequality constraints of Program (12) are continuously differentiable convcave functions and the equality constraints of Program (12) are affine functions. So Karush-Kuhn-Tucker conditions are necessary and sufficient for optimality of Program (12) ([Boyd and Vandenberghe(2004)]). □

Obviously, Program (12) is a minimization problem with nonnegativity constraints and linear equality. The Karush-Kuhn-Tucker conditions of such formulation are as following:

$$
\begin{aligned}
&\textbf{Stationarity}\\
&-\frac{\partial z}{\partial f_{ij}^k} = -\mu_{ij}^k + (\lambda_i^k - \lambda_j^k), \forall k \in K, (i,j) \in A\\
&\textbf{Primal feasibility}\\
&\sum_{j\in\delta^+(i)} f_{ij}^k - \sum_{j\in\delta^-(i)} f_{ji}^k = \begin{cases} d_k, \text{ if } i = s_k \\ -d_k, \text{ if } i = t_k \\ 0, \text{ if } i \in V - \{s_k, t_k\} \end{cases}\\
&-f_{ij}^k \le 0, \forall k \in K, (i,j) \in A\\
&\textbf{Dual feasibility}\\
&\mu_{ij}^k \ge 0, \forall k \in K, (i,j) \in A\\
&\textbf{Complementary slackness}\\
&\mu_{ij}^k f_{ij}^k = 0, \forall k \in K, (i,j) \in A
\end{aligned}
\tag{13}
$$

Obviously,

$$\frac{\partial z}{\partial f_{ij}^k} = h(f_{ij})\frac{\partial f_{ij}}{\partial f_{ij}^k} = h(f_{ij})$$

Substituting the expression above into Stationarity expression in KKT conditions,

$$h(f_{ij}) = \mu_{ij}^k + (\lambda_j^k - \lambda_i^k), \forall k \in K, (i,j) \in A$$

For a path $p = (s_k, v_1, v_2, \cdots, v_m, t_k)$, the length of p is

$$
\begin{aligned}
length(p) &= h(f_{s_k v_1}) + h(f_{v_1 v_2}) + \cdots + h(f_{v_{m-1} v_m}) + h(f_{v_m t_k})\\
&= \mu_{s_k v_1}^k + (\lambda_{v_1}^k - \lambda_{s_k}^k) + \mu_{v_1 v_2}^k + (\lambda_{v_2}^k - \lambda_{v_1}^k) + \cdots + \mu_{v_m t_k}^k + (\lambda_{t_k}^k - \lambda_{v_m}^k)\\
&= \lambda_{t_k}^k - \lambda_{s_k}^k + \mu_{s_k v_1}^k + \mu_{v_1 v_2}^k + \cdots + \mu_{v_m t_k}^k
\end{aligned}
$$

The condition above holds for every path between any pair in the network. For an arc (i,j) on a used path p_{used} between $pair$ (s_k, t_k), the flow f_{ij}^k is greater than zero. By complementary slackness $\mu_{ij}^k f_{ij}^k = 0$ in KKT conditions, we have $\mu_{ij}^k = 0$. Therefore,

$$
\begin{aligned}
length(p_{used}) &= \lambda_{t_k}^k - \lambda_{s_k}^k + \mu_{s_k v_1}^k + \mu_{v_1 v_2}^k + \cdots + \mu_{v_m t_k}^k\\
&= \lambda_{t_k}^k - \lambda_{s_k}^k
\end{aligned}
$$

By the expression above, all the used paths between $pair$ (s_k, t_k) have the same length $(\lambda^k_{t_k} - \lambda^k_{s_k})$.

For an unused path p_{unused} between $pair$ (s_k, t_k), the length of p_{unused} is

$$length(p_{unused}) = \lambda^k_{t_k} - \lambda^k_{s_k} + \mu^k_{s_k v_1} + \mu^k_{v_1 v_2} + \cdots + \mu^k_{v_m t_k}$$

By dual feasibility $\mu^k_{ij} \geq 0$ in KKT conditions, $length(p_{unused})$ is greater or equal to $length(p_{used})$.

With this interpretation above, it is now clear that:

(i) all the used paths connecting $s_k$and t_k have equal and minimum length;

(ii) all the unused paths connecting $s_k$and t_k have greater or equal length;

That is, the optimal solution of Program (12) is an equilibrium pseudo-flow.

3.3 Frank-Wolfe Algorithm

The Program (12) includes a convex objective function, a linear constraint set and a non-negative constraint set, which could be efficiently solved by Frank-Wolfe algorithm ([Frank and Wolfe(1956)]). Applying Frank-Wolfe algorithm to Program (12), at the nth iteration, needs the following linear program:

$$\begin{aligned}
\mathbf{min} \quad & z^n(\mathbf{y}) = \sum_{k,ij} \frac{\partial z(\mathbf{f}_n)}{\partial f^k_{ij}} y^k_{ij} = \sum_{k,ij} h(f_{ij,n}) y^k_{ij} \\
\mathbf{s.t} \quad & \sum_{j \in \delta^+(i)} y^k_{ij} - \sum_{j \in \delta^-(i)} y^k_{ji} = \begin{cases} d_k, \text{ if } i = s_k \\ -d_k, \text{ if } i = t_k \\ 0, \text{ if } i \in V - \{s_k, t_k\} \end{cases} \\
& y^k_{ij} \geq 0, \forall k \in K, (i,j) \in A
\end{aligned} \tag{14}$$

where $f_{ij,n}$ is the flow on arc (i, j) at the nth iteration.

Note that this program doesn't have capacity constraints and the penalties are not flow-dependent. In other words, the program minimizes the total penalties over a network with fixed penalties $\{h(f_{ij,n}) : \forall(i, j) \in A\}$. Obviously, the penalties will be minimized by assigning all $s_k - t_k$ flows to the shortest path connecting s_k and t_k. Such an assignment is performed by computing the shortest paths between all pairs. Since the penalty of each arc is 0 at $0th$ iteration, we can establish the initial pseudo-flow $\mathbf{f}$ by assigning all the $s_k - t_k$ flow to certain path connecting s_k and t_k. Therefore, The Frank-Wolfe algorithm applied to solve Program (12) can be given as follows:

Algorithm 1 Frank-Wolfe Algorithm applied to MCF

Initialization: examine each pair (s_k, t_k) in turn and assign all the $s_k - t_k$ flow to certain path connecting s_k and t_k. This yields $\mathbf{f}_1$. Set $n = 1$;
repeat
  Update: set $\{h(f_{ij,n}) : \forall(i, j) \in A\}$ as the weights of every arc;
  Direction-finding: compute the shortest paths between all pairs and assigning all $s_k - t_k$ flows to the shortest path connecting s_k and t_k, which yields $\mathbf{y}_n$.
  Line search: find α_n by solving $\mathbf{min}_{0 \leq \alpha \leq 1} \sum_{(i,j) \in A} \int_0^{f_{ij,n} + \alpha(y_{ij,n} - f_{ij,n})} h(\omega) d\omega$
  Move: set $f_{ij,n+1} = f_{ij,n} + \alpha_n(y_{ij,n} - f_{ij,n})$
until some convergence criterion is met

Remark 2 *The convergence of the Frank–Wolfe algorithm above is sublinear, that is, the error in the objective function to the optimum is $O(1/k)$ after k iterations[Frank and Wolfe(1956)].*

Remark 3 *In fact, all the conclusion in this paper is true if the penalty function h satisfies the following definition,*

$$h(f_{ij}) = \begin{cases} 0 & if\ f_{ij} \leq u_{ij} \\ g(f_{ij} - u_{ij}) & if\ f_{ij} > u_{ij} \end{cases}$$

where $g(0) = 0$ and $g(x)$ is strictly monotone increasing function when $x \geq 0$.

Remark 4 *If the penalty function h is defined as following,*

$$h(f_{ij}) = \begin{cases} c_{ij} & if\ f_{ij} \le u_{ij} \\ c_{ij} + M(f_{ij} - u_{ij}) & if\ f_{ij} > u_{ij} \end{cases}$$

Program (12) is a description of minimum cost multi-commodity flow problem*, where M is big enough and $\{c_{ij} : (i,j) \in A\}$ is the cost of every arc. Therefore, by defining the penalty function h as above, the content in this paper can be easily generalized to minimum cost multi-commodity flow problem.*

Remark 5 *There exist several variants of the Frank-Wolfe algorithm which have faster convergence([Lacoste-Julien and Jaggi(2015), Pena and Rodriguez(2019)]). We would not discuss here.*

Remark 6 *Since the objective function z is convex, we have $z(\mathbf{x}) \ge z(\mathbf{y}) + (\mathbf{x} - \mathbf{y})^T \nabla z(\mathbf{y})$, which also holds for the optimal solution $\mathbf{f}^*$. That is, $z(\mathbf{f}^*) \ge z(\mathbf{f^n}) + (\mathbf{f}^* - \mathbf{f^n})^T \nabla z(\mathbf{f^n}) \ge min_{\mathbf{x} \in D}\{z(\mathbf{f^n}) + (\mathbf{x} - \mathbf{f^n})^T \nabla z(\mathbf{f^n})\} = z(\mathbf{f^n}) - \mathbf{f^n}^T \nabla z(\mathbf{f^n}) + min_{\mathbf{x} \in D}\{\mathbf{x}^T \nabla z(\mathbf{f^n})\}$, where $\mathbf{f^n}$ is the flow at the nth iteration and D is the feasible region. If $z(\mathbf{f^n}) - \mathbf{f^n}^T \nabla z(\mathbf{f^n}) + min_{\mathbf{x} \in D}\{\mathbf{x}^T \nabla z(\mathbf{f^n})\} > 0$, the objective function z must be greater than zero and there exists no zero-equilibrium pseudo-flow. The algorithm can be terminated in advance.*

4 Conclusion

This paper gives a combinatorial algorithm for the multi-commodity flow problem. Unlike other methods, the combinatorial algorithm does not maintain the capacity constraints throughout the execution. The algorithm, however, maintains a pseudo-flow, which just satisfies the flow conservation on every nodes. We introduce a *penalty function h* for each arc, which is positively related to the quantity that the flow exceeds the capacity. Then a *non-linear* description of the multi-commodity flow problem is given whose solution is equilibrium pseudo-flow and we also give a combinatorial algorithm to obtain equilibrium pseudo-flow. Generally, we give a new idea to solve the multi-commodity flow problem, but it need further studies to design more effective algorithm to obtain the equilibrium pseudo-flow.

References

[Iri(1971)] Iri M (1971) On an extension of the max-flow min-cut theorem for multicommodity flows. *J. Oper. Res. Soc. Japan.* 13: 129-135.

[Matula and Shahrokhi(1986)] Matula D, Shahrokhi F (1986) The maximum concurrent flow problem and sparsest cuts. Tech. ReportSouthern Methodist Univ.

[Onaga and Kakusho(1971)] Onaga K, Kakusho O (1971) On feasibility conditions of multicommodity flows in networks. *Circuit Theory, IEEE Transactions on.* 18(4): 425-429.

[Ford and Fulkerson(1962)] Ford Jr L R, Fulkerson D R (1962) Flows in networks[M]. Princeton university press.

[Hu(1963)] Hu T C (1963) Multi-commodity network flows[J]. Operations research, 11(3): 344-360.

[Assad(1978)] Assad A A (1978) Multicommodity network flows—a survey[J]. Networks, 8(1): 37-91.

[Edmonds and Karp(1972)] Edmonds J, Karp R M (1972) Theoretical improvements in algorithmic efficiency for network flow problems[J]. Journal of the ACM (JACM), 19(2): 248-264.

[Dinic(1970)] Dinic E A (1970) An algorithm for the solution of the problem of maximal flow in a network with power estimation.[J]. Soviet Math Doklady, 11:754-757.

[Goldberg and Tarjan(1988)] Goldberg A V, Tarjan R E (1988) A new approach to the maximum-flow problem[J]. Journal of the ACM (JACM), 35(4): 921-940.

[Klein(1967)] Klein M (1967) A primal method for minimal cost flows with applications to the assignment and transportation problems[J]. Management Science, 14(3): 205-220.

[Bellman(1958)] Bellman R (1958) On a routing problem[J]. Quarterly of applied mathematics, 16(1): 87-90.

[Moore(1959)] Moore E F (1959) The shortest path through a maze[C]//Proc. Int. Symp. Switching Theory, 1959: 285-292.

[Goldberg and Radzik(1993)] Goldberg A, Radzik T (1993) A heuristic improvement of the Bellman-Ford algorithm[R]. STANFORD UNIV CA DEPT OF COMPUTER SCIENCE.

[Pallottino(1984)] Pallottino S (1984) Shortest-path methods: Complexity, interrelations and new propositions[J]. Networks, 14(2): 257-267.

[Tarjan(1981)] Tarjan, R.E (1981) Shortest Paths. Technical report, AT&T Bell Laboratories, Murray Hill, NJ.

[Beckmann et al.(1956)] Beckmann M J, Mcguire C B, Winsten C B, et al (1956) Studies in the economics of transportation[J]. Economic Journal, 26(1):820-821.

[Wardrop (1953)] Wardrop J G (1953) Some Theoretical Aspects of Road Traffic Research[J]. OR, 4(4):72-73.

[Boyd and Vandenberghe(2004)] Boyd S, Vandenberghe L (2004) Convex optimization[M]. Cambridge university press.

[Frank and Wolfe(1956)] Frank M, Wolfe P (1956) An algorithm for quadratic programming[J]. Naval research logistics quarterly, 3(1-2): 95-110.

[Pena and Rodriguez(2019)] Pena J, Rodriguez D. Polytope conditioning and linear convergence of the Frank–Wolfe algorithm[J]. Mathematics of Operations Research, 2019, 44(1): 1-18.

[Lacoste-Julien and Jaggi(2015)] Lacoste-Julien S, Jaggi M. On the global linear convergence of Frank-Wolfe optimization variants[C]//Advances in Neural Information Processing Systems. 2015: 496-504.