A COMBINATORIAL ALGORITHM FOR THE MULTI-COMMODITY FLOW PROBLEM

A PREPRINT

Pengfei Liu
Building No.1, Zhongguancun Road
Haidian District
Beijing, China
liupengfei89@qq.com

October 21, 2018

ABSTRACT

This paper researches combinatorial algorithms 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. Then we design a combinatorial algorithm to obtain equilibrium pseudo-flow. If the equilibrium pseudo-flow is a non-zero-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. At last, a non-linear description of the multi-commodity flow problem is given, whose solution is 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 \( u : A \rightarrow Q^+ \), \( K \) 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:

\[
\begin{align*}
\sum_{k \in K} f^k_{ij} & \leq u_{ij}, \forall (i, j) \in A \\
\sum_{j \in b^+(i)} f^k_{ij} - \sum_{j \in b^-(i)} f^k_{ji} & = \begin{cases} 
  d_k, & \text{if } i = s_k \\
  -d_k, & \text{if } i = t_k \\
  0, & \text{if } i \in V - \{s_k, t_k\}
\end{cases} \\
f^k_{ij} & \geq 0, \forall k \in K, (i, j) \in A
\end{align*}
\]  

(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 de-
composition, 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.

The network notation introduced here is summarized in Table 1. Further notation is introduced as needed.

1.1 Our contribution

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

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 \( f : K \times V \times V \rightarrow \mathbb{R}^+ \) that just satisfies the flow conservation on every node. That is, a pseudo-flow \( f \) is a feasible solution of Expression (2).

\[
\sum_{j \in \delta^+ (i)} f^k_{ij} - \sum_{j \in \delta^- (i)} f^k_{ji} = \begin{cases} 
    d_k, & \text{if } i = s_k \\
    -d_k, & \text{if } i = t_k \\
    0, & \text{if } i \in V - \{s_k, t_k\}
\end{cases} \quad (2)
\]

\( f^k_{ij} \geq 0, \forall k \in K, (i, j) \in A \)

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

\[
h(f_{ij}) = \begin{cases} 
    0 & \text{if } f_{ij} \leq u_{ij} \\
    f_{ij} - u_{ij} & \text{if } f_{ij} > u_{ij}
\end{cases} \quad (3)
\]

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 \( f \) 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 \( s_k \rightarrow t_k \) 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 \( f \) is called zero-equilibrium pseudo-flow if \( \{h(f_{ij}) = 0, \forall (i, j) \in A\} \). Otherwise, it is called non-zero-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:

\[
\sum_{k \in K} l_{s_k, t_k}^p d_k \leq \sum_{(i, j) \in A} \mu_{ij} u_{ij}, \forall \mu : A \rightarrow \mathbb{Z}_+^+ \cup \{0\} 
\]

(4)

where \( l_{s_k, t_k}^p \) is the length of the shortest path from \( s_k \) to \( t_k \) using \( \mu \) 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 \( f_k^p \) be the flow on path \( p \) connecting \( s_k \) and \( t_k \) and \( \delta_{a,p}^k \) indicator variable where

\[
\delta_{a,p}^k = \begin{cases} 
1 & \text{if arc } a \text{ is on path } p \text{ connecting } s_k \text{ and } t_k \\
0 & \text{otherwise}
\end{cases}
\]

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

\[
\sum_{p \in P_k} f_k^p = d_k \quad \forall k \in K
\]

(6)

Let \( l_{s_k, t_k} \) be the length of the shortest path from \( s_k \) to \( t_k \) and \( l_{s_k, t_k}^p \) 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 \( l_{s_k, t_k}^p \) and \( \{h(f_a) : \forall a \in A\} \).

\[
l_{s_k, t_k}^p = \sum_{a \in A} h(f_a) \delta_{a,p}^k 
\]

(7)

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

\[
f_a = \sum_{k \in K} \sum_{p \in P_k} \delta_{a,p}^k f_k^p
\]

(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,

\[
\begin{align*}
\{ l_{s_k, t_k}^p = l_{s_k, t_k} & \text{ if } f_k^p > 0 \\
l_{s_k, t_k}^p \geq l_{s_k, t_k} & \text{ if } f_k^p = 0
\end{align*}
\]

(9)

Then we have

\[
\sum_{k \in K} l_{s_k, t_k} d_k = \sum_{k \in K} l_{s_k, t_k} \left( \sum_{p \in P_k} f_k^p \right) \quad \text{\( \\text{by Expression (6)} \)}
\]

\[
= \sum_{k \in K} \left( \sum_{p \in P_k} l_{s_k, t_k} f_k^p \right) \quad \text{\( \\text{by Expression (9)} \)}
\]

\[
= \sum_{k \in K} \left( \sum_{p \in P_k} l_{s_k, t_k}^p f_k^p \right) \quad \text{\( \\text{by Expression (7)} \)}
\]

\[
= \sum_{k \in K} \left( \sum_{p \in P_k} h(f_a) \delta_{a,p}^k f_k^p \right) \quad \text{\( \\text{by Expression (8)} \)}
\]

\[
= \sum_{a \in A} h(f_a) f_a
\]

(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 $f_a > u_a$, $\sum_{a \in A} h(f_a)f_a > \sum_{a \in A} \sum_{a \in A} h(f_a)u_a$. That is,

$$\sum_{k \in K} l_{s_k, t_k}d_k = \sum_{a \in A} h(f_a)f_a \quad \text{by Formulation (10)}$$

$$> \sum_{a \in A} h(f_a)u_a$$

(11)

By Lemma 1 viewing $h$ as $\mu$, 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 4.

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 Combinatorial Algorithm

In this section, we give a combinatorial algorithm, called Cycle-canceling algorithm, to obtain the equilibrium pseudo-flow for the multi-commodity flow problem. The cycle-canceling algorithm is firstly proposed by [Klein (1967)] for minimum-cost flow problem.

#### 3.1 Optimal Condition

The algorithm relies on the concept of residual networks. The residual network $G(f, s_k, t_k)$ corresponding to a pseudo-flow $f$ and pair $(s_k, t_k)$ is defined as follows. Each arc $(i, j) \in A$ is replaced by two arcs $(i, j)$ and $(j, i)$. The arc $(i, j)$ has cost $c_{ij} = h(f_{ij})$ and residual capacity $r_{ij} = +\infty$, and the arc $(j, i)$ has cost $c_{ji} = -h(f_{ij})$ and residual capacity $r_{ji} = f_{ji}^k$. The residual network doesn’t consist of arcs with non-positive residual capacity.

First, we give the following theorem, which is called negative cycle condition.

**Theorem 4** A pseudo-flow $f$ is an equilibrium pseudo-flow if and only if it satisfies the negative cycle condition: namely, the residual network $G(f, s_k, t_k)$ contains no negative cost (directed) cycle for any pair $(s_k, t_k)$.

**Proof:** Proof. Suppose that $f$ is a pseudo-flow and that $G(f, s_k, t_k)$ contains a negative directed cycle. Without loss of generality, assume the cycle $C = \{v_0, v_1, v_2, v_3, \ldots, v_r, v_0\}$. For convenience, we need the following definitions. A point $v_i \in C$ is called an alternating point if it satisfies that $e_{v_{i-1}v_i} < 0$ and $e_{v_iv_{i+1}} \geq 0$. An alternating point $v_i$ is called positive alternating point if it satisfies $e_{v_{i-1}v_i} < 0$ and $e_{v_iv_{i+1}} \geq 0$. An alternating point $v_i$ is called negative alternating point if it satisfies $e_{v_{i-1}v_i} \geq 0$ and $e_{v_iv_{i+1}} < 0$. Obviously, there are even alternating points on a cycle. Let $\mathcal{N}(C) = \{v_{a_1}, v_{a_2}, \ldots, v_{a_{2m}}\}$ (arranged in order) be the set of alternating points on cycle $C$. Without loss of generality, assume $v_{a_1}$ is a negative alternating point. Apparently, $\mathcal{N}^+(C) = \{v_{a_2}, v_{a_4}, \ldots, v_{a_{2m}}\}$ is the set of positive alternating points on cycle $C$ and $\mathcal{N}^-(C) = \{v_{a_1}, v_{a_3}, \ldots, v_{a_{2m-1}}\}$ the set of negative alternating points.

Let $v_{a_1}, v_{a_{i+1}}$ be the path from $v_{a_1}$ to $v_{a_{i+1}}$ on the cycle $C$ and $-v_{a_1}, v_{a_{i+1}}$ the path in the opposite direction. According to the definition of positive and negative alternating points, the arcs on $v_{a_{2i-1}}, v_{a_{2i}}$ have negative weights. By the definition of the residual network, there is $(s_k, t_k)$ flow on $-v_{a_{2i-1}}, v_{a_{2i}}$. So there exists a used path $(p_{s_k, v_{a_1}}, -v_{a_{2i-1}}, v_{a_{2i}}, v_{a_{2i+1}}, -v_{a_{2i+1}}, t_k)$. For arc $v_{a_{2i}}, v_{a_{2i+1}}$, there may, or may not, exist $(s_k, t_k)$ flow on it. Since $(p_{s_k, v_{a_1}}, -v_{a_{2i-1}}, v_{a_{2i}}, v_{a_{2i+1}}, -v_{a_{2i+1}}, t_k)$ is a used path and $(p_{s_k, v_{a_1}}, v_{a_{2i}}, v_{a_{2i+1}}, t_k)$ may be an unused path, according the definition of equilibrium pseudo-flow, we have

$$\text{len}(p_{s_k, v_{a_1}}) + \text{len}(-v_{a_{2i-1}}, v_{a_{2i}}) + \text{len}(p_{a_{2i-1}, t_k})$$

$$\leq \text{len}(p_{s_k, v_{a_1}}) + \text{len}(v_{a_{2i}}, v_{a_{2i+1}}) + \text{len}(p_{a_{2i+1}, t_k}) \quad \forall i \in \{1, 2, \ldots, m\}$$

(12)
where \( \text{len}(p) \) means the length of path \( p \) (may be negative). Note that \( v_{a2m+1} \) is \( v_{a1} \).

Sum over all \( i \),

\[
\sum_{i=1}^{m} \left( \text{len}(p_{sk,v_{a2i}}) + \text{len}(-v_{a2i-1,v_{a2i}},p_{v_{a2i-1},t_k}) \right)
\]

\[
\leq \sum_{i=1}^{m} \left( \text{len}(p_{sk,v_{a2i}}) + \text{len}(v_{a2i,v_{a2i+1}}) + \text{len}(p_{v_{a2i+1},t_k}) \right)
\]

\[
\downarrow
\]

\[
\sum_{i=1}^{m} \left( \text{len}(p_{sk,v_{a2i}}) + \text{len}(v_{a2i,v_{a2i+1}}) + \text{len}(p_{v_{a2i+1},t_k}) \right)
\]

By the last equation, the cycle is a non-negative cycle, which is in contradiction with the assumption that \( C \) is a negative cycle. Therefore, if \( f \) is an equilibrium pseudo-flow, \( G(f, s_k, t_k) \) contains no negative cycle.

Assume \( f \) is not an equilibrium pseudo-flow. By the definition of equilibrium pseudo-flow, there are two cases:

(i) for certain pair \((s_k, t_k)\), there exist two used paths \( p_1(s_k, t_k) \) and \( p_2(s_k, t_k) \) whose length are not equal. Without loss of generality, assume \( \text{len}(p_1(s_k, t_k)) < \text{len}(p_2(s_k, t_k)) \).

(ii) for certain pair \((s_k, t_k)\), there exist an unused path \( p_1(s_k, t_k) \) and a used path \( p_2(s_k, t_k) \) that satisfy \( \text{len}(p_1(s_k, t_k)) < \text{len}(p_2(s_k, t_k)) \).

Let \( v_{a1}, v_{a2}, \cdots, v_{am} \) be the shared points of \( p_1(s_k, t_k) \) and \( p_2(s_k, t_k) \) in order. Note that \( v_{a1} \) is \( s_k \), \( v_{am} \) is \( t_k \). Since \( \text{len}(p_1(s_k, t_k)) < \text{len}(p_2(s_k, t_k)) \), by the drawer principle, there exists at least a \( j \in \{1, 2, \cdots, m\} \) that satisfies \( \text{len}(p_1(v_{aj}, v_{aj+1})) < \text{len}(p_2(v_{aj}, v_{aj+1})) \). So \( p_1(v_{aj}, v_{aj+1}) \) and \( -p_2(v_{aj}, v_{aj+1}) \) constitute a negative cycle. That is, \( G(f, s_k, t_k) \) contains a negative cycle. Therefore, if \( G(f, s_k, t_k) \) contains no negative cycle, \( f \) is an equilibrium pseudo-flow.

\[\square\]

### 3.2 Cycle-canceling Algorithm

The optimality condition above suggests a simple algorithmic approach for solving the multi-commodity flow problem, which is called the cycle-canceling algorithm here.

Firstly, the algorithm establishes a pseudo-flow \( f \) in the network. There are many ways to establish an initial pseudo-flow \( f \) in the network. For example, 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 \( t_k \).

Then it iteratively finds negative cost-directed cycles in the residual network and augments flows on these cycles until the residual network contains no negative cost-directed cycle. We give the MCF Cycle-canceling Algorithm as following:
Algorithm 1 MCF Cycle-canceling Algorithm

establish an initial pseudo-flow \( f \) in the network by examining each pair \((s_k, t_k)\) in turn and assigning all the \( s_k - t_k \) flow to certain path connecting \( s_k \) and \( t_k \);

repeat
  use some algorithm to identify a negative cycle \( W \) in any residual network \( G(f, s_k, t_k) \);
  compute \( \delta \) by Program (14);
  augment \( \delta \) units flow in the cycle \( W \), that is, update \( \{ f_{ij}^k := f_{ij}^k + \delta, \forall (i, j) \in W^+ \} \) and \( \{ f_{ij}^k := f_{ij}^k - \delta, \forall (j, i) \in W^- \} \);
  update residual network \( G(f, s_k, t_k) \);
until \( \{ G(f, s_k, t_k), \forall k \in K \} \) contain no negative cycle

where \( \{ W^+ : e_{ij} \geq 0, (i, j) \in W \} \) and \( \{ W^- : e_{ij} < 0, (i, j) \in W \} \). That is, \( W^+ \) is the set of arcs that have non-negative cost and \( W^- \) the set of arcs that have negative cost in the cycle \( W \).

\[
\begin{aligned}
\max_{\forall (i,j) \in W^+} & \quad \sum h(f_{ij} + \delta) - \sum_{\forall (j,i) \in W^-} h(f_{ij} - \delta) \\
\text{s.t} & \quad \sum_{\forall (i,j) \in W^+} h(f_{ij} + \delta) - \sum_{\forall (j,i) \in W^-} h(f_{ij} - \delta) \leq 0 \\
& \quad 0 \leq \delta \leq r_{ij}, \forall (i,j) \in W
\end{aligned}
\tag{14}
\]

The objective function is the sum of the cost of the cycle \( W \) after augmenting \( \delta \) units flow in the cycle \( W \). The first constraint simply states that the value of the objective function is no greater than zero, which means that the cycle \( W \) should not be a positive cycle after augmenting \( \delta \) units flow. The second constraint means that \( \delta \) does not exceed residual capacity.

There are many algorithms for identifying a negative cycle like Bellman-Ford-Moore algorithm ([Bellman(1958] [Ford and Fulkerson(1962)] [Moore(1959)]), the Goldberg-Radzik algorithm ([Goldberg and Radzik(1993)]), the algorithm of Pallottino ([Pallottino(1984)]), the algorithm of Tarjan ([Tarjan(1981)]). We omit it here.

When MCF Cycle-canceling Algorithm terminates, we obtain an equilibrium pseudo-flow. If the equilibrium pseudo-flow is a zero-equilibrium pseudo-flow, there exists no feasible solution for Expression (1); if the equilibrium pseudo-flow is a zero-equilibrium pseudo-flow, we get a feasible solution for Expression (1) and the zero-equilibrium pseudo-flow is the feasible solution. As far as we know, the Cycle-canceling Algorithm above is the first combinatorial algorithm for the multi-commodity flow problem.

4 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.

4.1 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 \( a \).

\[
\begin{aligned}
\min \quad & z = \sum_a \int_0^{f_a} h(\omega) d\omega \\
\text{s.t} \quad & \sum_{j \in \delta^+(i)} f_{ij}^k - \sum_{j \in \delta^+(i)} f_{ji}^k = \begin{cases} 
  d_k, & \text{if } i = s_k \\
  -d_k, & \text{if } i = t_k \\
  0, & \text{if } i \in V \setminus \{s_k, t_k\}
\end{cases} \\
& f_{ij}^k \geq 0, \forall k \in K, (i, j) \in A
\end{aligned}
\tag{15}
\]

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 queuing 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

\[
\sum_{i} \text{time of every car to go through the intersection is 55}
\]

and the time of the last car is 10. Now let’s look at the objective function. The penalty of the last unit flow to pass through the arc a is

\[
\sum_{a} \text{unit flow passes through the arc a}.
\]

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.

### 4.2 Equivalence

To demonstrate the equivalence between the equilibrium pseudo-flow and Program (15), it has to be shown that any flow pattern that solves Program (15) satisfies the equilibrium conditions. This equivalency is demonstrated in this part by proving that the Karush-Kuhn-Tucker conditions for Program (15) are identical to the equilibrium conditions.

#### Lemma 2

\[
t(a) = \int_{0}^{f_{a}} h(\omega) d\omega \text{ is a convex function.}
\]

**Proof**: Proof. The derivative of \( t(a) \) is \( h(f(a)) \), which is monotone nondecreasing function. So \( t(a) = \int_{0}^{f_{a}} h(\omega) d\omega \) is a convex function.

#### Lemma 3

Let \( f^* \) be a solution of Program (15). \( f^* \) is the optimal solution of Program (15) if and only if \( f^* \) satisfies the Karush-Kuhn-Tucker conditions of Program (15).

**Proof**: Proof. By Lemma 2, \( t(a) = \int_{0}^{f_{a}} h(\omega) d\omega \) is a convex function. Therefore, the objective function \( z = \sum \int_{0}^{f_{a}} h(\omega) d\omega \) is a convex function. Besides, the inequality constraints of Program (15) are continuously differentiable convex functions and the equality constraints of Program (15) are affine functions. So Karush-Kuhn-Tucker conditions are necessary and sufficient for optimality of Program (15) ([Boyd and Vandenberghe(2004)]). Obviously, Program (15) is a minimization problem with nonnegativity constraints and linear equality. The Karush-Kuhn-Tucker conditions of such formulation are as following:

**Stationarity**

\[
- \frac{\partial z}{\partial f_{ij}} = -\mu_{ij}^k + (\lambda_{ij}^k - \lambda_{ij}^s), \forall k \in K, (i, j) \in A
\]

**Primal feasibility**

\[
\sum_{j \in \delta^+(i)} f_{ij}^k - \sum_{j \in \delta^-(i)} f_{ji}^k = \begin{cases} 
    d_k, & \text{if } i = s_k \\
    -d_k, & \text{if } i = t_k \\
    0, & \text{if } i \in V - \{s_k, t_k\} 
\end{cases}
\]

\[
- f_{ij}^k \leq 0, \forall k \in K, (i, j) \in A
\]

**Dual feasibility**

\[
\mu_{ij}^k \geq 0, \forall k \in K, (i, j) \in A
\]

**Complementary slackness**

\[
\mu_{ij}^k f_{ij}^k = 0, \forall k \in K, (i, j) \in A
\]
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
\[
length(p) = h(f_{s_kv_1}) + h(f_{v_1v_2}) + \cdots + h(f_{v_mv_m}) + h(f_{v_mt_k})
\]
\[
= \mu_{s_kv_1}^k + (\lambda_{v_1}^k - \lambda_{s_k}^k) + \mu_{v_1v_2}^k + (\lambda_{v_2}^k - \lambda_{v_1}^k) + \cdots + \mu_{v_mt_k}^k + (\lambda_{t_k}^k - \lambda_{v_m}^k)
\]
\[
= \lambda_{t_k}^k - \lambda_{s_k}^k + \mu_{s_kv_1}^k + \mu_{v_1v_2}^k + \cdots + \mu_{v_mt_k}^k
\]
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,
\[
length(p_{used}) = \lambda_{t_k}^k - \lambda_{s_k}^k + \mu_{s_kv_1}^k + \mu_{v_1v_2}^k + \cdots + \mu_{v_mt_k}^k
\]
By the expression above, all the used paths between pair \((s_k, t_k)\) have the same length \((\lambda_{t_k}^k - \lambda_{s_k}^k)\).
For an unused path \( p_{unused} \) between pair \((s_k, t_k)\), the length of \( p_{unused} \) is
\[
length(p_{unused}) = \lambda_{t_k}^k - \lambda_{s_k}^k + \mu_{s_kv_1}^k + \mu_{v_1v_2}^k + \cdots + \mu_{v_mt_k}^k
\]
By dual feasibility \( \mu_{ij}^k \geq 0 \) in KKT conditions, \( length(p_{unused}) \) is greater or equal to \( length(p_{used}) \).
With this interpretation above, it is now clear that:
1. all the used paths connecting \( s_k \) and \( t_k \) have equal and minimum length;
2. all the unused paths connecting \( s_k \) and \( t_k \) have greater or equal length;
That is, the optimal solution of Program (15) is an equilibrium pseudo-flow.

### 4.3 Frank-Wolfe Algorithm

The Program (15) 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 (15), at the \( nth \) iteration, needs the following linear program:

\[
\begin{align*}
\min \quad & z^n(y) = \sum_{k,ij} \frac{\partial z}{\partial f_{ij}^k} y_{ij}^k = \sum_{k,ij} h(f_{ij,n}) y_{ij}^k \\
\text{s.t.} \quad & \sum_{j \in \delta^+(i)} y_{ij}^k - \sum_{j \in \delta^-(i)} y_{ji}^k = \begin{cases} d_k, & \text{if } i = s_k \\ -d_k, & \text{if } i = t_k \\ 0, & \text{if } i \in V - \{s_k, t_k\} \end{cases} \\
& y_{ij}^k \geq 0, \forall k \in K, (i, j) \in A
\end{align*}
\]
(17)
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 \( \{h(f_{ij,n}) : \forall (i, j) \notin A\} \). 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 0\( 0^{th} \) iteration, we can establish the initial pseudo-flow \( f \) as Algorithm 4. Therefore, The Frank-Wolfe algorithm applied to solve Program (15) can be given as follows:
Algorithm 2 Frank-Wolfe Algorithm applied to MCF

**Initialization:** establish the initial pseudo-flow $f$ as Algorithm 1 in the network. This yields $f_1$. Set $n = 1;
repeat
  \text{Update: set } \{h(f_{ij,n}) : \forall (i, j) \in A\} \text{ as the weights of every arc;}
  \text{Direction-finding: compute the shortest paths between all pairs and assigning all } s_k - t_k \text{ flows to the shortest path connecting } s_k \text{ and } t_k, \text{ which yields } y_n.
  \text{Line search: find } \alpha_n \text{ by solving } \min_{0 \leq \alpha \leq 1} \sum_{(i,j) \in A} \int f_{ij,n} + \alpha(y_{ij,n} - f_{ij,n}) h(\omega) d\omega
  \text{Move: set } f_{ij,n+1} = f_{ij,n} + \alpha_n(y_{ij,n} - f_{ij,n})
\until \text{some convergence criterion is met}

**Remark 2** In fact, all the conclusion in this paper is true if the penalty function $h$ satisfies the following definition,
\[
h(f_{ij}) = \begin{cases} 
0 & \text{if } f_{ij} \leq u_{ij} \\
g(f_{ij} - u_{ij}) & \text{if } f_{ij} > u_{ij}
\end{cases}
\]
where $g(0) = 0$ and $g(x)$ is strictly monotone increasing function when $x \geq 0$.

**Remark 3** If the penalty function $h$ is defined as following,
\[
h(f_{ij}) = \begin{cases} 
0 & \text{if } f_{ij} \leq u_{ij} \\
c_{ij} + M(f_{ij} - u_{ij}) & \text{if } f_{ij} > u_{ij}
\end{cases}
\]
Program (15) 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.

**5 Conclusion**
This paper gives combinatorial algorithms 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 by introducing the conception of equilibrium pseudo-flow, we design a combinatorial algorithm for the multi-commodity flow problem. Besides, a non-linear description of the multi-commodity flow problem is given, whose solution is 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.
A Combinatorial Algorithm for the Multi-commodity Flow Problem

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

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

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

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

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

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

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

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

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

Table 1: Basic Network Notation

| $G(V, A)$ | a directed graph |
| $V$ | node (index) set |
| $A$ | arc (index) set |
| $K$ | set of commodities |
| $f^k_{ij}$ | flow of commodity $k$ on arc $(i, j)$, $f = (\cdots, f^k_{ij}, \cdots)$ |
| $f_{ij}$ | flow on arc $(i, j)$, i.e. $f_{ij} = \sum_{k \in K} f^k_{ij}$ |
| $(s_k, t_k, d_k)$ | $s_k$ and $t_k$ are the origin and destination of commodity $k$, and $d_k$ is the demand |
| $u_{ij}$ | the capacity of arc $(i, j)$ |
| $\delta^+(i)$ | $\{j | (i, j) \in A\}$ |
| $\delta^-(i)$ | $\{j | (j, i) \in A\}$ |
| $h(f_{ij})$ | penalty function of arc $(i, j)$ |
| $e_{ij}$ | the cost of arc $(i, j)$ in the residual network $G(f, s_k, t_k)$ |