International Journal of Management Science
Vol 15, No 1, May 2009

Polynomial Time Algorithms for Solving the Multicommodity Flow Problems on Two Types of Directed Cycles *

Young-Soo Myung**

Department of Business Administration, Dankook University, Cheonan, Chungnam 330-714, Korea

(Received: August 15, 2008 / Revised: December 26, 2008 / Accepted: January 6, 2009)

ABSTRACT

This paper considers the two kinds of integer multicommodity flow problems, a feasibility problem and a maximization problem, on two types of directed cycles, a unidirectional and a bidirectional cycle. Both multicommodity flow problems on an undirected cycle have been dealt with by many researchers and it is known that each problems can be solved by a polynomial time algorithm. However, we don't find any result on the directed cycles. Here we show that we can also solve both problems for a unidirectional and a bidirectional cycle in polynomial time.

Keywords: Multicommodity Integer Flows, Cycles, Polynomial Time Algorithm

1. Introduction

A directed multicommodity flow is defined on a directed graph $D = (V, A)$ with a nonnegative capacity $c(a)$ for each arc $a \in A$. We are also given a set of source-sink pairs (s_i, t_i), $i \in K$ where K is the index set of source-sink pairs and a commodity is defined for each source-sink pair. A flow from s_i to t_i is the flow of commodity i for each $i \in K$ and the flows of all commodities should satisfy *the capacity constraint* that restricts the sum of flows routed through an arc under the capacity of the arc. Similarly, an undirected multicommodity flow is defined on an undirected graph $G = (V, E)$ with a nonnegative capacity $c(e)$ for each edge $e \in E$. The capacity constraint in an undirected multicommodity flow limits the sum of flows routed through an edge in both directions to the capacity of the edge. Therefore, it does not make a difference whichever

* The present research was conducted by the research fund of Dankook University in 2006.
** Corresponding author, E- mail: myung@dankook.ac.kr

node in a source-sink pair is a source node. So, we use $\{s_i, t_i\}$ $(s_i < t_i)$, $i \in K$ to denote a set of source-sink pairs in an undirected graph. We assume that the capacities and demands are all integers.

Multicommodity flow problems are well known topics in combinatorial optimization and have good applications such as routing problems in telecommunication networks and the design of VLSI circuits [11]. Among many problems, two basic ones, a feasibility problem and a maximization problem, have received much research attention. In a feasibility problem, a nonnegative demand $d(i)$ is given for each source-sink pair (s_i, t_i) $(\{s_i, t_i\})$, $i \in K$. Given c and d, we will say a multicommodity flow is feasible, if the flow of commodity i is $d(i)$ for each $i \in K$ and all flows satisfy the capacity constraint. Then a feasible problem is to find a feasible multicommodity flow. In a maximization problem, no demand is specified and the goal is to maximize the sum of all flows subject to the capacity constraint. We will call a feasibilty problem simply the multicommodity flow problem (MFP) and a maximization problem the maximum MFP. If we require an integer flow in each of the two problems, we will call the problems the integer multicommodity flow problem (IMFP) and the maximum IMFP, respectively. We sometimes add the adjective 'fractional,' to clarify that no integrality is required. A feasibility problem can be solved using an algorithm for a maximization problem. For example, the MFP on a directed graph can be reduced to the maximum MFP on a directed graph that is extended from the original graph by adding a new source node s_i' and an arc of capacity $d(i)$ from s_i' to s_i for each $i \in K$. Therefore, in the resulting maximum MFP, a source-sink pair (s_i', t_i) replaces (s_i, t_i) for each $i \in K$ and the MFP on the original graph is feasible if and only if the maximization problem has the maximum objective value of $\sum_{i \in K} d(i)$. The MFP on an undirected graph can be reduced to the maximum MFP in the same way. However, in many cases, for example, when a given graph has a simple structure, it is better to develop an algorithm to exclusively solve a feasibility problem, since the reduced graph no longer has the specific structure of the original graph. For more details on the definition of multicommodity flow problems, refer to [11].

The fractional multicommodity flow problems can be formulated as a linear programming (LP) model and the integer ones as an integer programming (IP) model. An LP algorithm can solve the fractional flow problems in polynomial time but an IP algorithm can not do in polynomial time in general. However, when a given graph has a specific structure, we may expect an efficient algorithm other than a general

purpose IP algorithm that solves the IMFP and the maximum IMFP in polynomial time. A *cut* is a useful tool to check the feasibility of a multicommodity flow in both undirected and directed graphs. Given an undirected graph $G = (V, E)$ and a set S of vertices, $\delta(S)$ represents the set of edges in E with exactly one endpoint in S and $\delta(S)$ is defined as a cut. For a digraph $D = (V, A)$ and a set $S \subseteq V$, $\delta^-(S)$ denotes the set of arcs $\{(i, j) \in A: i \notin S, j \in S\}$ and $\delta^-(S)$ is a cut. For an undirected graph $G = (V, E)$ and a set $S \subseteq V$, let $d(S)$ denote $\sum\{d(i): |\{s_i, t_i\} \cap S| = 1, i \in K\}$. Similarly, for a digraph $D = (V, A)$ and a set $S \subseteq V$, we define $d(S) = \sum\{d(i): |s_i \notin S, t_i \in S\}$. If M is an edge set or an arc set, we denote $\sum_{i \in N} c(i)$ for $N \subseteq M$ by $c(N)$. It is well known that every multicommodity flow satisfies the following cut condition: $c(\delta(S)) \geq d(S)$, in an undirected graph, and $c(\delta^-(S)) \geq d(S)$, in a directed graph. It is well known that the computational complexity of multicommodity flow problems of a given graph is closely related with whether the cut condition is sufficient for the existence of a fractional, a half-integer, or an integer flow on the graph [11].

In this paper, we consider multicommodity flow problems on two types of directed cycle graphs, a unidirectional cycle and a bidirectional cycle. In a directed cycle, we have a node set $V = \{1, 2, \cdots, n\}$ where the indices of nodes are counted modulo n. We say that an arc $(i, i+1)$ for $i = 1, \cdots, n$ is a clockwise arc and has a clockwise direction. On the other hand, we say $(i+1, i)$ for $i = 1, \cdots, n$ is a counterclockwise arc and has counterclockwise direction. Let $A^+ = \{(1, 2), (2, 3), \cdots, (n-1, n), (n, 1)\}$ and $A^- = \{(1, n), (n, n-1), \cdots, (3, 2), (2, 1)\}$. Then a unidirectional cycle is a directed graph $G = (V, A)$ with $A = A^+$ or $A = A^-$, i.e., a directed cycle whose arcs have the same direction. If we do not specify the type of an arc set, we assume that a unidirectional cycle is $D = (V, A)$ with $A = A^+$. A bidirectional cycle is a directed graph $D = (V, A)$ with $A = A^+ \cup A^-$. An undirected cycle is an undirected graph $G = (V, E)$ with an edge set $E = \{\{1, 2\}, \{2, 3\}, \cdots, \{n-1, n\}, \{n, 1\}\}$. The demand between s_i and t_i can be routed in either of the two directions, clockwise and counterclockwise. In an undirected cycle, we say that a flow is routed in the *clockwise* (*counterclockwise*) direction if a flow passes through the node sequence $\{s_i, s_{i+1}, \cdots, t_{i-1}, t_i\}$ ($\{s_i, s_{i-1}, \cdots, t_{i+1}, t_i\}$). In a directed cycle, we say that a flow is routed in the clockwise (counterclockwise) direction if a flow passes through clockwise (counterclockwise) arcs.

In many of the multicommodity flow applications, the network models assume cycle graphs. For example, the Synchronous Optical Network (SONET) has a structure of a cycle graph and various multicommodity flow problems arouse in the de-

sign and the operation of SONET [2, 4, 5, 6, 10, 13, 14, 15]. When focusing on the voice traffic, the SONET rings are modeled as undirected cycles but when considering the data traffic, either unidirectional or bidirectional cycles are more suitable SONET models [14]. Other examples of cycle graphs are also found in the design of VLSI circuits [3, 11, 12]. In an undirected cycle, the MFP has a half-integer solution if and only if the graph satisfies the cut condition [9]. Based on this nice property, Myung [7] has developed a linear time algorithm solving the IMFP on an undirected cycle. Myung [8] also showed that the maximum IMFP on an undirected cycle can be solved in polynomial time. However no known researches are found for the IMFP and the maximum IMFP on directed cycles. Note that the MFP and the IMFP on a unidirectional cycle can be trivially solved since a flow from s_i to t_i for each $i \in K$ is determined in a unique way. However, the situation is not so simple in the remaining cases.

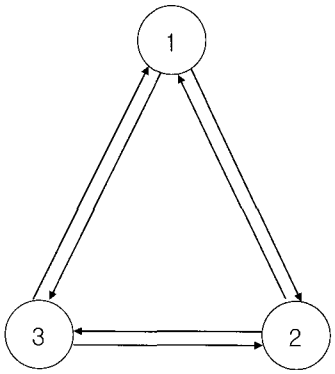
In this paper, we study a couple of issues not cleared yet. First, we check whether the cut condition is sufficient for the existence of a fractional, a half-integer, or an integer flow on a bidirectional cycle, and develop a polynomial time algorithm to solve the IMFP problem. Next, we show that the maximum IMFP on an undirected cycle and that on a bidirectional cycle can be transformed to a maximum IMFP on a unidirectional cycle and develop a polynomial time algorithm to solve the maximum IMFP on a unidirectional cycle

2. Integer multicommodity flow problem on a bidirectional cycle

In this section, we consider the IMFP on a bidirectional cycle. Throughout this section, if we do not specify a given graph, we assume a bidirectional cycle. In an undirected cycle, the cut condition is sufficient for the existence of a fractional multicommodity flow and a half-integer flow. This nice property enables a linear time algorithm to solve the IMFP on an undirected cycle [7]. Here, we show that the cut condition is not sufficient for the existence of a feasible flow on a bidirectional cycle and also show that every feasible (fractional) MFP does not always include a half-integer solution. Although the cut condition does not characterize the feasibility of the MFP, we show that we can solve the IMFP in polynomial time.

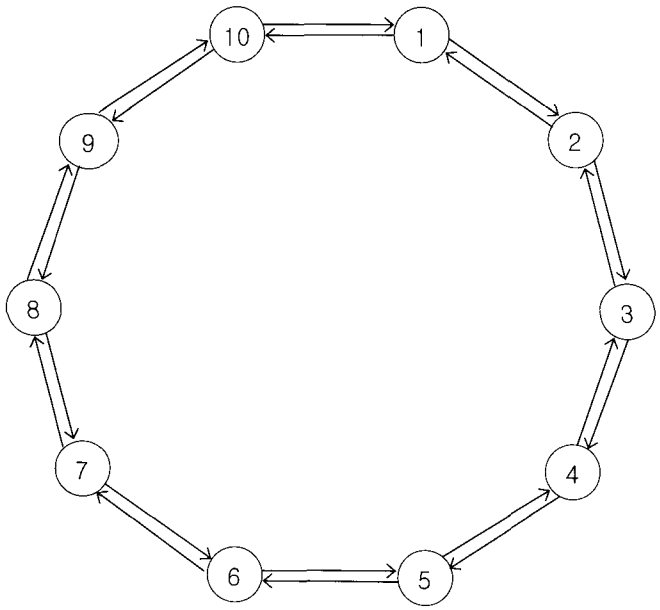
We give two examples of bidirectional cycles, one for showing that the cut condition is not sufficient and the other for showing that a half-integer flow does not al-

ways exist in a feasible MFP. In the first example given in Figure 1, the cut condition holds but no multicommodity flow exists. In the second example given in Figure 2, a unique feasible flow is to route 1/3 in the clockwise direction and to route the remaining 2/3 in the counterclockwise direction for each commodity.



(s_k, t_k)	$d(k)$
(1, 3)	2
(2, 1)	2
(3, 2)	2

Figure 1. A bidirectional cycle where the cut condition holds but no multicommodity flow exists



$$C(a) = \begin{cases} 2, & \forall a \in A^+ \\ 1 & \forall a \in A^- \end{cases}$$

(s_k, t_k)	$d(k)$
(1, 6)	1
(3, 8)	1
(5, 10)	1
(7, 3)	1
(9, 4)	1

Figure 2. A bidirectional cycle where a fractional flow exists but a half-integer flow does not

Now we develop a polynomial time algorithm to solve the IMFP. Recall that we

assume that the capacities and demands are all integers in the IMFP. Wan and Yang [14] developed a polynomial time algorithm for the load balancing problem on a bidirectional cycle where each arc is required to have the same capacity and the objective is to find the smallest arc capacity that enables the existence of a feasible multicommodity flow. We will show that we can solve the IMFP in polynomial time by modifying Wan and Yang's algorithm. Let's describe the feasibility condition in a mathematical way. As a flow between a source-sink pair can be routed in only two directions, one variable is enough to represent the flow. For each $i \in K$, let's define variable $x(i)$ that denotes the amount of the total demand from from s_i to t_i routed in the clockwise direction. Therefore, $d(i)$-$x(i)$ is the amount of the flow routed in the counterclockwise direction. Let $X = \{\mathbf{x} \in R^{|K|} \mid 0 \le x(i) \le d(i) \text{ for each } i \in K\}$, and for a given multicommodity flow $\mathbf{x} \in X$, let $g(\mathbf{x}, a)$ denotes the sum of the flows routed through an arc a. Therefore, $\mathbf{x} \in X$ is a feasible multicommodity flow if $g(\mathbf{x}, a) \le c(a)$ for each $a \in A$. Wan and Yang [14] introduced a semi-integer multicommodity flow defined as a multicommodity flow such that the sum of flows routed in the clockwise direction is an integer. They have shown the following result.

Lemma 1: (Wan and Yang [14]) From any semi-integer multicommodity flow $x \in X$, we can obtain an integer multicommodity flow $\mathbf{y}$ such that $g(\mathbf{y}, a) < g(\mathbf{x}, a) + 1$ for each $a \in A$.

Based on the above observation, we develop an algorithm for the IMFP. As all the capacities are integer values, Lemma 1 implies that if we have a semi-integer solution for the MFP, we can obtain an integer solution, i.e., a solution of the IMFP. Now we describe how to find a semi-integer solution. Consider the following LP problem.

$$\begin{aligned} \text{(P1)} \quad \min \quad & \sum_{i \in K} x(i) \\ s.t. \quad & g(\mathbf{x}, a) \le c(a), \qquad \forall\, a \in A \\ & \mathbf{x} \in X \end{aligned}$$

Let v(P1) be the optimal objective value of (P1). If (P1) has no feasible solution, neither does the IMFP. If v(P1) is integer value, the current solution is a semi-integer multicommodity flow. If $v(P1)$ is fractional value, we add the constraint $\sum_{i \in K} x(i)$

$\geq \lfloor v(\text{P1}) \rfloor$ ($\lfloor m \rfloor$ denotes the least integer greater than or equal to m) and solve the resulting LP problem. Note that if the new LP is feasible, the optimal objective value must be $\lfloor v(\text{P1}) \rfloor$ and the optimal solution is a semi-integer solution. If the new LP has no solution, then (P1) has no semi-integer solution and thus the IMFP is infeasible.

3. Algorithms for the maximum integer multicommodity flow problems

In this section, we consider the maximum IMFP. We first show that we can solve the problem on an undirected cycle and that on a bidirectional cycle by solving an equivalent problem on a unidirectional cycle. First consider the case for a bidirectional cycle. Note that a bidirectional cycle is composed of two unidirectional cycles. It is easy to know that the maximum multicommodity flow of a bidirectional cycle is the sum of the two maximum multicommodity flows on the two unidirectional cycles constituting the bidirectional cycle. For the case of an undirected cycle, a unidirectional cycle having the same maximum multicommodity flow can be constructed as follows. Given an undirected cycle $G = (V, E)$ with $V = \{1, 2, \cdots, n\}$ and $E = \{\{1, 2\}, \{2, 3\}, \cdots, \{n\text{-}1, n\}, \{n, 1\}\}$, we associate a unidirectional cycle, $D = (V, A)$ with $A = \{(1, 2), (2, 3), \cdots, (n\text{-}1, n), (n, 1)\}$. In the constructed unidirectional cycle, the capacity of an arc $(i, i\text{+}1)$ for $i = 1, \cdots, n$ is set equal to the capacity of an edge $\{i, i\text{+}1\}$ and two commodities (s_i, t_i) and (t_i, s_i) are associated with each commodity $\{s_i, t_i\}$, $i \in K$. Note that a clockwise (counterclockwise) directional flow between s_i and t_i in the given undirected cycle corresponds to a flow from s_i to t_i (t_i to s_i) in the associated unidirectional cycle.

Now we focus on an algorithm to solve the maximum IMFP on a unidirectional cycle. From now on, if we do not specify a given graph, we assume an unidirectional cycle. Let $x(i)$ be the amount of flow from s_i to t_i for each $i \in K$ and let $g(\mathbf{x}, a)$ denote the sum of the flows routed through an arc a. Then the maximum IMFP can be formulated as follows:

$$
\begin{aligned}
(\text{P2}) \quad \max \quad & \sum_{i \in K} x(i) \\
s.t. \quad & g(\mathbf{x}, a) \leq c(a), \qquad \forall\, a \in A \\
& \mathbf{x} \geq \mathbf{0} \text{ and integer}
\end{aligned}
$$

For any pair of source-sink pairs (s_i, t_i) and (s_k, t_k) for $i, k \in K$, if $s_i \leq s_k < t_k \leq t_i$, then any flow from s_i to t_i can be replaced by the flow from s_k to t_k without violating the capacity constraint. We call (s_i, t_i) for $i \in K$ a *dominated* source-sink pair if there exists (s_k, t_k) for $k \in K$ such that $s_i \leq s_k < t_k \leq t_i$. So, all the dominated source-sink pairs can be deleted without affecting the optimal solution of the maximum IMFP. If we assume that no dominated source-sink pair is in K and source-sink pairs are sorted such that $s_1 \leq s_2 \leq \cdots \leq s_{|K|}$, then the constraint matrix of (P2) is a 0-1 matrix called a row circular matrix where the 1's in each row appear consecutively (the first and the last columns are considered to be consecutive).

As a direct consequence of Bartholdi *et al.* [1], the optimal solution of (P2) can be obtained as follows.

Lemma 2: Let $\mathbf{x}^*$ is an optimal solution of the LP relaxation of (P2) where the integer restriction of the variables is removed. Then the optimal solution of (P2) is as follows: $x(1) = \lfloor x^*(1) \rfloor$ and $x(i) = \lfloor x^*(1) + \cdots + x^*(i\text{-}1) \rfloor$ for $i = 2, \cdots, |K|$.

For the maximum IMFP on an undirected cycle, Myung [8] developed a similar procedure to solve the problem.

References

[1] Bartholdi, J. J., J. B. Orlin, and H. D. Ratliff, "Cyclic scheduling via integer programs with circular ones," *Operations Research* 28 (1980), 1074-1085.

[2] Cosares, S., N. D. Deutch, I. Saniee, and O. J. Wasem, "SONET toolkit: A decision support system for designing robust and cost-effective fiber-optic networks," *Interfaces* 25 (1995), 20-40.

[3] Frank, A., T. Nishizeki, N. Saito, H. Suzuki, and E. Tardos, "Algorithms for routing around a rectangle," *Discrete Applied Mathematics* 40 (1992), 363-378.

[4] Myung, Y.-S., H.-G. Kim, and D.-W. Tcha, "Optimal load balancing on SONET bidirectional rings," *Operations Research* 45 (1997), 148-152.

[5] Myung, Y.-S., "An efficient algorithm for the ring loading problem with integer demand splitting," *SIAM J. Discrete Mathematics* 14 (2001), 291-298.

[6] Myung, Y.-S. and H.-G. Kim, "On the ring loading problem with demand splitting," *Operations Research Letters* 32 (2004), 167-173.

[7] Myung, Y.-S., "Multicommodity flows in cycle graphs," *Discrete Applied Mathematics* 154 (2006), 1615-1621.

[8] Myung, Y.-S., "Algorithms for maximum integer multiflow and multicut in a ring network," *Journal of the KORMS* 32 (2007), 89-97.

[9] Okamura, H. and P. D. Seymour, "Multicommodity flows in planar graphs," *Journal of Combinatorial Theory Series B* 31 (1981), 75-81.

[10] Schrijver, A., P. Seymour, and P. Winkler, "The ring loading problem," *SIAM J. Discrete Math* 11 (1998), 1-14.

[11] Schrijver, A., "Combinatorial Optimization," Springer, Berlin, 2003.

[12] Suzuki, H., A. Ishiguro, and T. Nishizeki, "Variable-priority queue and doughnut routing," *Journal of Algorithms* 13 (1992), 606-635.

[13] Vachani, R., A. Shulman, and P. Kubat, "Multicommodity flows in ring networks," *INFORMS Journal on Computing* 8 (1996), 235-242.

[14] Wan, P.-J. and Y. Yang, "Load-balanced routing in counter rotated SONET rings," *Networks* 35 (2000), 279-286.

[15] Wang, B.-F., "Linear time algorithm for the ring loading problem with demand splitting," *Journal of Algorithms* 54 (2005), 45-57.