ON EQUIVALENT REPRESENTATIONS OF CERTAIN MULTICOMMODITY NETWORKS AS SINGLE COMMODITY FLOW PROBLEMS

James R. EVANS

University of Cincinnati, Cincinnati, OH, U.S.A.

Received 10 January 1977
Revised manuscript received 6 September 1977

Several classes of multicommodity networks have been shown to have the property that they can be transformed to equivalent uncapacitated single commodity flow problems. We show that many of these networks can be further reduced to smaller, semi-capacitated flow problems using the inverse of a result of Ford and Fulkerson. This appears to be a useful computationally-oriented tool for developing practically efficient algorithms. These concepts are also used to establish a generalization of a previous result concerning multicommodity transportation problems.

Key words: Network Flows, Multicommodity Networks.

1. Introduction

Several classes of multicommodity network flow problems have been shown to have the property that they can be transformed into equivalent, single commodity, uncapacitated flow problems [4,6]. Recently, these results have been synthesized and generalized to more arbitrary network structures [5]. In [3], the author has shown that one of these classes, namely m-source, 2-sink, r-commodity transportation problems \( m, r \geq 2 \) can be reduced to more "compact" representations as ordinary capacitated transportation problems. In this paper we provide a unifying theory for this result which applies to more general cases, and generalize the result in [3] to a larger class of multicommodity transportation problems.

We are not claiming any new or significant theoretical results; rather, the purpose of this discussion is oriented toward developing practically efficient algorithms for multicommodity flow problems. Some computational aspects are presented along with potential applications.

2. Transformations of capacitated networks

The key observation is simple and depends on "inverting" a well-known result due to Ford and Fulkerson, namely, that any capacitated single commodity network flow problem can be transformed into an equivalent, uncapacitated flow
problem [7, pp. 127–130]. In order to effectively use this result, we provide an alternate characterization. Consider the single-commodity capacitated flow problem on a network $G = (N, E)$.

**CFP:**

\[
\begin{align*}
\text{minimize} & \quad cx, \\
\text{subject to} & \quad Ax = b, \quad 0 \leq x \leq u
\end{align*}
\]

where $x_{ij}$ is the flow from node $i$ to node $j$ with unit cost $c_{ij}$; $A$ is the node-arc incidence matrix of the network, $u$ is a vector of arc capacities, and $b$ represents the supply-demand vector at the nodes. Consider a typical arc $(i, j)$ as in Fig. 1. Suppose that this arc is “split” into two arcs as shown in Fig. 2, where $s_{ij}$ is the slack variable associated with the constraint $x_{ij} \leq u_{ij}$. The resulting network $G'$ is bipartite since, by construction, every elementary cycle has an even number of arcs. Also, each conservation of flow constraint in $G'$ is a $(+1, -1, 0)$ linear combination of the constraints of CFP. For nodes of the form $(ij)$ we have $x_{ij} + s_{ij} = u_{ij}$ explicitly. For all other nodes in $G'$ we have

\[
\sum_{j \in N} x_{ij} + \sum_{j \in N} s_{ji} = \sum_{j \in N} u_{ji} + b_i \quad (1)
\]

which is simply the sum of

\[
\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = b_i \quad (2)
\]

and

\[
\sum_{j \in N} x_{ij} + \sum_{j \in N} s_{ji} = \sum_{j \in N} u_{ji}. \quad (3)
\]

In (1) and (2), $b_i$ is the supply/demand at node $i$, with a positive value corresponding to a supply. One readily verifies that this is precisely the same construction as given in [7].

Now let us consider the inverse transformation. Suppose one is given an uncapacitated flow problem in which some node $j$ has in-degree, denoted $\delta_j^-$, of exactly two (Fig. 3) and demand $d_j$. Then the two arcs can be replaced by a single capacitated arc as shown in Fig. 4. The slack on this arc is simply the value of the other flow variable. This extremely simple observation could conserve storage and exploit upper-bounded capabilities of current network codes [1, 2, 8, 9] for many large network problems.

![Fig. 1.](image1)

![Fig. 2.](image2)