MULTI-COMMODITY DISCONNECTING SETS*†

M. BEILMORE,‡ H. J. GREENBERG,§ AND J. J. JARVIS‖ ¶

A comparison of single-commodity versus multi-commodity networks is provided. Also, two combinatorial algorithms are presented for finding \( D_0 \), the minimal disconnecting set.

Consider a network, \( G \), consisting of a set of nodes, \( N \), and a set of arcs, \( A \). Associated with each arc, \( a \in A \) is a positive integer, \( c(a) \) referred to as its arc capacity. The capacity of any set of arcs, \( D \), is \( C(D) = \sum_{a \in D} c(a) \). We define \( F \) as a vector of flows defined on the arcs in \( G \) and \( F_0 \) as a vector of maximal flows. The value of any flow, \( F_0 \), in \( G \) is \( V(F) \).

In addition, there are two subsets of nodes, \( S = \{ s_1, s_2, \ldots, s_r \} \) and \( T = \{ t_1, t_2, \ldots, t_r \} \). The \( j \)th source is \( s_j \) and the \( j \)th sink is \( t_j \). A set of arcs, \( D_j \), is called a multi-commodity disconnecting set if when all the arcs of \( D_j \) are removed from \( G \) no directed chains exist from \( s_j \) to \( t_j \), for \( j = 1, 2, \ldots, r \). Let \( D_0 \) be a multi-commodity disconnecting set with minimal capacity.

In the case \( r = 1 \), i.e. single commodity flow, \( D_0 \) becomes a cut-set. It is also well known that in this case the mathematical programming formulation for the minimum cut-set is the linear programming dual of the formulation for the maximum flow in the network. However, this duality relationship is not retained, in general, for multi-commodity networks.

Disconnecting sets per se are of importance in networks. Suppose one sets the capacity of an arc equal to the cost of destroying that arc by an attacker. Then, if an attacker desires to completely neutralize the effectiveness of the system, his optimal strategy would be to destroy the arcs of \( D_0 \). Areas of application might be in transportation, communications, or pipeline networks.

Differences

A modification of an example which appears in Ford and Fulkerson [3] is shown in Figure 1. This example illustrates some of the differences between single-commodity and multi-commodity networks. In this case both \( F_0 \) and \( D_0 \) are unique.

Observations from this example which have been well known for some time are: (1) \( F_0 \) may not be integer, and (2) \( V(F_0) \neq C(D_0) \) in general. Further, the example illustrates the fact that all arcs of \( D_0 \) need not be saturated (flow equal to capacity).

* Received October 1968; revised July 1969.
† This research was supported by the National Bureau of Standards under contract CST 348.
‡ The Johns Hopkins University
§ The Southern Methodist University
‖ The Georgia Institute of Technology
¶ The authors wish to thank the referees for their helpful comments.
1 See Ford and Fulkerson [3], page 26.
2 Hu [6] has proven a max-flow min-cut theorem for certain classes of two-commodity networks. Rothschild and Whinston [8] have shown equality for certain classes of \( r \)-commodity networks (\( r \) any positive integer). However counterexamples do exist for the general case.
3 This is true even if \( F_0 \) is integer. To see this double all capacities in Figure 1 yielding \( V(F_0) = 9 \) and \( C(D_0) = 10 \).
Figure 1. An example of three-commodity flow. The numbers beside an arc represent its capacity.

Figure 2. A two-commodity network and its "apparent" equivalent network.

Hence, although it can be shown that $C(D_0) \geq V(F_0)$ the maximal flow provides little other information about $D_0$.

Jewell [7] shows that undirected arcs cannot be handled by two parallel directed arcs with respect to the maximal flow problem in a multi-commodity network. This is also true with respect to the minimal disconnecting set problem. For the network of Figure 2, $D_0 = \{a_1\}$ with $C(D_0) = 3$. If the undirected arcs were replaced by parallel directed arcs the final result would be $D_0 = \{a_1, a_3\}$ with $C(D_0) = 4$.

One might be tempted, in finding the minimal disconnecting set, to create a related single-commodity network by appending a "supreme source," $u$, and arcs of infinite capacity connecting $u$ to $s_j$ for $j = 1, 2, \cdots, r$. Additionally, one would add a "supreme sink," $v$, and arcs of infinite capacity connecting $t_j$ to $v$, for $j = 1, 2, \cdots, r$. If one performs such a transformation on the graph of Figure 1 the "supreme cut" would be $\{a_1, a_3, a_5\}$ with capacity of 8, which is not equal to that of $D_0$. However, the "supreme cut" does provide an upper bound on the capacity of $D_0$ since it is a multi-commodity disconnecting set.
Figure 3. A two-commodity counterexample

Figure 3 illustrates a graph where the minimal disconnecting set, $D_0$, is not equal to the union of $r$ individual single-commodity minimum cut-sets.

However, Robacker [10] shows that $D_0$ is a union of $r$ single-commodity cut-sets (not necessarily minimal).

Two algorithms will now be presented which produce the minimal disconnecting set for any multi-commodity network. To aid in the description of the succeeding algorithms we define a 0-1 variable $d_i$ associated with arc $i$ such that $d_i = 1$ if arc $i$ is to be cut and $d_i = 0$ otherwise.

**Combinatorial Flow Approach**

This section presents an implicit enumeration algorithm which implicitly enumerates all subsets of the arcs in $G$ to find $D_0$. The algorithm utilizes the property that $D_0$ is a union of single commodity cut sets to speed convergence. The algorithm is a version of the Geoffrion [4] Implicit Enumeration Algorithm which has been set up to take advantage of the network structure of this particular problem. $D^*$ is the best disconnecting set enumerated so far by the algorithm. At termination $D^*$ will be $D_0$, the minimal multi-commodity disconnecting set. $S^k$, an ordered vector of integers, with $M$ (the number of arcs in $G$) or less components will represent a partial solution at iteration $k$. If a component of $S^k$ is $+i$, then this implies $d_i = 1$ in the partial solution (i.e. arc $i$ is a member of the disconnecting set); if a component is $-i$, then $d_i = 0$ in the partial solution (i.e. arc $i$ is not a member of the disconnecting set).

The remaining arcs are not committed, i.e. they are “free.” A characteristic of the algorithm is that only one vector $S^k$ need be saved at any one time, hence in a computer code, storage for only one $M$ component vector is required for $S^k$.

The Algorithm becomes:

**Step 0 (Initial Solution):** Solve for the “supreme cut” and initialize $D^*$ as the set of arcs in this cut-set. Initialize the partial solution, $S^0$, as follows. If arc $j$ is in $D^*$ augment $j$ to $S^0$ as a positive entry. It should be noted that $S^0$ is fathomed. Set $k = 0$.

**Step 1 (Backtracking):** If $S^k$ is empty, STOP as $D_0 = D^*$. Otherwise, suppose $S^k = (i_1, \ldots, i_p)$. If $i_p < 0$, set $S^{k+1} = (i_1, \ldots, i_{p-1})$ and repeat step 1 with $k = k + 1$ ($i_p$ is now free). If $i_p > 0$, set $S^{k+1} = (i_1, \ldots, i_{p-1}, -i_p)$. Go to step 2 with $k = k + 1$ and $j = 1$.

**Step 2 (Feasibility Drive):** Let $I(S^k)$ be the set of arcs with positive entries in $S^k$, and let $E(S^k)$ be the set of arcs with negative entries in $S^k$. $C(I(S^k))$ is the capacity of $I(S^k)$. Temporarily, set the capacity of each arc in $I(S^k)$ to 0, and temporarily set the capacity of each arc in $E(S^k)$ equal to infinity. All free arcs have their original capacity. Find the single-commodity, minimal cut-set, denoted by $A_j$, which separates $s_j$ from $t_j$. Let $H = A_j - I(S^k) \cap A_j$. If $C(I(S^k)) + C(H) \geq C(D^*)$, go to step 1 as $S^k$.

---

4 The notation used to describe the algorithm is that of Geoffrion [4].
is fathomed, since no completion of $S^k$ could be better than $D^*$. Otherwise, set $S^{k+1} = (S^k, H)$. If $j = r$, set $D^* = I(S^{k+1}); k = k + 1$ and go to step 1. On the other hand, if $j < r$ set $k = k + 1; j = j + 1$ and repeat step 2.

This algorithm generalizes in a straightforward manner to the case where an adversary cannot afford to disconnect all sinks from their respective sources. One may now wish to know the minimum multi-commodity disconnecting set which disconnects only those sinks in $T^*$ (a subset of $T$). If $T^*$ is prescribed, then this is another multi-commodity problem as already described and $T$ is merely redefined to be $T^*$. However, suppose $T^*$ is not prescribed but carries the restriction that $R(T^*) \geq V_0$, where $R(I)$ is a set function describing a return of value $R(I)$ given that the commodities in $I$ are disconnected, and $I \subset \{1, 2, \ldots r\}$.

Let us generalize somewhat the definition of a multi-commodity disconnecting set to allow this problem. A multi-commodity disconnecting set, $D^{(I)}$, relative to a set of commodities, $I$, is a set of arcs which, when removed from the graph, disconnects all commodities in $I$.

We shall assume that $R(I)$ is monotonic in that if $I_1 \subset I_2$, then $R(I_1) \leq R(I_2)$. 

**Step 1'** (Backtracking): If $\cdots$. Go to step 2' with $k = k + 1, j = 1$, and $T^+ = \emptyset$ (the empty set).

**Step 2'** (Feasibility Drive): Let $\cdots$. Let $H = A_j - I(S^k) \cap A_j$. If $C(S^k) + C(H) \geq C(D^*)$ go to step 3'; otherwise, go to step 4.'

**Step 3'** (Updating Skip Set): If $j = r$, go to step 1' as $S^k$ is fathomed. Otherwise, update the skip set by setting $T^+ = \{T^+, j\}$. Let $R^* = R(T - T^+)$, which is an upper bound on the maximum return achieved by completions of $S^k$. If $R^* < V_0$ then go to step 1' as $S^k$ has been fathomed; otherwise, set $j = j + 1$ and go to step 2'.

**Step 4'** (Updating $S^k$): Update the partial solution by setting $S^{k+1} = \{S_k, H\}$. Define $T^0$ as the set of commodities which is cut by the partial solution $S^{k+1}$. If $R(T^0) \geq V_0$, set $D^* = I(S^{k+1}); k = k + 1$ and go to step 1' if $R(T^0) < V_0$ and $j < r$ then set $j = j + 1; k = k + 1$ and go to step 2'. Finally, if $R(T^0) < V_0$ and $j = r$, then set $k = k + 1$ and go to step 1'.

The modified combinatorial flow algorithm has been coded in FORTRAN IV for the IBM 7094 and computer times for two test problems have been encouraging. Problem 1 had 90 directed arcs, 35 nodes, 5 commodities and $R(t_j) = 1$ for all $j$ A complete parametric analysis of $V_0$ over its range, viz. 0, 5, required less than 10 min. The same problem was solved for $V_0 = R(T) = 5$ in less than 2 minutes (when $V_0 = R(T)$, then the original minimum disconnecting set problem is solved). Problem 2 contained 68 directed arcs, 33 nodes, 5 commodities, and $R(t_j) = 1$ for all $t_j$. The algorithm obtained the optimal solution for $V_0 = 5$ in 5.32 minutes of execution time.

**Arc-Chain Formulation**

This formulation is based on the fact that a disconnecting set must cut at least one arc, $a_i$, of every directed chain between each source and its respective sink. Thus, the problem becomes one of finding that set of arcs, $D_0$, with minimum capacity which contains at least one arc of every directed chain. Before stating this as a mathematical program we consider a more general problem.

---

4 Note that for this $S^k$ we have $C(H) = C(A)$ since $H$ is the set $A_j$ minus those arcs which presently have capacity equal to 0.

5 $T^+$ is called the “Skip Set” and is the set of commodities $l < j$ which have not been cut when solving for the cut-set $A_j$ in step 2'. During step 2', once a commodity is skipped it will never be cut for that value of $k$. 
Suppose we have associated with each arc $a \in A$ a number $l(a) \geq 0$ defined as the length of the arc. The length of a directed chain will be taken as the sum of the lengths of the arcs in the chain. In many real situations one would like to restrict his attention to chains which are not "too long". We define an "effective" chain as a directed chain from $s_i$ to $t_j$ whose length is less than or equal to some prescribed maximum acceptable length, $L$. Suppose there are $W_L$ such effective chains, $P_1, P_2, \ldots, P_{W_L}$.

A set of arcs which when removed from $G$ will cut all chains from $s_i$ to $t_j$, $j = 1, 2, \ldots, r$ of length $L$ or less, i.e. effective chains, is defined to be a disconnecting $L$-set, and is denoted $D^L$. $D_0^L$ is the minimal disconnecting $L$-set. One may find $D_0$ as a special case; i.e. take $l(a) = 1$ for all $a \in A$, and $L = \infty$.

Using the 0-1 variable, $d_i$, described earlier, the arc-chain formulation for the minimum disconnecting $L$-set becomes:

$$\min \sum_{a \in A} c(a_i) d_i$$

s.t. $\sum_{a \in P_j} d_i \geq 1$ for $j = 1, 2, \ldots, W_L$, $d_i = 0, 1$.

Consider for the moment, the case when $L = \infty$. Here one seeks the minimum disconnecting set which disconnects all chains between each source and its respective sink. S. Okada [9] has shown that for a single commodity network with undirected arcs there can be as many as $2^{m-n+1}$ chains where $m$ is the number of arcs and $n$ is the number of nodes. For multi-commodity undirected networks this upper limit is approximately $r2^{m-n+1}$, where $r$ is the number of commodities.

It is obvious that for reasonable size networks one cannot enumerate all the chains. For example, suppose $m = 150$, $n = 50$, and $r = 5$. Then, there could be approximately $(5)2^{101} > 10^{50}$ chains in the network. It is clear that any algorithm which uses chains as constraints cannot require the explicit enumeration of all chains in the network and still be expected to solve large problems. An iterative algorithm will now be described which only implicitly enumerates all the chains in order to obtain $D_0^L$.

Denote by $H(k)$ the set of the chains (not by any means all of the chains) which have been enumerated at iteration $k$, $k = 0, 1, \ldots$. Initially, $H(0) = \emptyset$, the empty set. Begin with $k = 0$ and

**Step 1.** Find a set $D_0^L(k)$ which solves

$$\min \sum_{a \in A} c(a_i) d_i$$

s.t. $\sum_{a \in P_j} d_i \geq 1$ for all $P_j \in H(k)$, $d_i = 0, 1$.

**Step 2.** Remove the arcs in $D_0^L(k)$ from the graph $G$. Solve for the shortest chain between each source and its respective sink, (i.e. obtain the shortest chain from $s_i$ to $t_j$ for $j = 1, 2, \ldots, r$).

**Step 3.** If there are no chains or if the length of every shortest chain is greater than $L$ then stop as $D_0^L = D_0^L(k)$ is the optimal disconnecting $L$-set desired. Otherwise, replace the arcs of $D_0^L(k)$ in the graph $G$, and pass to step 4.

**Step 4.** Set $k = k + 1$ and add those chains from step 3 whose lengths are $L$ or less to $H(k)$. Pass to step 1.

The solution obtained by this iterative algorithm is the minimum disconnecting $L$-set, $D_0^L$, since the optimal solution to formulation II is a feasible solution to formulatio

---

7 Actually, $l(a)$ may be negative, as long as there are no cycles with negative length.

8 The length restriction might be interpreted in terms of delays in a transportation network, a limitation on the number of relays in a communications network, etc.
The notation "u" indicates undirected arcs; "d" indicates directed arcs.

In this iterative algorithm chains are enumerated when they bind (restrict) the optimal solution. The integer program which must be solved at each iteration belongs to the special class known as covering problems. Fortunately, the special structure of covering problems makes them easier to solve than general integer programming. Blandini [2] describes four principles of dominance which can be applied to the covering problem to greatly reduce the number of rows and columns of the integer program.

For many of the test problems to which the above algorithm has been applied the reduction techniques have completely solved the integer program (i.e., reduced it to a 0 by 0 problem) at every iteration. Many others were solved in this manner at a majority of their iterations. Nonetheless, there comes a time when the reduction techniques do not solve the integer program.

Two techniques which appear to have promise in solving the reduced integer program are the dual methods of Gomory [5] and others, and the implicit enumeration methods of Geoffrion [4], Balas [1], etc. It has been reported by Balinski [2] that the Gomory cutting plane techniques have been most successful with covering problems.

Concerning the number of chains required—it is felt (and experience with test problems has strengthened this) that probably not more than \( M \) chains (\( M \) being the number of arcs) will be necessary. In fact, the solution of the test problems has required slightly less than \( M/2 \) or less chains.

A FORTRAN IV program for the IBM 7094 was written for the arc-chain algorithm. The program utilized the reduction techniques and an implicit enumeration scheme similar to Geoffrion [4]. A directed 86 arc-20 node-5 commodity problem, where it was desired to find \( D_0 \) (i.e. \( L = \infty \)), required 26 seconds. Table 1 presents running times for other test problems. Problems (Graphs) 1 thru 5 were highly symmetric graphs, which have many alternate optimal solutions and hence represent a difficult class of graphs for any implicit enumeration algorithm. Problems 6 and 7 are identical except that Problem 6 has directed arcs and Problem 7 has undirected arcs. As would be expected, the undirected case is more difficult since it generally has more chains. Problem 8 is the largest problem attempted (5 commodities, 201 nodes, and 470 undirected arcs) and is typical of the graphs which originally motivated this research.

In conclusion, if all commodities are to be disconnected, then both algorithms apply.

\* Note that Problem 5 did not converge in 1200 seconds.
Experience seems to indicate that the arc-chain approach is faster but storage limits problem size. The node-arc approach, on the other hand, may be slower but can solve very large problems. When arc capacities are nearly all equal, the arc-chain approach is very fast because greater advantage is taken of the dominance properties of the integer program. By contrast the node-arc approach is more efficient when capacities differ widely because there is less chance of near optimal solution. Finally, the node-arc approach handles the special case of values on sinks and the arc-chain approach handles the special case of length restrictions on chains.

References