Computers Math. Applic. Vol. 20, No. 1, pp. 29–38, 1990
Printed in Great Britain. All rights reserved

0097-4943/90 \$3.00 + 0.00
Copyright © 1990 Pergamon Press plc

NEW METHODS FOR MULTI-COMMODITY FLOWS

N. Boland and A. I. Mees
Mathematics Department, University of Western Australia, Nedlands, WA 6009, Australia

(*Received* 14 *June* 1989)

Abstract—Multi-commodity flow problems arise naturally in telecommunications and scheduling. Such applications typically involve large network optimization problems with constraints of a form which make standard optimization techniques inapplicable. Linear programming methods can solve only small instances of such problems. New theoretical work by Soviet researchers on network-based techniques is shown in this paper to yield practical algorithms which enable certain types of multi-commodity flow problems to be solved quickly. As these methods generalize efficient single-commodity methods, there is promise that further research will allow more general problems to be solved in an analogous manner.

1. INTRODUCTION

It is well-known that problems involving the flow of a single-commodity in a network can be solved elegantly and efficiently by methods which combine the merits of graph theory and of linear programming. Unfortunately, this does not turn out to be the case in problems involving the flow of several commodities which have to share resources, such as telephone calls or information packets. Indeed, many important properties are lost in the transition from the single-commodity to the multi-commodity case. For example, in the single-commodity case, if capacity limits on arcs in the network are all integers, then typical optimization problems will have solutions in integers also. For the multi-commodity case, this does not hold.

Most approaches to multi-commodity flows which are presently successful have used the ideas of linear programming [1]. While these approaches are useful for problems of modest size, they are unable to handle the sort of large problems that one would like to be able to solve in practice, because the computational difficulty grows too quickly with the size of the problem. On the other hand, there has been rather little success in generalizing network algorithms to the multi-commodity case, although approximate methods are available for some types of problems [2]. Efficient exact algorithms are yet to be developed. Recently Lomonosov [3] has reported on work by himself and his colleagues which seems to the present authors to offer promise in generalizing existing graph-theory based methodology to the multi-commodity case. In this paper we describe some of the theoretical work of Lomonosov and show how it may be implemented to provide an extremely efficient and practical algorithm for solving certain multi-commodity flow problems. We also discuss the limitations of the approach as it stands at present and in particular the restricted class of problems which we are able to solve.

We shall begin by describing the theory behind part of Lomonosov's work. In doing so we show how to develop practical algorithms from the theorems that Lomonosov proves. Then we display an example of a network problem solved by this method with very modest computing resources. Finally, we discuss future developments.

2. MULTI-COMMODITY FLOWS

This section introduces some necessary notation and some basic network and multi-commodity flow concepts.

The following notation is used throughout this paper. For sets X and Y we write:

$[X]$—the set of all unordered pairs $[x, y]$ of $x, y \in X, x \neq y$;
(X)—the set of all ordered pairs (x, y) of $x, y \in X, x \neq y$;
$[X, Y]$—the set of all unordered pairs $[x, y]$ of $x \in X, y \in Y$ where $X \cap Y = \varnothing$;
(X, Y)—similarly for (x, y);
$\mathbb{R}_+$—the set of non-negative real numbers.

In discussing multi-commodity flows, it is convenient to deal with undirected networks. A *network* is defined to be a triple $(V, T; c)$ where V is the set of nodes, $T \subseteq V$ is the set of terminal nodes (nodes which may originate, or be a destination for, flow in the network) and $c:[V] \rightarrow \mathbb{R}_+$ is a capacity function on the set of all possible undirected arcs. There is no arc set defined as such; it may be taken to be the set of arcs on which the capacity function is non-zero. A convenient example to bear in mind is that of a set of telephone exchanges (the nodes) with lines connecting some pairs of exchanges (the arcs). The capacity of an arc in this example would be the number of circuits between the given pair of exchanges.

A commodity flowing in such a network is identified by its source and sink nodes. So a *flow* in a network $(V, T; c)$ is defined to be a function $f_{st}:(V) \rightarrow \mathbb{R}_+$ with the following properties:

$$\sum_{\substack{y \in V \\ y \neq x}} (f_{st}(x, y) - f_{st}(y, x)) = 0, \quad \forall x \in V \backslash \{s, t\}$$

and

$$\sum_{\substack{y \in V \\ y \neq s}} (f_{st}(s, y) - f_{st}(y, s)) = -\sum_{\substack{y \in V \\ y \neq t}} (f_{st}(t, y) - f_{st}(y, t)) = \|f_{st}\|,$$

where $\|f_{st}\|$ is defined by the above and is known as the *magnitude* of the flow.

A multi-commodity flow or *multiflow* F is defined to be a set of flows and the magnitude of the multiflow is given by

$$\|F\| = \sum_{f \in F} \|f\|.$$

It is natural to define some sort of "traffic" function of a multiflow, describing the amount of traffic due to the multiflow in any given arc. This function $\zeta_F:[V] \rightarrow \mathbb{R}_+$ is defined by

$$\zeta_F[x, y] = \sum_{f \in F} (f(x, y) + f(y, x)), \quad \forall [x, y] \in [V].$$

The multi-commodity flow problems solved by Lomonosov appear in the following form. Given a network $(V, T; c)$, find a multiflow $F = \{f_{st}:[s, t] \in [T]\}$ which solves the external multiflow problem (EMF):

maximize

$$\sum_{[s, t] \in [T]} a[s, t] \|f_{st}\|,$$

subject to

$$\zeta_F[x, y] \leqslant c[x, y], \quad \forall [x, y] \in [V],$$

where

$$a:[V] \rightarrow \mathbb{R}_+$$

is the objective function.

Lomonosov shows how to solve such problems for restricted cases of the objective function. Solution involves repeated solution of a subsidiary problem, known as the locking problem, which employs a generalized version of the max-flow algorithm of Ford and Fulkerson [4].

3. LOCKING AND DRAINING

In order to generalize the max-flow algorithm we need to make a few definitions.

A *path* P is a sequence of nodes $P = [x_0, x_1, \ldots, x_n]$. Sometimes we abbreviate this to $P[x_0 \ldots x_n]$. A segment of the path from node x_i to node x_j is denoted by $P[x_i \ldots x_j]$. The concatenation of two paths $P[r \ldots s]$ and $Q[s \ldots t]$ with a common end-point s is written $P[r \ldots s] \cdot Q[s \ldots t]$ and is a path N with endpoints r and t, having the property that $N[r \ldots s] = P$ and $N[s \ldots t] = Q$.

A *cut* in a network $(V, T; c)$ is a set $X \subset V$. The *capacity* of a cut is written $c[X, \bar{X}]$ and is defined:

$$c[X, \bar{X}] = \sum_{[x,y] \in [X, \bar{X}]} c[x, y].$$

Given a network $(V, T; c)$ together with a multiflow F in the network and a set $A \subset T$, the *internal* and *external* flows of F with respect to A are:

$$F_A^i = \{f_{st} \in F : s, t \in A\}\,(\text{internal}),$$

$$F_A^e = \{f_{st} \in F : [s, t] \in [A, T \backslash A]\}\,(\text{external}).$$

Note that as the network is not directed, we may take all flows in F_A^e to be directed from the source node in A to a sink node in $T \backslash A$ (denoted by $\bar{A}$).

The set $A \subset T$ is said to be *locked* by a multiflow F if F_A^e is a maximal multiflow from A to $\bar{A}$, that is if there exists a set $X \subset V$ with $X \cap T = A$ and $\|F_A^e\| = c[X, \bar{X}]$. The *locker* of A, written $\mathscr{L}(A)$, is defined to be the smallest subset of V with this property. Any set X with this property is a minimal capacity cut in the sense that

$$c[X, \bar{X}] = \min\{c[Y, \bar{Y}] : Y \subset V, Y \cap T = A\}.$$

Our objective is to take an intitial flow, and transform it so as to lock A, without diminishing its total magnitude. This is achieved by an operation called *draining*. This operation improves external flows of F with respect to A by sacrificing internal flows. In a telecommunications problem, this process might be reducing internal traffic in an exchange group to free up capacity for external traffic between that group and some other. The mechanism used to achieve this is analogous to the max-flow algorithm of Ford and Fulkerson.

Given a network $(V, T; c)$ a set $A \subset T$ and a multiflow F in the network the *marsh* of A, written $M(A)$, is defined to be:

$$M(A) = A \cup \{x \in V : \zeta_{f_{st}}(x, y) > 0, \exists y \in V, \exists s, t \in A\}.$$

In the exchange group example, the marsh of a group of exchanges consists of the exchanges themselves together with any exchange which is used in the routing of internal calls.

In the draining operation, the analogy to the familiar flow-augmenting path of single commodity network theory is the *active* path. This is defined to be a path $P = [x_0, x_1, \ldots, x_n]$ with the property that for all $i = 1, \ldots, n$, either

$$\zeta_F[x_{i-1}, x_i] < c[x_{i-1}, x_i]$$

or

$$F_A^e(x_i, x_{i-1}) > 0.$$

Thus, each arc of P either has available capacity or carries some external flow from A in the backwards direction. Ford and Fulkerson showed in the single commodity case that a flow is maximal if and only if there is no flow-augmenting path from the source node to the sink node. In our example with exchanges, this means there is no route available for a call to get through. Ford's and Fulkerson's result extends to multiflows in the following way:

Proposition (Lomonosov)

Given a network $[V, T; c)$, a set $A \subset T$ and a multiflow F, F_A^e is a maximal multiflow (i.e. F locks A) if and only if there is no active path from $M(A)$ to $M(\bar{A})$.

Proof. ($\Rightarrow$) Let F lock A and assume all flows in F_A^e are oriented with source in A and sink in $\bar{A}$. Then there some minimal set $\mathscr{L}(A) \subset V$ with $\mathscr{L}(A) \cap T = A$ and $\|F_A^e\| = c[\mathscr{L}(A).\overline{\mathscr{L}(A)}]$. Clearly $M(A) \subseteq \mathscr{L}(A)$ and $M(\bar{A}) \subseteq \overline{\mathscr{L}(A)}$ so that for all $x \in \mathscr{L}(A)$ and $y \in \overline{\mathscr{L}(A)}$, both

$$\zeta_F[x, y] = c[x, y]$$

and

$$F_A^e(y, x) = 0,$$

hold. So there can be no active path from $M(A)$ to $M(\bar{A})$.

($\Leftarrow$) Assume there are no active paths from $M(A)$ to $M(\bar{A})$. Consider the set X of nodes reachable along active paths from $M(A)$. Then clearly $M(A) \subseteq X$ and by the hypothesis, $X \cap M(\bar{A}) = \varnothing$, so $M(\bar{A}) \subseteq \bar{X}$. Let $x \in X$ and $y \in \bar{X}$. Then

$$\zeta_F[x, y] = c[x, y] \quad \text{and} \quad F^e_A(y, x) = 0,$$

since otherwise an active path reaching x could be extended by y. Now $\zeta_{F^i_A}[x, y] = 0$, since $M(A) \subseteq X$, and $\zeta_{F^i_{\bar{A}}}[x, y] = 0$ since $M(\bar{A}) \subseteq \bar{X}$, so since $F = F^e_A \cup F^i_A \cup F^i_{\bar{A}}$,it must be the case that $\zeta_F[x, y] = F^e_A(x, y)$. Hence for all $[x, y] \in [X, \bar{X}]$, $F^e_A(x, y) = \zeta_F[x, y] = c[x, y]$ and so $\| F^e_A \| = c[X, \bar{X}]$. Hence F locks A. □

This proposition demonstrates the strong relationship between active paths and maximal multiflows from subsets of the terminal nodes. But how can active paths be used to augment flow from A to $\bar{A}$? This is achieved by the draining operation described below.

3.1. *The drain procedure*

Given a network with a multiflow F and a subset A of the terminal nodes, the drain procedure uses active paths to improve $\| F^e_A \|$ by sacrificing $\| F^i_A \|$ and $\| F^i_{\bar{A}} \|$. Repeated applications of the drain procedure will transform F so that it locks A without diminishing the total $\| F \|$. The drain procedure works as follows.

Let F be a multiflow in a network $(V, T; c)$ and let $A \subset T$. Assume F does not lock A. Then there exists an active path from $M(A)$ to $M(\bar{A})$: say $P = [p = x_0, x_2, \ldots, x_n = q]$ is an active path with $p \in M(A)$ and $q \in M(\bar{A})$.

Let $f_{st} \in F$ and define a *line* of f_{st} to be the flow of f_{st} along a path $L = [s = x_0, x_1, x_2, \ldots, x_{n-1}, x_n = t]$. So $f_{st}(x_{i-1}, x_i) > 0$, for all $i = 1, \ldots, n$. The *width* of the line is defined to be:

$$w(L) = \min\{f_{st}(x_{i-1}, x_i) : 1 \leqslant i \leqslant n\}.$$

So the width is the maximum amount by which the flow f_{st} can be reduced along L in the draining operation.

Now $p \in M(A)$ means either $p \in A$ or there exists a line of some flow in F^i_A which passes through p. If the latter is the case, call such a line $K = [k_0, k_1, \ldots, k_i = p, \ldots, k_m]$. Similarly $q \in M(\bar{A})$ means either $q \in \bar{A}$ or there is some line of $F^i_{\bar{A}}$ passing through q. Call such a line $L = [l_0, l_1, \ldots, l_j = q, \ldots, l_r]$ (refer to Fig. 1). The first part of the drain procedure is to use the active path to "draw" K up towards q until K passes through q. This is achieved as follows.

Case 1

The first arc of the active path has some residual capacity, i.e. $\zeta_F[x_0, x_1] < c[x_0, x_1]$. In the case K can be "threaded" up to x_1 by setting the new width of K to be:

$$w(K) = \min\{w(K), (c[x_0, x_1] - \zeta_F[x_0, x_1])/2\}$$

and extending K to $[k_0, k_1, \ldots, k_i = x_0, x_1, x_0, k_{i+1}, \ldots, k_m]$ (refer to Fig. 2).

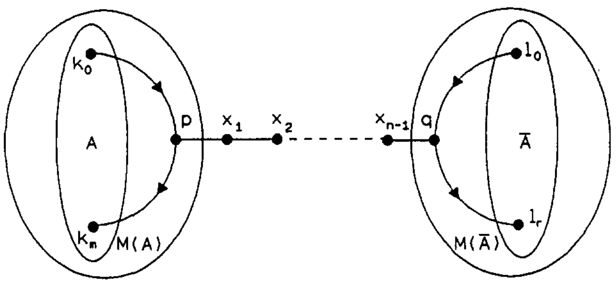

Fig. 1. The general situation prior to a drain procedure showing the active path and the K and L lines.

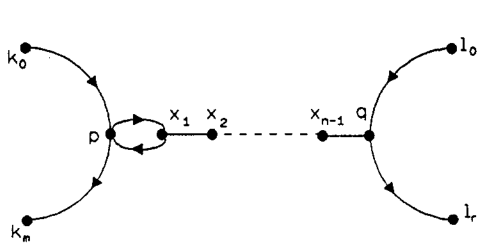

Fig. 2. The K line is "threaded" up towards the L line by utilizing available capacity.

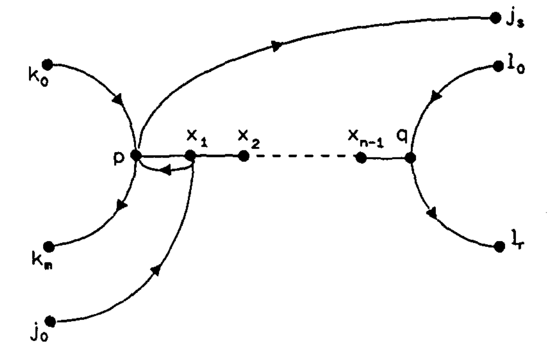

Fig. 3(a). The first arc of the active path has some backwards flow of F^e_A passing along the J line.

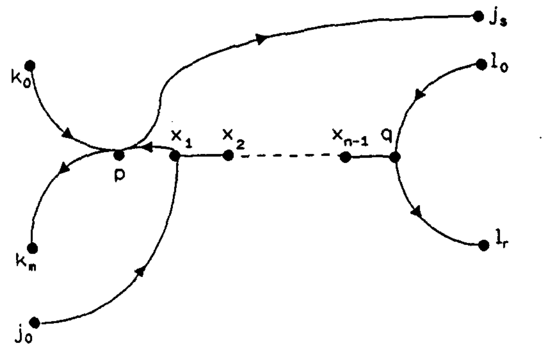

Fig. 3(b). Flow is swapped between the J and K lines. The new K line passes from j_0 to k_m.

Case 2

The first arc of the active path has some backward flow, i.e. $F^e_A(x_1, x_0) > 0$. In this case there must exist a line of some flow in F^e_A which passes through the arc (x_1, x_0): say $J = [j_0, j_1, \ldots, j_i = x_1, j_{i+1} = x_0, \ldots, j_s]$, where $f_{j_0 j_s} \in F^e_A$ is such a line [refer to Fig. 3(a)]. Now K can be "swapped" with J so that the new K would be $[j_0, j_1, \ldots, j_i = x_1, j_{i+1} = x_0 = k_i, k_{i+1}, \ldots, k_m]$ and would have new width $w(K) = \min\{w(K), w(J)\}$, while J would be swapped to $[k_0, k_1, \ldots, k_i = x_0 = j_{i+1}, \ldots, j_s]$, with the same width as the new K [refer to Fig. 3(b)].

In either case, the new K line passes through x_1. This process can be repeated until the active path has degenerated to a single node: until the lines of F^i_A and $F^i_{\bar{A}}$ (the K and L lines, respectively) pass through the common node, q [refer to Fig. 4(a)]. Once this is achieved, the lines of F^i_A and $F^i_{\bar{A}}$ may be drained, to enable F^e_A to be increased. This is performed as follows [refer to Fig. 4(b)].

Set $\epsilon = \min\{w(K), w(L)\}$. Decrease the flow along the K line (this is a flow in F^i_A) by ϵ and decrease the flow along the L line (a flow in $F^i_{\bar{A}}$) by ϵ also. Increase flow from the source of K, along the K line to q, and along the L line to the sink of L by ϵ and increase the flow from the sink of K, backwards along K (remember the network is undirected) to q, and then backwards along the L line to the source of L by ϵ also. Since the terminal nodes of K are in A and the terminal nodes of L are in $\bar{A}$, this process increases $\| F^e_A \|$ by 2ϵ, while decreasing $\| F^i_A \|$ and $\| F^i_{\bar{A}} \|$ by ϵ each. Note that if a special case occurs, such as if $q \in \bar{A}$ so the L line is not needed, one or both of the latter flows may not be decreased. So it is in fact possible to increase $\| F \|$.

It can be shown that if the capacities of the network are integral, with half-integral flows which have an integral traffic function, then the drain operation preserves this situation.

It is clear that repeated applications of the draining operation will result in a multiflow which locks some required subset of the terminal nodes.

3.2. Solving multiflow problems

The drain procedure is useful as it can be shown that certain classes of multi-commodity flow problems can be solved by simultaneously locking a number of subsets of the terminal nodes. Deciding exactly what families of subsets of the terminal nodes are simultaneously lockable is a

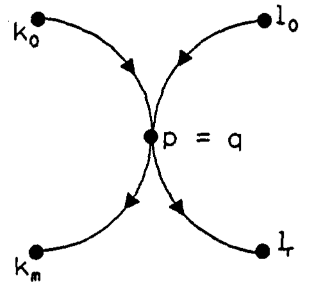

Fig. 4(a). The degenerate case: the active path has been reduced to a single node by drawing the K line up towards the L line.

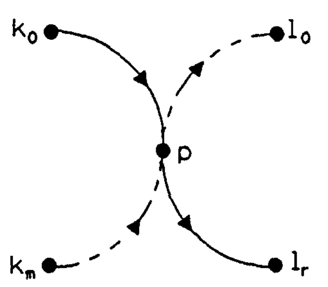

Fig. 4(b). Flow is drained from the K and L lines and the free capacity is used to push flow from A to $\bar{A}$.

problem known as the locking problem. Lomonosov shows how to solve this problem, classifying the types of families of sets which may be locked.

It can be shown that the problem of finding a multiflow of maximum magnitude in a network with terminal set T can be solved by simultaneously locking the family of proper subsets of T given by $\{(t): t \in T\}$. This family can be shown to be lockable, so the maximal multiflow problem may be solved.

Lomonosov also shows how to solve the EMF for special cases of $\{0, 1\}$-valued objective functions a. This type of problem is just the maximal multiflow problem where only flow of specified commodities is required. The solution involves locking a family of subsets of the terminal nodes and then manipulating the resulting solution.

4. CONCLUSIONS

4.1. *An example of problems solved*

The following example of a maximal multiflow problem on a network with 10 nodes and 6 terminal nodes was solved using a desktop computer (IBM-AT clone) in a matter of seconds. Figure 5 shows the network together with the capacities on the arcs. The maximal multiflow in this network is shown in Figs 6(a)–6(d). These figures depict the flows originating the nodes 1, 2, 3 and 4, respectively. To avoid clutter, only arcs along which the flow is non-zero are represented. Arcs are directed to correspond to the direction of the flow and the amount of flow is indicated against each arc. The net flow into or out of a terminal node is indicated by an incoming or outgoing arrow. Note that these figures actually represent the combined flow of several commodities. For example Fig. 6(b) shows the flows f_{23}, f_{24}, f_{25} and f_{26}.

4.2. *Performance*

The algorithm was tested on randomly generated networks with a range of values for the number of nodes, the number of commodities which is $(|\binom{t}{2}|)$, where t is the number of terminal nodes and the density of arcs. The performance is much as one would expect.

Figure 7 shows the times taken for the algorithm to run for the single commodity case on networks having a range of sizes, but all having about two thirds of the possible arcs. A quadratic function fitted the data well.

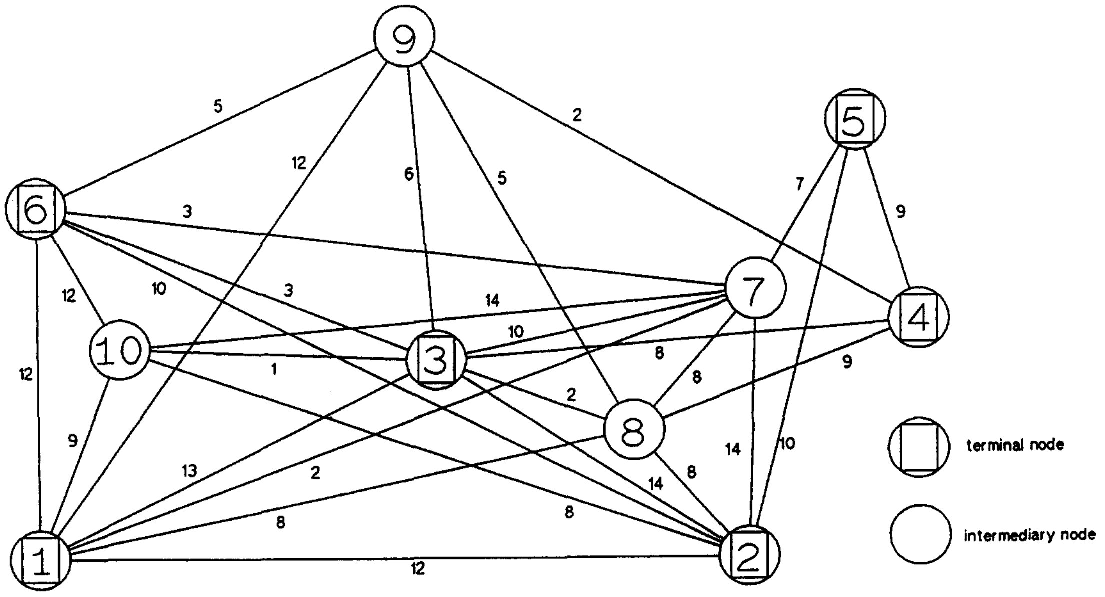

Fig. 5. Example of a network with 10 nodes, six of them terminal nodes. Capacities are indicated against each arc. Arcs not drawn have zero capacity. Capacities are integral and range from 0 to 14.

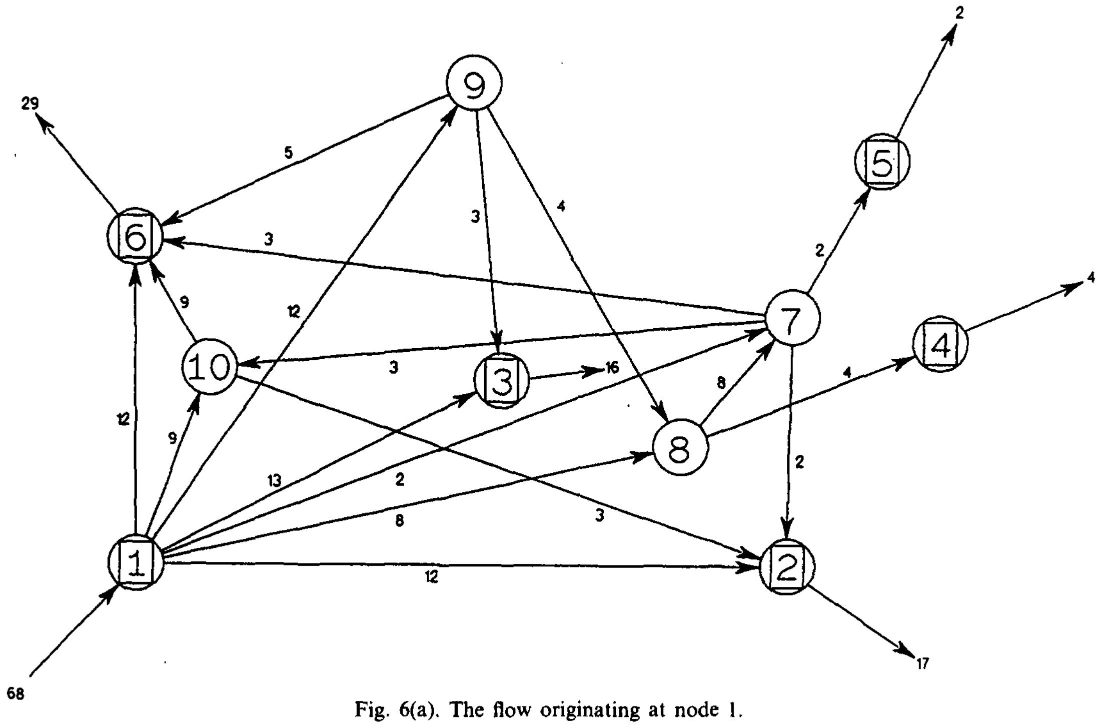

Fig. 6(a). The flow originating at node 1.

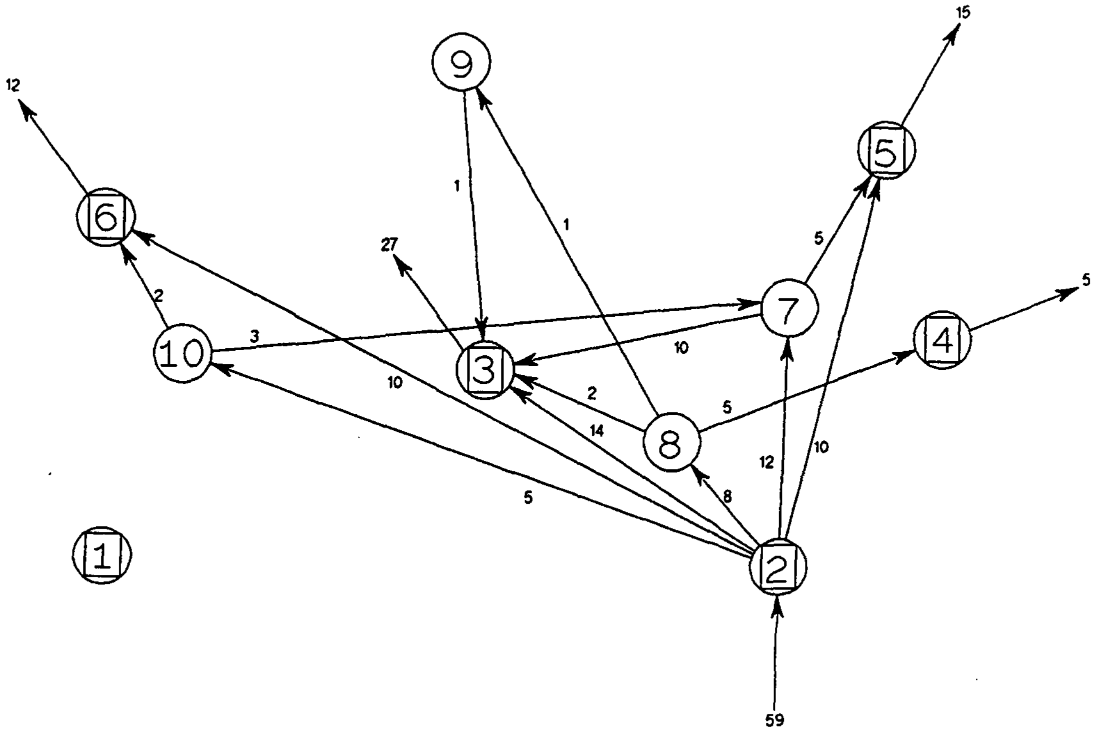

Fig. 6(b). The flow originating at node 2.

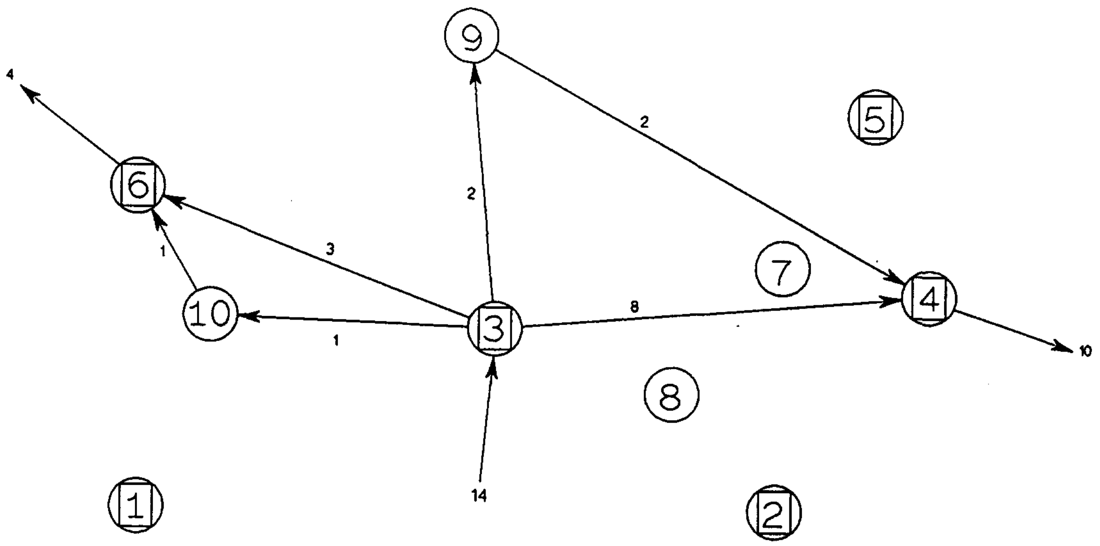

Fig. 6(c). The flow originating at node 3.

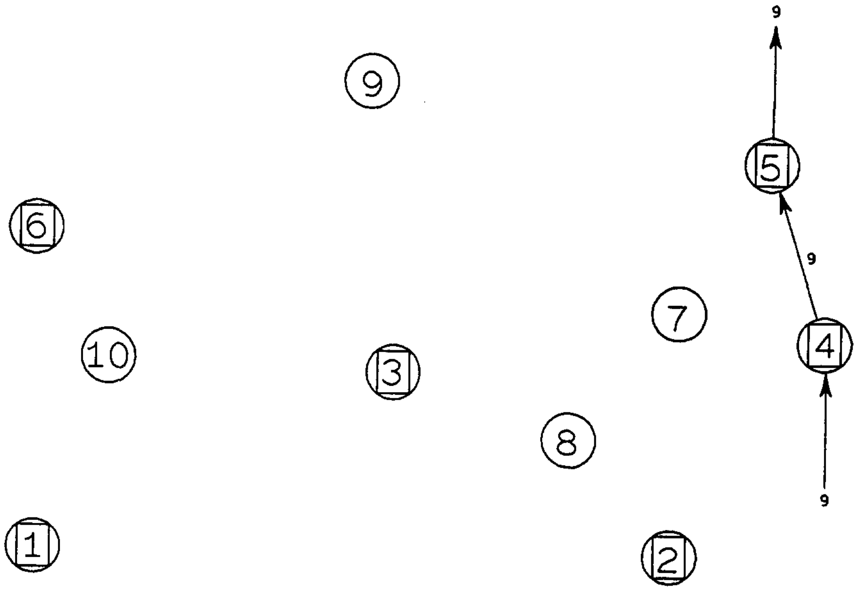

Fig. 6(d). The flow originating at node 4.

Figure 8 shows a typical case of the variation of time taken for the algorithm to run as the number of commodities (or terminal nodes) varies. The significant part of this graph is the lower part, since as the terminal nodes get very dense in the network, most arcs have both their nodes being terminal, and so may be immediately saturated with flow, thus speeding up the process of maximizing flow. Growth of time taken with number of terminal nodes is not readily fitted by a lower-order polynomial; we would expect it to be possibly exponential.

Figure 9 shows a typical case of the times taken when the number of commodities is fixed while the number of nodes varies. This relationship appears to be fitted well by a quadratic function.

To verify these assertions rigorously will require a lengthy complexity analysis of the algorithm.

4.3. *Future work*

The results so far are sufficiently encouraging to warrant pursuit of generalizations to a wider range of problems.

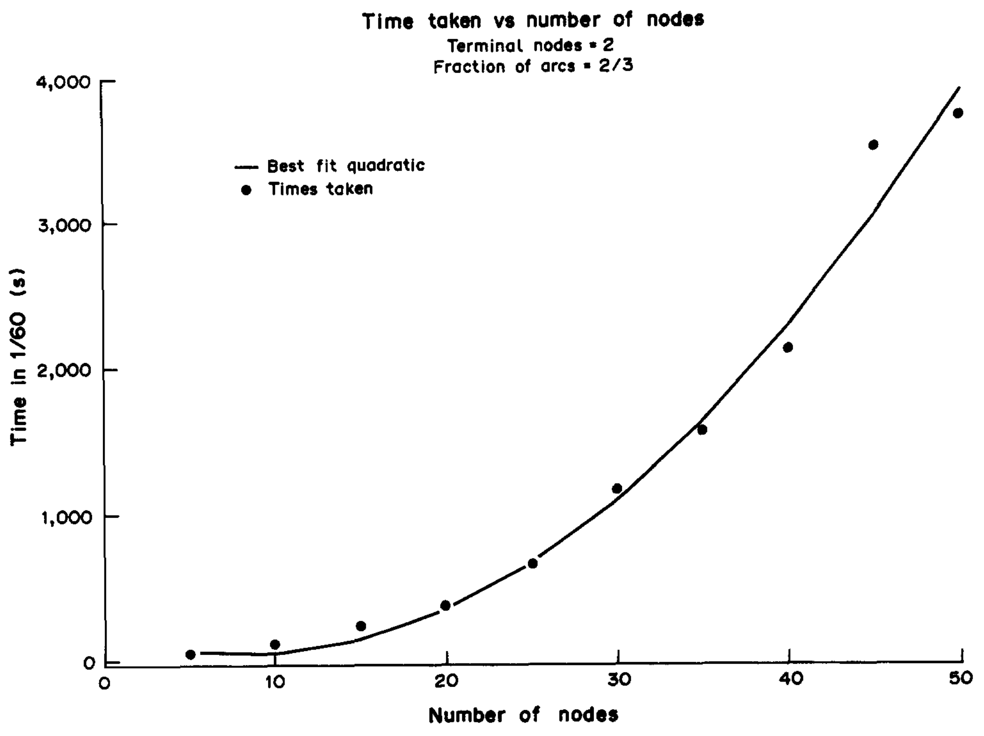

Fig. 7. The plotted points show the times taken from the algorithm to run on networks with varying numbers of nodes for the single commodity case. A quadratic function has been fitted to the data.

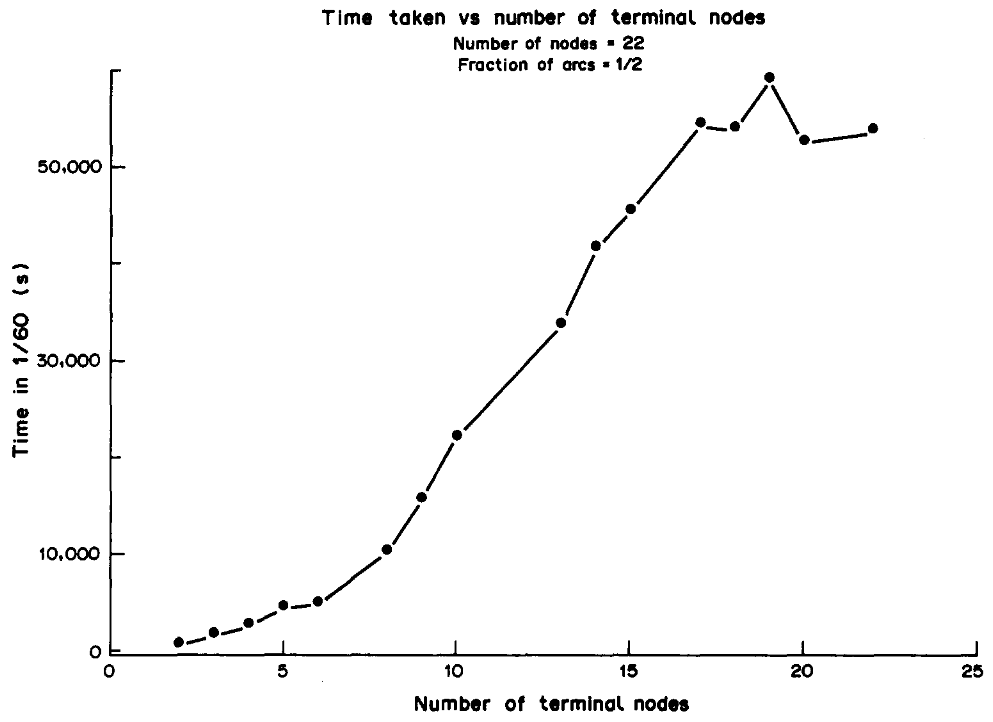

Fig. 8. The plotted points show the times taken for the algorithm to run on networks with a fixed number of nodes (22 nodes) but with an increasing number of terminal nodes (commodities).

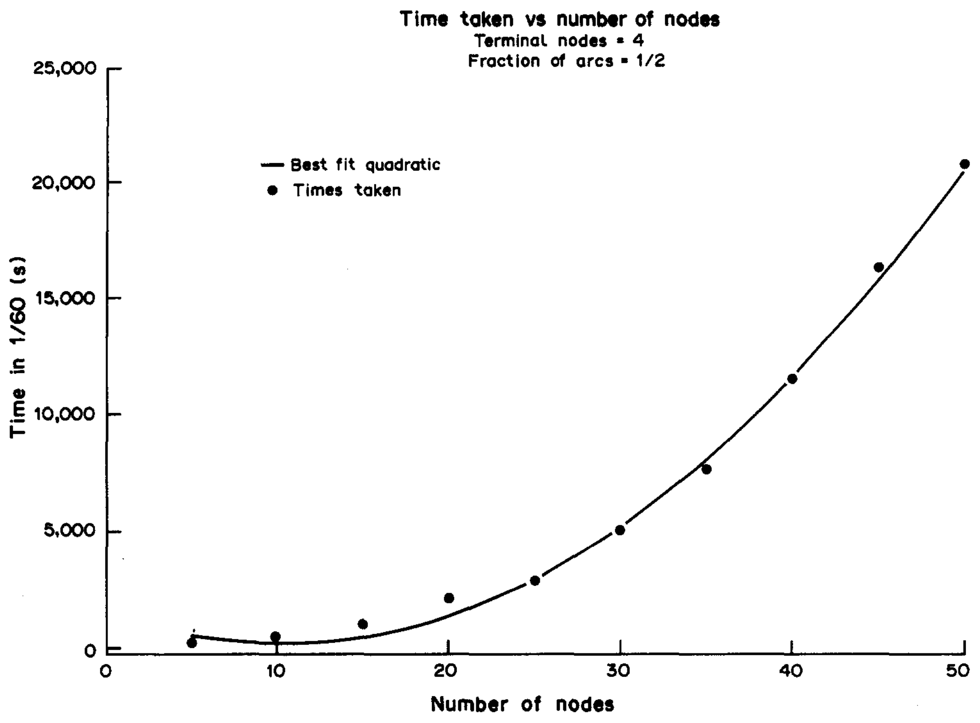

Fig. 9. The plotted points show the times taken for the algorithm to run on networks with varying numbers of nodes but all having 4 terminal nodes (or 6 commodities). A quadratic function has been fitted to the data.

In some multi-commodity flow applications the ability to solve the same sorts of problems for directed networks is important. However, the extension to the directed case is not, as one might suppose, a trivial one. Since the directed case is less important in most telecommunications applications, we shall leave discussion of this problem to another time.

Perhaps the greatest limitation of the algorithm presented is that it merely maximizes total traffic without giving performance to any node pairs. A partial solution is given by the $\{0, 1\}$-valued objective functions mentioned at the end of Section 3.2, in which only selected commodities are maximized. Such a problem can be solved by a relatively straightforward extension of the methods of Section 3. To determine a complete solution it is necessary to consider more general objective functions. Solutions to such general problems appear to require a more fully developed duality theory. This is a major thrust of our research.

Acknowledgement—This research was supported in part by an Australian Telecommunications and Electronic Research Board Scholarship.

REFERENCES

1. A. A. Assad, Multicommodity network flows—a survey. *Networks* **8**, 37–91 (1978).
2. M. Gondran and M. Minoux, *Graphs and Algorithms*. Wiley, New York (1984).
3. M. V. Lomonosov, Combinatorial approaches to multiflow problems. *Discr. appl. Math.* **11**(1), 1–93 (1985).
4. L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton Univ. Press, Princeton, N.J. (1962).