# Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems

JACK EDMONDS

*University of Waterloo, Waterloo, Ontario, Canada*

AND

RICHARD M. KARP

*University of California, Berkeley, California*

ABSTRACT. This paper presents new algorithms for the maximum flow problem, the Hitchcock transportation problem, and the general minimum-cost flow problem. Upper bounds on the numbers of steps in these algorithms are derived, and are shown to compare favorably with upper bounds on the numbers of steps required by earlier algorithms.

First, the paper states the maximum flow problem, gives the Ford-Fulkerson labeling method for its solution, and points out that an improper choice of flow augmenting paths can lead to severe computational difficulties. Then rules of choice that avoid these difficulties are given. We show that, if each flow augmentation is made along an augmenting path having a minimum number of arcs, then a maximum flow in an $n$-node network will be obtained after no more than $\frac{1}{4}(n^3 - n)$ augmentations; and then we show that if each flow change is chosen to produce a maximum increase in the flow value then, provided the capacities are integral, a maximum flow will be determined within at most $1 + \log_{M/(M-1)} f^*(t, s)$ augmentations, where $f^*(t, s)$ is the value of the maximum flow and $M$ is the maximum number of arcs across a cut.

Next a new algorithm is given for the minimum-cost flow problem, in which all shortest-path computations are performed on networks with all weights nonnegative. In particular, this algorithm solves the $n \times n$ assignment problem in $O(n^3)$ steps. Following that we explore a "scaling" technique for solving a minimum-cost flow problem by treating a sequence of derived problems with "scaled down" capacities. It is shown that, using this technique, the solution of a Hitchcock transportation problem with $m$ sources and $n$ sinks, $m \leq n$, and maximum flow $B$, requires at most $(n + 2) \log_2 (B/n)$ flow augmentations. Similar results are also given for the general minimum-cost flow problem.

An abstract stating the main results of the present paper was presented at the Calgary International Conference on Combinatorial Structures and Their Applications, June 1969. In a paper by Dinic (1970) a result closely related to the main result of Section 1.2 is obtained. Dinic shows that, in a network with $n$ nodes and $p$ arcs, a maximum flow can be computed in $O(n^2p)$ primitive operations by an algorithm which augments along shortest augmenting paths.

KEY WORDS AND PHRASES: network flows, transportation problem, analysis of algorithms

CR CATEGORIES: 5.3, 5.4, 8.3



Authors' addresses: J. Edmonds, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada; R. M. Karp, College of Engineering, Operations Research Center, University of California, Berkeley, CA 94720; the latter author's research has been partially supported by the National Science Foundation under Grant GP-15473 with the University of California.

## 1. *The Maximum Flow Problem*

1.1. THE LABELING METHOD. A *network* $N$ is a finite set $\{u, v, \cdots\}$ called *the nodes* and a subset of the ordered pairs $(u, v)$, $u \neq v$, called *the arcs*. Network $N$ has a special *return arc* $(t, s)$. Node $s$ is called the *source* in $N$ and node $t$ is called the *sink* in $N$. The set of all arcs of $N$, except $(t, s)$, we denote by $A$. For each $(u, v) \in A$ there is given a number $c(u, v) > 0$ called the *capacity* of arc $(u, v)$.

A nonnegative function $f(u, v)$, ranging over all arcs $(u, v)$ of $N$, is called a *flow* in $N$ if

(i) for every $(u, v) \in A$, $f(u, v) \leq c(u, v)$; and

(ii) for every node $u$,

$$\sum_v f(u, v) - \sum_v f(v, u) = 0,$$

where each sum is over every $v$ for which the summand is defined.

For each arc $(u, v)$ of $N$, $f(u, v)$ represents the amount of flow in arc $(u, v)$, and also represents the net amount of flow from $v$ to $u$ in the rest of the network "$N - (u, v)$."

The maximum network flow problem is to find a flow $f$ in $N$ such that $f(t, s)$, the net amount of flow in $N - (t, s)$ from $s$ to $t$, is maximum.

Let $u_1, u_2, \cdots, u_p$ be a sequence of distinct nodes such that, for each $i = 1, 2, \cdots, p - 1$, either $(u_i, u_{i+1})$ or $(u_{i+1}, u_i)$ is an arc. Singling out, for each $i$, one of these possibilities, we call the resulting sequence of arcs a *path* from $u_1$ to $u_p$. Arcs $(u_i, u_{i+1})$ that belong to the path are called *forward arcs* of the path; the other arcs of the path are called *reverse arcs*.

Relative to any given flow $f$ in $N$, a (flow) *augmenting path* is a path from $s$ to $t$ such that:

*Case (a):* If $(u_i, u_{i+1}) \in A$ and $(u_{i+1}, u_i) \notin A$, then

$$\epsilon_i = c(u_i, u_{i+1}) - f(u_i, u_{i+1}) > 0;$$

*Case (b):* If $(u_i, u_{i+1}) \in A$ and $(u_{i+1}, u_i) \in A$, then

$$\epsilon_i = f(u_{i+1}, u_i) > 0;$$

*Case (c):* If $(u_i, u_{i+1}) \in A$ and $(u_{i+1}, u_i) \in A$, then

$$\epsilon_i = c(u_i, u_{i+1}) - f(u_i, u_{i+1}) + f(u_{i+1}, u_i) > 0.$$

For a given augmenting path $P$, let $\epsilon = \min \epsilon_i > 0$. Call each arc $(u_i, u_{i+1})$ or $(u_{i+1}, u_i)$ in $P$ such that $\epsilon_i = \epsilon$ a *bottleneck arc* relative to $P$ and the flow $f$.

Now alter the flow $f$ as follows[1]: increase $f$ by $\epsilon$ on the return arc $(t, s)$; in Case (a), increase the flow on arc $(u_i, u_{i+1})$ by $\epsilon$; in Case (b), decrease the flow on arc $(u_{i+1}, u_i)$ by $\epsilon$; in Case (c), increase the flow on arc $(u_i, u_{i+1})$ by $\min(\epsilon, c(u_i, u_{i+1}) - f(u_i, u_{i+1}))$ and decrease the flow on arc $(u_{i+1}, u_i)$ by $\max(0, \epsilon - c(u_i, u_{i+1}) + f(u_i, u_{i+1}))$. It is easily checked that the $f'$ thus defined is a flow in $N$. Thus, since $f'(t, s) = f(t, s) + \epsilon$, the flow $f$ is not maximum. It can

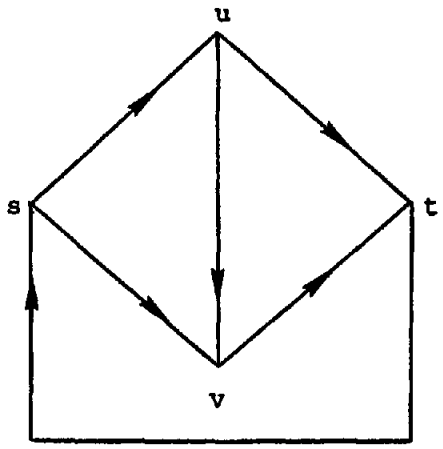
[1] The method of augmentation presented here differs [in Case (c)] from the method originally given by Ford and Fulkerson (cf. [5]). The results of this paper apply, with minor changes, to the Ford-Fulkerson method as well.

be shown that, conversely, a flow $f$ in $N$ is not maximum only if there is an augmenting path with respect to $f$.

The *labeling method* constructs a sequence $F = f^0, f^1, f^2, \cdots$ of flows in $N$, starting with, say, the zero flow, by finding an augmenting path with respect to $f^k$ if one exists, and then augmenting to obtain $f^{k+1}$. The sequence terminates only when a maximum flow has been obtained.

Assuming that all the capacities $c(u, v)$ are integers, then clearly for any augmenting path $P$ relative to any integer-valued flow $f$, $\epsilon$ is a positive integer. Thus, since $f^0$ is integer-valued, all the later flows $f^k$ in the sequence $F$ are integer-valued. It follows that the sequence terminates after a number of labelings not greater than the final value of $f(t, s)$.

The following example illustrates that this upper bound on the number of labelings can actually occur.




Suppose that the arc $(u, v)$ has capacity 1, and the capacity of each of the other arcs in $A$ is $M$, a positive integer. Then the maximum value of $f(t, s)$ is $2M$, and $2M$ labelings will be required if the labeling process alternates between selecting $(s, u)(u, v)(v, t)$ and $(s, v)(v, u)(u, t)$ as an augmenting path. For, in each case, either $(u, v)$ or $(v, u)$ is a bottleneck arc, and $\epsilon = 1$.

Assuming that all the capacities are mutually commensurable, we can obtain an equivalent integer-valued problem by multiplying all the capacities by a large constant. Thus, in this case also, the sequence $F$ is finite.

Ford and Fulkerson show by an example that if the capacities are not commensurable then the sequence $F$ need not terminate, and in fact, may converge to a nonmaximum flow.

Since numerical computation is always, in practice, performed on numbers expressed to a finite precision, this nonfiniteness is not from a practical viewpoint a very serious matter. It does serve as another indication of the tendency of the number of augmentations to grow as the precision to which the capacities are expressed increases.

We will show that these theoretical difficulties, which could conceivably be a practically serious matter, can be avoided. In particular, by making a certain refinement of the labeling method which is so simple that it is likely to be incorporated

innocently into a computer implementation, we get a bound of at most $\frac{1}{4}(n^3 - n)$ terms in the sequence $F$ (regardless of commensurability), where $n$ is the number of nodes. In addition, a second refinement of the labeling method is shown to yield a bound on the length of $F$, applicable only in the case of integer capacities, of $1 + \log_{M/(M-1)} f^*(t, s)$, where $f^*(t, s)$ is the value of a maximum flow, and $M \leq n^2/2$.

1.2. A REFINEMENT. The labeling method requires as a subroutine a *labeling process* for finding, if one exists, an augmenting path $P$ relative to a given flow $f$ in $N$. This is essentially a method for finding, in a certain network $N^f$ having the same nodes as $N$, a directed path from $s$ to $t$. A *directed path* from $s$ to $t$ is a path such that all arcs are forward arcs. The ordered pair $(u, v)$ is an arc of $N^f$ if and only if either

$$(u, v) \in A \quad \text{and} \quad c(u, v) - f(u, v) > 0$$

or

$$(v, u) \in A \quad \text{and} \quad f(v, u) > 0.$$

The arcs of any directed path $P^f$ from $s$ to $t$ in $N^f$ are in one-one correspondence with the arcs of an augmenting path $P$ in $N$ relative to $f$. The arc of $P^f$ corresponding to a bottleneck arc of $P$ is also referred to as a bottleneck arc.

The labeling process for finding a directed path in $N^f$ from $s$ to $t$ is as follows: First $s$ gets "labeled." Then at each successive step of the process some labeled but "unscanned" node gets scanned. To *scan* a labeled node $u$ means to label every node $v$ not already labeled and such that the arc $(u, v)$ is in $N^f$. If $v$ gets labeled when $u$ is scanned, then $u$ is the *predecessor* of $v$ in the labeling.

As soon as the sink $t$ gets labeled, then $t$, the predecessor of $t$, the predecessor of that predecessor, and so on back to $s$, is the reverse sequence of a directed path in $N^f$ from $s$ to $t$. On the other hand, if every labeled node gets scanned without $t$ getting labeled, then there is evidently no directed path in $N^f$ from $s$ to $t$. Clearly the labeling process terminates in one or the other of these two situations.

The refinement treated here, which gives an upper bound of $\frac{1}{4}(n^3 - n)$ on the number of applications of the labeling process before obtaining a maximum flow, is the following: In the labeling process, scan on a "first-labeled first-scanned" basis. That is, before scanning a labeled node $u$, scan the nodes that got labeled before $u$.

It can be shown that a directed path in $N^f$ from $s$ to $t$, obtained by this version of the labeling process, is one which contains a minimum number of arcs. Thus, the upper bound can be stated as follows:

THEOREM 1. *If, in the labeling method for finding a maximum flow in a network on $n$ nodes, each flow augmentation is done along an augmenting path having fewest arcs, then a maximum flow will be obtained after no more than $\frac{1}{4}(n^3 - n)$ augmentations.*

For present purposes, we will regard the number of arcs in a path as its length. The "distance" from a node $u$ to a node $v$ in $N^f$ is the minimum length of a directed path from $u$ to $v$ in $N^f$, or else $\infty$ if there is no such path.

Let $F = f^0, f^1, f^2, \cdots$ be any sequence of flows in $N$ such that $f^{k+1}$ is obtained from $f^k$ by an augmentation corresponding to a shortest directed path $P^k$ in $N^{f^k}$. Let $N^k$ denote $N^{f^k}$, and let $\delta^k(u, v)$ denote the distance from $u$ to $v$ in $N^k$.

LEMMA 1. *If $k < m$ and $(u, v)$ is a bottleneck arc relative to $P^k$ and $f^k$, and also relative to $P^m$ and $f^m$, then, for some $l$ such that $k < l < m$, $(v, u) \in P^l$.*

LEMMA 2. *If $k < l$, $(u, v) \in P^k$ and $(v, u) \in P^l$, then $\delta^l(s, t) \geq \delta^k(s, t) + 2$.*

Given these lemmas, the proof of Theorem 1 is at hand. Let $\{u, v\}$ be any pair of nodes such that $(u, v) \in A$ or $(v, u) \in A$. Let the sequence $\{k_i\}$ consist of all indices $k_i$ such that either $(u, v)$ or $(v, u)$ is a bottleneck arc relative to $P^{k_i}$ and $f^{k_i}$. By Lemma 1, one can find a sequence $\{l_j\}$, containing $\{k_i\}$ as a subsequence, such that

$$(u, v) \in P^{l_j}, \quad j \text{ odd} \quad \text{and } (v, u) \in P^{l_j}, \quad j \text{ even}$$

or

$$(u, v) \in P^{l_j}, \quad j \text{ even} \quad \text{and } (v, u) \in P^{l_j}, \quad j \text{ odd}.$$

By Lemma 2, $\delta^{l_{j+1}}(s, t) \geq \delta^{l_j}(s, t) + 2$, $j = 1, 2, \cdots$. Thus, $\delta^{l_j}(s, t) \geq 2(j - 1)$. But the length of any directed path in $N^k$ is at most $n - 1$ so that $\delta^{l_j}(s, t) \leq n - 1$ for all $j$. The length of the sequence $\{l_j\}$ is therefore at most $\frac{1}{2}(n - 1) + 1 = \frac{1}{2}(n + 1)$, and thus the number of occurrences of $(u, v)$ or $(v, u)$ as a bottleneck arc throughout the entire labeling method is at most $\frac{1}{2}(n + 1)$. The number of occurrences of bottleneck arcs altogether is therefore bounded by

$$\frac{n + 1}{2}\binom{n}{2} = \frac{n^3 - n}{4}.$$

And, since every augmentation determines a bottleneck arc, the number of augmentations is also bounded by $\frac{1}{4}(n^3 - n)$.

The proof of Lemma 1 employs two simple propositions.

PROPOSITION 1. *If $(u, v)$ is a bottleneck arc relative to $P^k$ and $f^k$, then $(u, v) \notin N^{k+1}$.*

PROOF. The augmentation from $f^k$ to $f^{k+1}$ is such that, if $(u, v) \in A$ then $f^{k+1}(u, v) = c(u, v)$, and if $(v, u) \in A$ then $f^{k+1}(v, u) = 0$; hence, $(u, v) \notin N^{k+1}$. ‖

PROPOSITION 2. *If $(u, v) \in N^{k+1}$ then $(u, v) \in N^k$ or $(v, u) \in P^k$.*

PROOF. Suppose $(u, v) \in N^{k+1}$ and $(u, v) \in N^k$; then, either $f^{k+1}(u, v) \neq f^k(u, v)$ or $f^{k+1}(v, u) \neq f^k(v, u)$. In either case, $(u, v)$ or $(v, u)$ must be in $P^k$. But $(u, v) \notin P^k$, since $(u, v) \notin N^k$; thus, $(v, u) \in P^k$.

We can now prove Lemma 1. By Proposition 1, $(u, v) \notin N^{k+1}$; since $(u, v) \in P^m$, $(u, v) \in N^m$. Let $l + 1 = \min \{t \mid t > k \text{ and } (u, v) \in N^t\}$. Then $(u, v) \in N^{l+1}$, $(u, v) \notin N^l$; hence, by Proposition 2, $(v, u) \in P^l$. This completes the proof of Lemma 1. ‖

The proof of Lemma 2 will make use of the following proposition.

PROPOSITION 3. *For $k = 0, 1, 2, \cdots$, and for all $u$,*

$$\delta^k(s, u) \leq \delta^{k+1}(s, u) \tag{1}$$

and

$$\delta^k(u, t) \leq \delta^{k+1}(u, t). \tag{2}$$

PROOF. We prove (1), the proof of (2) being similar. If $\delta^{k+1}(s, u) = \infty$, the result is evident. Assuming $\delta^{k+1}(s, u) = h$ is finite, let $s = u_0, u_1, \cdots, u_h = u$ be the node sequence of a shortest directed path from $s$ to $u$. Then $\delta^k(s, u_0) = 0$ and we claim that

$$\delta^k(s, u_{i+1}) \leq 1 + \delta^k(s, u_i), \qquad i = 0, \cdots, h - 1. \tag{3}$$

For, since $(u_i, u_{i+1}) \in N^{k+1}$, Proposition 2 tells us that $(u_i, u_{i+1}) \in N^k$ or $(u_{i+1}, u_i) \subset P^k$. In the former case, $\delta^k(s, u_{i+1}) \leq 1 + \delta^k(s, u_i)$, since the arc $(u_i, u_{i+1})$ enables us to get a directed path from $s$ to $u_{i+1}$ in $N^k$ having no more than $1 + \delta^k(s, u_i)$ arcs. In the latter case, $\delta^k(s, u_i) = 1 + \delta^k(s, u_{i+1})$, so $\delta^k(s, u_{i+1}) =$

$-1 + \delta^k(s, u_i) < 1 + \delta^k(s, u_i)$. Summing the inequalities given in (3),

$$\delta^k(s, u) \leq h + \delta^k(s, u_0) = h = \delta^{k+1}(s, u),$$

and (1) is proved. ‖

To prove Lemma 2 note that, since $(u, v) \in P^k$, $\delta^k(s, t) = \delta^k(s, u) + 1 + \delta^k(v, t)$. Also, $\delta^k(s, v) = 1 + \delta^k(s, u)$ and $\delta^k(u, t) = 1 + \delta^k(v, t)$. Since $(v, u) \in P^l$, $\delta^l(s, t) = \delta^l(s, v) + 1 + \delta^l(u, t)$. But, by Proposition 3, $\delta^l(s, v) \geq \delta^k(s, v)$ and $\delta^l(u, t) \geq \delta^k(u, t)$, so that $\delta^l(s, t) \geq \delta^k(s, v) + 1 + \delta^k(u, t) = (1 + \delta^k(s, u)) + 1 + (1 + \delta^k(v, t)) = 2 + \delta^k(s, t)$. Thus, Lemma 2 is proved, and we are done. ‖

The proof of Theorem 1 can be modified quite simply to supply bounds on the numbers of augmentations required in certain other refinements of the Ford-Fulkerson labeling method. Let $a(u, v)$ be a real-valued function defined whenever $(u, v) \in A$ or $(v, u) \in A$, such that $b(u, v) = a(u, v) + a(v, u) > 0$. Let the weight of a path $P$ in $N$ having the node sequence $u_1, u_2, \cdots, u_p$ be $\sum_{i=1}^{p-1} a(u_i, u_{i+1})$. Consider a variant of the labeling method in which each augmentation is along a flow-augmenting path of minimum weight. Then the number of augmentations cannot exceed $(S \sum_{(u,v)\in A} 1/|b(u, v)|) + |A|$, where $S$ is the maximum weight of a path from source to sink. Theorem 1 corresponds to the case where $a(u, v) = 1$ for all pairs $(u, v)$. Another case, corresponding to the rule: "select a flow-augmenting path with as few reverse arcs as possible," has $a(u, v) = 1$ if $(v, u) \in A$, and $a(u, v) = 0$ otherwise. A bound on the number of iterations in this case is $\frac{1}{2}(n^3 - n^2)$.

1.3. A Second Refinement. In this section we consider the following refinement of the labeling method: at each iteration choose a flow-augmenting path which gives the largest possible augmentation.

Let $N$ be a network in which every capacity is an integer. Let $M > 1$ be a positive integer such that, for any partition of the nodes of $N$ into two sets, $X$ and $\bar{X}$, with $s \in X$ and $t \in \bar{X}$, the number of arcs with one end in $X$ and the other in $\bar{X}$ is less than or equal to $M$. Let $f^*(t, s)$ denote the value of a maximum flow.

Theorem 2. *If, in the labeling method for finding a maximun flow in $N$, a network with all capacities integral, each augmentation is done along an augmenting path giving the maximum possible augmentation, then a maximum flow will be obtained after no more than $1 + \log_{M/(M-1)} f^*(t, s)$ augmentations.*

Before proving Theorem 2, we show how the rule under consideration can be implemented. Suppose we are seeking a flow-augmenting path in $N$ relative to a flow $f$. Associate with each arc $(u, v) \in N^f$ a number $e(u, v)$ equal to the value of $\epsilon$ that would result if $(u, v)$ were a bottleneck arc in a flow-augmenting path relative to $N$ and $f$. Specifically,

(i) if $(u, v) \in A$ and $(v, u) \notin A$, then $e(u, v) = c(u, v) - f(u, v)$,
(ii) if $(u, v) \notin A$ and $(v, u) \in A$, then $e(u, v) = f(u, v)$,
(iii) if $(u, v) \in A$ and $(v, u) \in A$, then $e(u, v) = c(u, v) - f(u, v) + f(v, u)$.

Then the labeling method seeks a directed path from $s$ to $t$ in $N^f$ in which the mallest value of $e(u, v)$ is as large as possible. This is a bottleneck problem of the ype studied in [4]. One method of finding such a path is to label $s$, and then to repeat he following step until $t$ is labeled: find an arc $(u', v') \in N^f$ such that $u'$ is labeled, $'$ is not labeled, and for any arc $(u, v)$ from a labeled node to an unlabeled node, $(u', v') \geq e(u, v)$. Label $v'$ and record $u'$ as the predecessor of $v'$. When $t$ is labeled,

tracing the sequence of predecessors back from $t$ gives a maximum-$\epsilon$ flow-augmenting path. If, at some step, there is no arc from a labeled node to an unlabeled one, then no flow-augmenting path exists.

PROOF OF THEOREM 2. Consider a partition of the nodes of $N$ into two sets, $X$ and $\bar{X}$, such that $s \in X$ and $t \in \bar{X}$. Define

$$c(X, \bar{X}) = \sum_{\substack{u \in X \\ v \in \bar{X} \\ (u,v) \in A}} c(u, v), \quad f(X, \bar{X}) = \sum_{\substack{u \in X \\ v \in \bar{X} \\ (u,v) \in A}} f(u, v)$$

and

$$f(\bar{X}, X) = \sum_{\substack{u \in \bar{X} \\ v \in X \\ (u,v) \in A}} f(u, v).$$

Then, for any flow $f$,

$$c(X, \bar{X}) \geq f(X, \bar{X}) - f(\bar{X}, X) = f(t, s).$$

Suppose the labeling method using maximum augmentations produces the sequence of flows $f^0, f^1, \cdots, f^k, \cdots$. Let $\epsilon^k = f^{k+1}(t, s) - f^k(t, s)$. Consider the augmentation from $f^k$ to $f^{k+1}$. Let the set $X$ consist of $s$ together with all nodes which can be reached from $s$ by a directed path in $N^k$ consisting of arcs $(u, v)$ such that $e(u, v) > \epsilon^k$; let $\bar{X}$ denote the remaining nodes. Then $t \in \bar{X}$ and every arc $(u, v)$ in $N^k$ such that $u \in X$ and $v \in \bar{X}$ satisfies $e(u, v) \leq \epsilon^k$.

$$c(X, \bar{X}) - [f^k(X, \bar{X}) - f^k(\bar{X}, X)] \leq \epsilon^k \left| \{ (u, v) \mid u \in X, v \in \bar{X}, (u, v) \in A \text{ or } (v, u) \in A \} \right| \leq \epsilon^k M.$$

Now $f^*(t, s) \leq c(X, \bar{X})$ and $f^k(t, s) = f^k(X, \bar{X}) - f^k(\bar{X}, X)$, so

$$f^*(t, s) - f^k(t, s) \leq \epsilon^k M; \quad \text{i.e.} \quad f^*(t, s) - f^k(t, s) < [f^{k+1}(t, s) - f^k(t, s)]M.$$

Equivalently,

$$f^*(t, s) - f^{k+1}(t, s) \leq [f^*(t, s) - f^k(t, s)](1 - M^{-1}).$$

Thus, by induction,

$$f^*(t, s) - f^k(t, s) \leq f^*(t, s)(1 - M^{-1})^k.$$

Now, since all the capacities are integers, each flow is integral. Thus, if $f^k$ is not a maximum flow, then

$$f^*(t, s) - f^k(t, s) \geq 1,$$

so

$$f^*(t, s)(1 - M^{-1})^k \geq 1$$

and

$$k \leq -\log_{1-1/M} f^*(t, s) = \log_{M/(M-1)} f^*(t, s),$$

so the total number of augmentations cannot exceed

$$1 + \log_{M/(M-1)} f^*(t, s).$$

Let $\bar{c}$ denote the average capacity of an arc in $A$. Then $f^*(t, s) \leq \bar{c}n^2$ and $M \leq \frac{1}{2}n^2$, so

$$\log_{M/(M-1)} f^*(t, s) \leq \log_{1+2/(n^2-2)}(n^2\bar{c}) = \frac{\ln n^2\bar{c}}{\ln (1 + 2/(n^2 - 2))}.$$

But

$$\ln\left(1 + \frac{2}{n^2 - 2}\right) \geq \ln\left(1 + \frac{2}{n^2}\right) \geq \frac{2}{n^2} - \frac{1}{2}\left(\frac{2}{n^2}\right)^2.$$

Using these estimates we find that the number of augmentations cannot exceed

$$1 + \frac{2 \ln n + \ln \bar{c}}{2(1/n^2 - 1/n^4)} = 1 + \frac{n^4}{2n^2 - 2}(2 \ln n + \ln \bar{c})$$
$$= n^2 \ln n + \tfrac{1}{2} n^2 \ln \bar{c} + O(n^2 \ln n + n^2 \ln \bar{c}).$$

Thus, although the present bound depends on the capacities and requires their integrality, it is superior to the bound of Section 1.2 in approximately the range $0 \leq \bar{c} \leq e^{n/4}$.

## 2. *The Minimum-Cost Flow Problem*

2.1. A LABELING METHOD. In this section we turn to the problem of finding a maximum flow of minimum cost. Given a network $N$, associate with each arc $(u, v) \in A$ a nonnegative *cost* $d(u, v)$ as well as the usual positive capacity $c(u, v)$. Let the *cost* of a flow $f$ be $\sum_{(u,v) \in A} d(u, v) f(u, v)$ and let its *value* be $f(t, s)$. We seek a flow of minimum cost among those with value $f^*(t, s)$.

Call a flow $f$ *extreme* if it is of minimum cost among flows with value $f(t, s)$. We mention some well-known characterizations of extreme flows. In doing so, we make use of the network $N^f$ associated with $f$. We recall that a network, by definition, has at most one arc from one given node to another. For convenience we also assume that $(u, v) \in A \Rightarrow (v, u) \notin A$. Obvious devices using "fictitious nodes" can be used to enforce this restriction if it does not originally hold. Associate with any arc $(u, v)$ of $N^f$ a *weight* $\Delta(u, v)$ as follows:

$$\Delta(u, v) = \begin{cases} d(u, v), & (u, v) \in A, \\ -d(v, u), & (v, u) \in A. \end{cases}$$

Define the weight of a subgraph of $N^f$ as the sum of the weights of its arcs. Define a *labeling function* as a function from the nodes to the real numbers.

THEOREM 3.[2] *Let $f$ be a flow. Then the following are equivalent:*

(i) $f$ is extreme,

(ii) every directed cycle in $N^f$ has nonnegative weight,

(iii) there exists a labeling function $\pi$ such that, for every arc $(u, v)$ of $N^f$, $\pi(u) + \Delta(u, v) - \pi(v) \geq 0$.

A restatement of (iii) in terms of the network $N$ is: for $(u, v) \in A$,

$$\begin{aligned} &\pi(u) - \pi(v) + d(u, v) > 0 \Rightarrow f(u, v) = 0, \\ &\pi(u) - \pi(v) + d(u, v) < 0 \Rightarrow f(u, v) = c(u, v). \end{aligned} \tag{4}$$

If the flow $f$ and the labeling function $\pi$ together satisfy (4), then $f$ and $\pi$ are called *compatible*.

Another basic result is the following.

[2] The equivalence of (i) and (iii), stated in a somewhat different form, can be found in [5, pp. 114–115].

THEOREM 4. ([5, p. 121]). *If $f$ is extreme and $P$ is a path of minimum weight in $N^f$ from $s$ to $t$, then a flow $f'$ obtained by augmenting along $P$ is extreme.*

For brevity call a path of minimum weight a "shortest path." Theorem 4 suggests the following method of solving the minimum-cost flow problem: starting with an extreme flow $f^0$, compute a sequence of extreme flows $f^0, f^1, \cdots, f^k, f^{k+1}, \cdots$, obtaining $f^{k+1}$ from $f^k$ by augmenting along a shortest path from $s$ to $t$ in $N^{f^k}$. A shortest path can be determined using the following algorithm.

ALGORITHM A: Shortest-Path Algorithm

Let $N^{f^k}$ have the set of arcs $A^k$, and let $\Delta(u, v)$ be the weight of arc $(u, v) \in A^k$.

(1) Set $\sigma(s) = 0$ and set $\sigma(u) = +\infty$, $u \neq s$.

(2) Set $S = \{s\}$.

(3) If $S = \phi$, halt; otherwise choose $u^*$ such that $u^* \in S$ and $\sigma(u^*) = \min_{u \in S} \sigma(u)$.

(4) For each $v$ such that $(u^*, v) \in A^k$, set

$$\sigma(v) = \min (\sigma(v), \sigma(u^*) + \Delta(u^*, v)).$$

If this process decreases $\sigma(v)$, adjoin $v$ to the set $S$.

(5) Delete $u^*$ from $S$ and go to (3).

Algorithm A has the following properties:

(1) Upon its termination, $\sigma(u)$ gives the weight of a shortest path from $s$ to $u$;

(2) If $\Delta(u, v) \geq 0$ for every arc $(u, v)$, then each vertex accessible from $s$ enters the set $S$ exactly once, so that the total amount of computation is proportional to the number of arcs;

(3) If no cycle is of negative weight, then each vertex accessible from $s$ enters the set $S$ at most $n - 1$ times, so that the total amount of computation has a bound proportional to $n - 1$ times the number of arcs;

(4) If there is a negative-weight cycle accessible from $s$, then the algorithm is nonterminating. One way to detect this is to keep a subgraph $T$ of tentative shortest paths. $T$ contains arc $(u, v)$ if $v$ last entered $S$ during an application of Step (4) with $u^* = u$. Any cycle in $T$ has negative weight; if a negative-weight cycle is accessible from $s$, then such a cycle will occur in $T$ by the time any vertex enters $S$ for the $n$th time.

The discussion of the shortest-path algorithm shows the efficiency to be gained in cases when all weights are nonnegative. Too little attention has been paid to this essential point in the development of algorithms for minimum-cost flows. We present in this section an algorithm designed so that all shortest-path calculations are done on networks with all weights nonnegative.

Let $f$ be a flow and let $\pi$ be a labeling function. Assign each arc $(u, v)$ of $N^f$ a weight $\bar{\Delta}(u, v) = \pi(u) + \Delta(u, v) - \pi(v)$.

Then clearly

(i) if $C$ is a directed cycle, then

$$\sum_{(u,v) \in C} \bar{\Delta}(u, v) = \sum_{(u,v) \in C} \Delta(u, v);$$

(ii) if $P$ is a directed path from $u^*$ to $v^*$, then

$$\sum_{(u,v) \in P} \bar{\Delta}(u, v) = \pi(u^*) - \pi(v^*) + \sum_{(u,v) \in P} \Delta(u, v).$$

Thus, $N^f$ has a cycle of negative weight with respect to the weights $\Delta(u, v)$ if

and only if $N'$ has a cycle of negative weight with respect to the weights $\bar{\Delta}(u, v)$. Also, $P$ is a shortest path from $s$ to $t$ with respect to the weights $\Delta(u, v)$ if and only if $P$ is a shortest path from $s$ to $t$ with respect to the weights $\bar{\Delta}(u, v)$. Consider the implications of these facts when $f$ and $\pi$ are compatible. Then $\bar{\Delta}(u, v) \geq 0$, and a minimum-weight flow-augmenting path (relative to the weights $\Delta(u, v)$) can be found by a shortest-path calculation using the nonnegative weights $\bar{\Delta}(u, v)$.

A variant of the algorithm suggested by Theorem 4 is now apparent which performs all shortest-path calculations on networks with all weights nonnegative.

ALGORITHM B: Minimum-Cost Flow Algorithm

(1) Set $f^0$ equal to the zero flow, and set $\pi^0$ equal to the identically zero labeling function;
(2) Given $f^k$ and $\pi^k$, determine $f^{k+1}$ by augmenting along a minimum-weight path from $s$ to $t$ in $N^{f^k}$ with respect to the (nonnegative) weights

$$\Delta^k(u, v) = \pi^k(u) + \Delta(u, v) - \pi^k(v).$$

If several minimum-weight paths exist, choose one with the fewest arcs.
(3) If $\sigma^k(u)$ denotes the weight of a shortest path from $s$ to $u$ with respect to the weights $\Delta^k$, set $\pi^{k+1}(u) = \pi^k(u) + \sigma^k(u)$; take $\sigma^k(u) = \pi^{k+1}(u) = +\infty$ if $u$ is inaccessible from $s$ in $N^{f^k}$.
(4) Halt when, for some $k$, no flow-augmenting path exists with respect to $f^k$.

Some properties of the algorithm are given in the following theorem.

THEOREM 5. *For each $k$, $f^k$ and $\pi^k$ are compatible. For each $k$ and $u$, $\pi^k(u)$ gives the weight of a shortest path from $s$ to $u$ in $N^{f^k}$ with respect to the weights $\Delta(u, v)$ and $\pi^{k+1}(u) \geq \pi^k(u)$.*

We present two bounds on the number of flow augmentations required by the minimum-cost flow algorithm.

THEOREM 6. *If all the capacities are integers, then the computation terminates after at most $f^*(t, s)$ flow augmentations.*

PROOF. Each flow $f^k$ is integral, and each augmentation increases the flow by a positive integer. ||

THEOREM 7. *Suppose the costs $d(u, v)$ are integers less than or equal to an integer $D$. Then the computation terminates after at most $1 + \frac{1}{4}(n^3 - n)(n - 1)D$ flow augmentations.*

PROOF. We show that the overall computation can be regarded as a sequence of at most $(n - 1)D + 1$ phases, each consisting of a maximum flow computation. Each phase corresponds to a period during which $\pi^k(t)$ remains constant. Suppose $\pi^k(t)$ is constant for $k_1 \leq k \leq k_2$. Then the flow augmentations involved in passing from $f^{k_1}$ to $f^{k_2}$ are along directed paths in the subnetwork $N'$ containing those arcs $(u, v)$ in $N^{f^{k_1}}$ such that $\pi^{k_1}(u) + \Delta(u, v) - \pi^{k_1}(v) = 0$. Hence, these augmentations are part of a maximum-flow computation in $N'$. The bound of Section 1.2 is applicable, since the algorithm selects, at each step, a path with fewest arcs among those of minimum weight. Hence, the number of augmentations per phase is at most $\frac{1}{4}(n^3 - n)$. Now, except at the last step, when $t$ is inaccessible by a flow-augmenting path and $\pi^k(t) = \infty$, $\pi^k(t)$ is the weight of some path in $N$ from $s$ to $t$, and hence is an integer between 0 and $(n - 1)D$. Thus, noting that $\pi^k(t)$ is nondecreasing with $k$, we see that the number of phases, excluding the last step, is at most $(n - 1)D$, and the theorem follows. ||

COROLLARY 1. *Algorithm B solves any minimum-cost flow problem in a finite number of steps (even when neither the capacities nor the costs are commensurable).*

PROOF. The first half of the proof of Theorem 7 is applicable in this case, and shows that there is a finite bound on the number of successive flow augmentations without an increase in $\pi^k(t)$. But, for any $k$, $\pi^k(t)$ is the weight of some directed path from $s$ to $t$ in $N'^k$, corresponding to some path without repeated vertices in $N$. Since the number of such paths is finite $\pi^k(t)$ increases only a finite number of times, so that the entire process must be finite. ||

Although it is comforting to know that the minimum-cost flow algorithm terminates, the bounds on the number of augmentations are most unfavorable. The scaling method of the next two sections is a variant of this algorithm in which the bound depends logarithmically, rather than linearly, on the capacities. A challenging open problem is to emulate the results of Section 1.2 for the maximum-flow problem by giving a method for the minimum-cost flow problem having a bound on computation which is a polynomial in the number of nodes, and is independent of both costs and capacities.

2.2. A SCALING METHOD FOR THE HITCHCOCK TRANSPORTATION PROBLEM. In this section and the following one, we present a technique for solving a minimum-cost flow problem by treating a sequence of problems with the same cost as the given problem, but with "scaled down" capacities which approximate those of the given problem to successively more digits of precision. The efficiency of this scaling method is based on the following two features:

(1) the capacities, and hence the flow augmentations, in the approximate problems are on a coarser scale than in the original problem;

(2) the final solution of each approximate problem yields a good initial flow for the next approximate problem.

We prove that the number of computation steps required by the scaling method is proportional not to the capacities (as in the method of Section 2.1) but to the numbers of digits in the binary representations of the capacities. Roughly speaking, the scaling method is related to the original method as binary arithmetic is to unary arithmetic (i.e. counting).

First we consider a special case in which the scaling technique is particularly simple. The *Hitchcock transportation problem* asks for a maximum flow of minimum cost through a network of the type shown in Figure 1.

The costs and capacities are as follows:

arc $(s, s_i)$ has cost 0 and capacity $a_i$, $i = 1, 2, \cdots, m$;
arc $(t_j, t)$ has cost 0 and capacity $b_j$, $j = 1, 2, \cdots, n$;
arc $(s_i, t_j)$ has cost $d_{ij}$ and capacity $+\infty$, $i = 1, 2, \cdots, m$, $j = 1, 2, \cdots, n$;
the return arc $(t, s)$ has cost 0 and capacity $+\infty$.

It is assumed that $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. The value of a maximum flow is clearly $\sum_{i=1}^{m} a_i$.

The standard interpretation of this problem is well known. Each vertex $s_i$ corresponds to a "source" at which $a_i$ units of a commodity are available; each vertex $t_j$ corresponds to a "destination" which demands $b_j$ units of the commodity. The cost per unit of shipping from $s_i$ to $t_j$ is $d_{ij}$, and a shipping pattern is sought which minimizes the cost of meeting the demands at the destinations from the supplies at the sources.

In the following specialization of the criteria for an extreme flow given in eq. (4), $u_i$ denotes $\pi(s_i)$ and $v_j$ denotes $\pi(t_j)$; also, $f_{ij}$ denotes $f(s_i, t_j)$ when $i \geq 1$ and $j \geq 1$; $f_{0i}$ denotes $f(s, s_i)$ and $f_{j0}$ denotes $f(t_j, t)$.

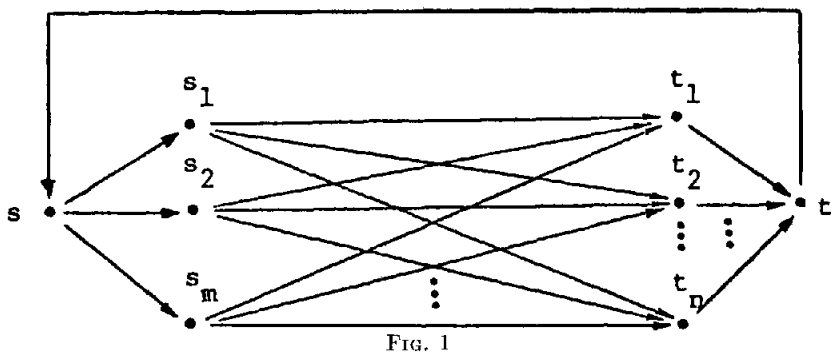



FIG. 1

THEOREM 8. *The flow $f$ is extreme among maximum flows for the network of Figure 1, if and only if there exist $u_0, u_1, \cdots, u_m$ and $v_0, v_1, \cdots, v_m$ such that*

$$u_i - v_j + d_{ij} \geq 0,\ i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n, \tag{5a}$$

$$u_i - v_j + d_{ij} > 0 \Rightarrow f_{ij} = 0,\ i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n, \tag{5b}$$

$$u_0 > u_i \Rightarrow f_{0i} = 0, \tag{5c}$$

$$u_0 < u_i \Rightarrow f_{0i} = a_i, \tag{5d}$$

$$v_j > v_0 \Rightarrow f_{j0} = 0, \tag{5e}$$

$$v_j < v_0 \Rightarrow f_{j0} = b_j \tag{5f}$$

Call a flow $f$ for the Hitchcock problem *pseudo-extreme* if there exist $u_i$ and $v_j$ satisfying (5a) and (5b). A pseudo-extreme maximum flow is extreme; for a maximum flow has $f_{0i} = a_i$, $i = 1, 2, \cdots, m$ and $f_{j0} = b_j$, $j = 1, \cdots, n$. Thus, if (5a) and (5b) are satisfied, we may satisfy (5c)–(5f) by setting $u_0 = \min_{i=1,2,\cdots,m} u_i$ and $v_0 = \max_{j=1,2,\cdots,n} v_j$. For a problem of the Hitchcock type with $\sum a_i \neq \sum b_j$, a pseudo-extreme maximum flow is not, in general, extreme.

Algorithm B can, of course, be used to solve the Hitchcock problem. An alternate method is based on the fact that a maximum pseudo-extreme flow is extreme. A sequence of pairs $(f^0, \pi^0), (f^1, \pi^1), \cdots, (f^k, \pi^k)$ is computed where, for each $k$, $(f^k, \pi^k)$ satisfies (5a) and (5b), so that $f^k$ is pseudo-extreme. The determination of $(f^{k+1}, \pi^{k+1})$ from $(f^k, \pi^k)$ differs from the corresponding step in the previous algorithm in only one respect: if $f^k_{0i} < a_i$ then arc $(s, s_i)$ in $N^{f^k}$ is assigned cost $\bar{\Delta}(s, s_i) = 0$, regardless of $\pi^k$; similarly $\bar{\Delta}(t_j, t) = 0$ if $f_{j0} < b_j$. It is easily checked that, if $(f^k, \pi^k)$ satisfies (5a) and (5b), then so does $(f^{k+1}, \pi^{k+1})$; thus, $f^{k+1}$ is pseudo-extreme if $f^k$ is. If the capacities $a_i$ and $b_j$ are integers then an upper bound on the number of flow augmentations is $\sum_{i=1}^{m} a_i$.

Now we are prepared to present the scaling method. For any nonnegative integer $p$, define Problem $p$ to have the same nodes, arcs, and costs as the given problem, but with the capacities changed as follows: the capacity of $(s, s_i)$ is $\lfloor a_i/2^p \rfloor$ and the capacity of $(t_j, t)$ is $\lfloor b_j/2^p \rfloor$.[3] Thus, the original problem is Problem 0 and, in general, the capacities in Problem $p$ are obtained by deleting the $p$ low-order digits in the binary representations of the original capacities.

[3] "$[x]$" means "greatest integer less than or equal to $x$."

LEMMA 3. *If $f$ is a pseudo-extreme flow in Problem $p$, then $2f$ is a pseudo-extreme flow in Problem $p-1$.*

Choose $l$ such that every finite capacity has at most $l$ digits in its binary expansion; i.e. $a_i < 2^l$, $i = 1, 2, \cdots, m$, and $b_j < 2^l$, $j = 1, 2, \cdots, n$. Then the scaling method computes maximum pseudo-extreme flows successively for Problems $l-1$, $l-2, \cdots, 0$. If $f$ is the maximum pseudo-extreme flow computed in Problem $p$, and $\pi$ is the associated labeling function, then $2f$ is taken as the initial pseudo-extreme flow in Problem $p-1$, with $\pi$ as its associated labeling function.

The following theorem bounds the number of flow augmentations in the solution of a transportation problem by the scaling method.

THEOREM 9. *The number of flow augmentations in applying the scaling method to a transportation problem with integral "supplies" $a_1, a_2, \cdots, a_m$ and integral "demands" $b_1, b_2, \cdots, b_n$ is less than or equal to*

$$max\ (m, n)\left(2 + \left\lfloor log_2 \frac{\sum_{i=1}^{m} a_i}{max\ (m, n)} \right\rfloor\right).$$

PROOF. Let $f_p^*$ denote the value of a maximum flow in Problem $p$. The initial flow in Problem $l-1$ is 0 and, for $p < l$, the initial flow in Problem $p-1$ is $2f_p^*$. Recalling that each augmentation gives a positive integral increase in the flow, the total number of augmentations is bounded above by

$$f_{l-1}^* + \sum_{p=1}^{l-1} (f_{p-1}^* - 2f_p^*) = f_0^* - \sum_{p=1}^{l-1} f_p^*. \tag{6}$$

Now

$$f_p^* = \min\left(\sum_{i=1}^{m} \left\lfloor \frac{a_i}{2^p} \right\rfloor, \quad \sum_{j=1}^{n} \left\lfloor \frac{b_j}{2^p} \right\rfloor\right) \geq 0.$$

We can write $a_i = 2p[a_i/2^p] + r_{ip}$, where $0 \leq r_{ip} \leq 2^p - 1$. Hence,

$$\sum_{i=1}^{m} \left\lfloor \frac{a_i}{2^p} \right\rfloor = \frac{\sum_{i=1}^{m} a_i - \sum_{i=1}^{m} r_{ip}}{2^p} \geq \frac{\sum_{i=1}^{m} a_i}{2^p} - m.$$

Similarly,

$$\sum_{i=1}^{m} \left\lfloor \frac{b_j}{2^p} \right\rfloor \geq \frac{\sum_{j=1}^{n} b_j}{2^p} - n.$$

Let $B$ denote the common value of $\sum_{i=1}^{m} a_i$ and $\sum_{j=1}^{n} b_j$, and let $L$ denote $[\log_2 (B/\max (m, n))]$. Then

$$f_p^* \geq \max\left(0, \frac{B}{2^p} - \max (m, n)\right)$$

and

$$\sum_{p=1}^{l} f_p^* \geq \sum_{p=1}^{L} \left(\frac{B}{2^p} - \max (m, n)\right).$$

Applying this inequality to (6), and noting that $f_0^* = B$, we find that an upper bound on the total number of flow augmentations is

$$B/2^L + L \max(m, n) \leq (L + 2) \max(m, n).$$

This completes the proof. ‖

We remark that this bound on the number of flow augmentations is approximately equal to the number of binary digits required to encode the data of the transportation problem. Each augmentation requires $O(m, n)$ computation steps, so that the number of computation steps in the entire process is bounded by a low-degree polynomial in the size of the problem, as measured by the length of the input text. In this sense the scaling method is a "good" algorithm.[4]

2.3. A Scaling Method for the Minimum-Cost Flow Problem. References [6] and [7] give a simple method of converting any minimum-cost flow problem having $|A|$ arcs and $n$ nodes into an "equivalent" Hitchcock transportation problem with $|A|$ sources, $n$ destinations, and a maximum flow of $\sum_{(u,v)\in A} c(u, v)$. By Theorem 9, the application of the scaling method to such a derived transportation problem requires at most

$$(L + 2)|A|$$

flow augmentations where $L = \log_2 (\sum_{(u,v)} c(u, v)/|A|)$. Thus, the approach of converting to an equivalent transportation problem which is solved by the scaling method yields a good algorithm for the minimum-cost flow problem.

In this section, we consider the direct application of the scaling method to the minimum-cost flow problem. The general approach is clear. Given a minimum-cost flow Problem on a network $N$ with costs $d(u, v)$ and capacities $c(u, v)$, define Problem $p$ as a problem identical with the given one except that the capacity of arc $(u, v)$ is given by $[c(u, v)/2^p]$. Choose $l$ as the least integer such that $2^l > c(u, v)$ for all $(u, v) \in A$. Then the plan is to solve Problems $l - 1, l - 2, \cdots, 0$ successively using Algorithm B, taking twice the final flow in Problem $p$ as the initial flow in Problem $p - 1$. There is a major difficulty, however. If $f_p^*$ is a minimum-cost maximum flow in Problem $p$, then $2f_p^*$ is a flow in Problem $p - 1$, but not, in general, an extreme flow. In the case of the transportation problem this difficulty was not serious, since it was possible to work with pseudo-extreme flows instead of extreme flows. For general minimum-cost flow problems the remedy for this difficulty is somewhat more complex.

We begin by showing that if $f$ is extreme in Problem $p$, then $2f$ is "almost extreme" in Problem $p - 1$. Since $f$ is extreme in Problem $p$, there is a labeling function $\pi$ such that

$$\pi(u) + d(u, v) - \pi(v) > 0 \Rightarrow f(u, v) = 0, \qquad (u, v) \in A,$$

$$\pi(u) + d(u, v) - \pi(v) < 0 \Rightarrow f(u, v) = [c(u, v)/2^p], \qquad (u, v) \in A.$$

Using the inequalities

$$2\left[\frac{c(u, v)}{2^p}\right] \leq \left[\frac{c(u, v)}{2^{p-1}}\right] \leq 2\left[\frac{c(u, v)}{2^p}\right] + 1,$$

[4] The concept of a "good algorithm" is discussed in detail in [2].

we have

$$\begin{aligned} &\pi(u) + d(u, v) - \pi(v) > 0 \Rightarrow 2f(u, v) = 0, \\ &\pi(u) + d(u, v) - \pi(v) < 0 \Rightarrow 2f(u, v) \leq [c(u, v)/2^{p-1}] \leq 2f(u, v) + 1. \end{aligned} \tag{7}$$

But compatibility of $f$ and $\pi$ in Problem $p - 1$ requires

$$\pi(u) + d(u, v) - \pi(v) < 0 \Rightarrow 2f(u, v) = [c(u, v)/2^{p-1}].$$

Thus, $2f$ and $\pi$ fail to be compatible in Problem $p - 1$ by at most one unit of flow on any arc. We give an efficient method of transforming $2f$ to a flow which has the same value and is extreme in Problem $p - 1$. The method can be regarded as a variant of the Fulkerson out-of-kilter algorithm [5] in which the "flow change" and "potential change" phases are combined into a single computation.

We state the method as it applies to an arbitrary integral feasible flow $g$ in Problem $p - 1$, and an arbitrary labeling function $\theta$. Define $\bar{d}(u, v) = \theta(u) + d(u, v) - \theta(v)$. Define $K_{g,\theta}(u, v)$, the *kilter number* of arc $(u, v)$ relative to $g$ and $\theta$, as

$$\begin{cases} g(u, v), & \text{if } \bar{d}(u, v) > 0, \\ 0, & \text{if } \bar{d}(u, v) = 0, \\ \left[\dfrac{c(u, v)}{2^{p-1}}\right] - g(u, v), & \text{if } \bar{d}(u, v) < 0. \end{cases}$$

Thus, $g$ and $\theta$ are compatible if and only if each arc has kilter number zero. Also, relative to $2f$ and $\theta$, the kilter number of each arc is 0 or 1.

The following algorithm derives, from an incompatible pair $(g, \theta)$, a new pair $(g', \theta')$, in such a way that

(i) $K_{g',\theta'}(u, v) \leq K_{g,\theta}(u, v), \qquad (u, v) \in A,$

and

(ii) $\sum_{(u,v)\in A} K_{g',\theta'}(u, v) \leq \sum_{(u,v)\subset A} K_{g,\theta}(u, v) - 1.$

ALGORITHM C: Kilter Number Reduction

(1) Form the augmentation network $N^g$, having $A^g$ as its set of arcs. For each arc $(u, v) \in A^g$, define

$$\bar{\Delta}(u, v) = \begin{cases} \bar{d}(u, v), & \text{if } (u, v) \in A \text{ and } g(u, v) < [c(u, v)/2^{p-1}], \\ -\bar{d}(u, v), & \text{if } (v, u) \in A \text{ and } g(v, u) > 0. \end{cases}$$

Label each arc $(u, v) \in A^g$ with the weight

$$\beta(u, v) = \max(\bar{\Delta}(u, v), 0).$$

(2) Choose an arc $(u^*, v^*)$ of $N^g$ such that

(a) $(u^*, v^*) \in A, \quad \bar{d}(u, v) < 0 \text{ and } g(u, v) < [c(u, v)/2^{p-1}],$
or
(b) $(v^*, u^*) \in A, \quad \bar{d}(u, v) > 0 \text{ and } g(u, v) > 0.$

(3) Let $N^* = \{x \mid x = v^* \text{ or } N^* \text{ has a directed path from } v^* \text{ to } x\}$. For $x \in N^*$, set $\delta(x)$ equal to the minimum weight of a directed path from $v^*$ to $x$. For $x \notin N^*$ set

$$\delta(x) = \max_{\{(v,u)\in A^g \mid v\notin N^*,\ u\in N^*\}} [\theta(u) - \bar{\Delta}(v, u)].$$

For each node $x$, let $\theta'(x) = \theta(x) + \delta(x)$.

(4) If $u^* \notin N^*$, then $g' = g$.
If $u^* \in N^*$, choose a cycle $C$ of $N^g$ consisting of $(u^*, v^*)$ together with a minimum-weight path from $v^*$ to $u^*$. Obtain $g'$ from $g$ by performing a flow augmentation around the cycle $C$.

We make the following assertions (omitting proofs):

(i) For each arc $(u, v)$, $K_{g',\theta'}(u, v) \leq K_{g,\theta}(u, v)$;
(ii) if $(u^*, v^*) \in A$,

$$K_{g',\theta'}(u^*, v^*) \leq K_{g,\theta}(u^*, v^*) - 1;$$

(iii) if $(v^*, u^*) \in A$,

$$K_{g',\theta'}(v^*, u^*) \leq K_{g,\theta}(v^*, u^*) - 1.$$

By iteration of Algorithm C, the pair $(2f, \pi)$ can be converted to an extreme flow for Problem $p - 1$ having the same value as $2f$. Since each iteration reduces the sum of the kilter numbers by an integer, the number of iterations will not exceed $\sum_{(u,v)\in A} K_{2f,\pi}((u, v))$. But, since $K_{2f,\pi}(u, v) \in \{0, 1\}$, the number of iterations is bounded by $|A|$.

We are now in a position to give a complete statement of the scaling algorithm.

ALGORITHM D: Scaling Algorithm for Minimum-Cost Flows

(1) Set $f = 0$ and $\pi = 0$. Choose $l$ such that, for all $(u, v) \in A$, $c(u, v) < 2^l$. Set $p = l - 1$.
(2) Solve Problem $p$ by the algorithm of Section 2.1, using $f$ as the initial flow and $\pi$ as the initial labeling function. After this step $f$ is a maximum flow of minimum cost in Problem $p$, and $\pi$ is compatible with $f$ in Problem $p$.
(3) If $p = 0$, halt. Otherwise replace $p$ by $p - 1$ and replace $f$ by $2f$.
(4) Apply Algorithm C repeatedly, starting with the pair $(f, \pi)$, until a compatible pair $(g, \theta)$ is obtained. Set $f = g$ and $\pi = \theta$. Go to 2.

The number of steps required in applying this algorithm can easily be bounded. The number of executions of Algorithm C in Step (4) is at most $|A|(l - 1)$.

The number of flow augmentations (each requiring an application of Algorithm A to a network with nonnegative weights) is bounded above by

$$f_{l-1}^* + \sum_{p=1}^{l-1} f_{p-1}^* - 2f_p^* = f_0^* - \sum_{p=1}^{l-1} f_p^*. \tag{8}$$

The number of applications of Algorithm A to networks with nonnegative weights to verify that a maximum flow has been reached is $l$.

To put an upper bound on (8) we establish a lower bound on $f_p^*$. Let $T$ be an upper bound on the number of arcs in a cut-set separating $s$ from $t$; i.e. if the nodes are partitioned into sets $X$ and $\bar{X}$ such that $s \in S$ and $t \in \bar{X}$, then the number of arcs directed from a node in $X$ to a node in $\bar{X}$ is less than or equal to $T$. According to the max-flow min-cut theorem ([5])

$$f_p^* = \min \sum_{\substack{u\in X \\ v\in\bar{X}}} \left[\frac{c(u, v)}{2^p}\right] = \sum_{\substack{u\in Y \\ v\in\bar{Y}}} \left[\frac{c(u, v)}{2^p}\right]$$

for some partition $(Y, \bar{Y})$. Now

$$\left[\frac{c(u, v)}{2^p}\right] \geq \frac{c(u, v)}{2^p} - 1,$$

so

$$f_p^* \geq \frac{1}{2^p} \sum_{\substack{u \in Y \\ v \in \bar{Y}}} c(u, v) - T \geq \frac{1}{2^p} f_0^* - T.$$

Substituting this inequality in (8) gives, as an upper bound on the number of flow augmentations,

$$\bar{f}_0 - \sum_{p=1}^{l-1} \left( \frac{1}{2^p} f_0^* - T \right) = \frac{1}{2^p} f_0^* + (l-1)T \leq lT.$$

The following theorem sums up our conclusions.

THEOREM 10. *Let $N$ be a network with $n$ nodes, $|A|$ arcs, and at most $T$ arcs in a cut-set separating $s$ from $t$. Let $l$ be the number of binary digits needed to represent the largest arc capacity. Then the scaling method solves the minimum-cost flow problem for $N$ using not more than $|A|(l-1)$ applications of Algorithm C and not more than $l - 1 + lT$ applications of Algorithm A to networks with nonnegative weights. Each network considered in the algorithm has $n$ nodes and at most $|A|$ arcs.*

REFERENCES

(*Note.* References [1, 3, 6, 7] are not cited in the text.)

1. DINIC, E. A. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Sov. Math. Dokl. 11* (1970), 1277–1280.
2. EDMONDS, J. Paths, trees and flowers. *Canadian J. Math. 17* (1965), 449–467.
3. EDMONDS, J., AND KARP, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. *Combinatorial Structures and Their Applications*. Gordon and Breach, New York, 1970, pp. 93–96 (abstract presented at Calgary International Conference on Combinatorial Structures and Their Applications, June 1969).
4. EDMONDS, J., AND FULKERSON, D. R. Bottleneck extrema. RAND Corp. Memorandum RM-5375-PR (Jan. 1968).
5. FORD, L. R., AND FULKERSON, D. R. *Flows in Networks*. Princeton U. Press, Princeton, N.J., 1962.
6. FULKERSON, D. R. On the equivalence of the capacity-constrained transshipment problem and the Hitchcock problems. RAND Corp. Memorandum RM-2480 (Jan. 1960).
7. WAGNER, H. M. On a class of capacitated transportation problems. *Manag. Sci. 5* (1959), 304 318.

RECEIVED SEPTEMBER 1970; REVISED AUGUST 1971