Two-Commodity Flow

ALON ITAI

Technion—Israel Institute of Technology

ABSTRACT An algorithm is given to find maximum two-commodity flow in an undirected graph The algorithm is an improvement on Hu's two-commodity flow algorithm using the methods of Dinic's single-commodity flow algorithm Karzanov's improvement of Dinic's algorithm can be applied to yield an $O(|V|^3)$ algorithm.

It is shown that finding maximum two-commodity flow in a directed graph is much more difficult, in fact it is as difficult as linear programming Finally, the problem of finding feasible flow in an undirected graph with lower and upper bounds on the edges is shown to be NP-complete even for a single commodity

KEY WORDS AND PHRASES augmenting path, directed graph, flow network, linear programming, max-flow, min-cut, multicommodity flow, NP-complete, polynomially equivalent, undirected graph

CR CATEGORIES 5 25, 5 32, 5 41

1. *Introduction*

Classical flow problems deal with a single commodity which has to be transferred from the source through the network to the terminal. The edges of the network have finite capacity and we have to maximize the flow from the source to the terminal while satisfying both the capacity constraints at the edges and the conservation of flow at the vertices.

Single-commodity flow problems were studied by Ford and Fulkerson [6] who introduced augmenting path algorithms. However, their algorithm is not polynomially bounded. Edmonds and Karp [3] presented an $O(|V||E|^2)$ algorithm and Dinic [2] and Even and Tarjan [5] presented an $O(|V|^2|E|)$ algorithm. Recently, Karzanov [10] implemented Dinic's algorithm in $O(|V|^3)$ time.

A natural generalization is to consider two distinct commodities each with its own source and terminal. As an application, consider a telecommunications network in which telephone and telex share the same lines. The maximum flow indicates the maximum number of bits of information which can pass through the network.

Hu [7] devised an algorithm to find maximum two-commodity real flow in an undirected graph. This algorithm is based upon a max-flow min-cut theorem. First he found a maximum flow of the first commodity while the second commodity flow was zero. Then he used pairs of augmenting paths to recirculate the first commodity and increase the second. The algorithm has the interesting and useful property that if all capacities are even integers the maximum flow found is in integers. The convergence proof is based on this fact. A drawback of this approach is that the number of steps of this algorithm is not bounded by a function of the number of edges but rather by the size of the capacities. This leads in some cases to an exponential number of steps (in terms of the length of the input data).

If we consider an abstract machine which can perform arithmetic on arbitrary real numbers, then Hu's process may not terminate. Guaranteeing termination for arbitrary real numbers is important for yet another reason. Hu stated the max-flow min-cut theorem

Author's address: Computer Science Department, Technion—Israel Institute of Technology, Haifa, Israel

for two-commodity flow. Its proof is based on the assumption that the algorithm terminates. Consequently, his proof is valid only for networks for which the capacities are expressed in rational numbers. The modified algorithm which converges for arbitrary real numbers removes this restriction on the proof.

We change Hu's algorithm in two ways. First, in each augmentation we increase the flow as much as possible, thus eliminating the need to repeat an augmenting path. Second, we find the augmenting paths in order of increasing length by a technique similar to that of Dinic [2] and Even and Tarjan [5]. The resulting algorithm has running time $O(|V|^2|E|)$ regardless of the size of the capacities. (We do assume, however, that all arithmetic operations on the capacities are primitive, i.e. can be done in one step). It is interesting to note here that the inclusion of only one of the improvements does not suffice to guarantee a polynomial time algorithm.

Karzanov's [10] method of improving Dinic's algorithm can be applied to yield an $O(|V|^3)$ algorithm. Ford and Fulkerson have also shown that for a single commodity, integer capacities imply the existence of a maximum flow which is also in integers. For two and more commodities this is no longer true. See Even et al. [4] for a proof that the two-commodity integer flow problem is NP-complete. The two-commodity flow problem for undirected graphs has then a somewhat unique feature. It is polynomially solvable in reals, and if the capacities are integers, the algorithm produces a maximum flow which may not be in integers but in units of one-half (e.g. 7/2, 17/2, 16/2). Yet, if one insists on a solution in integers, the problem is NP-complete.

Considering the complexity of two-commodity flow in a directed network, it is shown that this problem is as difficult to solve as linear programming in the sense that the two problems are polynomially equivalent [9].

Finally, the problem of a single-commodity flow with upper and lower bounds on the edges is considered. Ford and Fulkerson showed that the single-commodity directed case is reducible to the problem of finding maximum flow for a directed single-commodity network with lower bounds equal to zero. We show that for $m \geq 2$ commodities, there exists a reduction to finding maximum directed flow. However, for the undirected case even the single-commodity flow problem with lower and upper bounds is NP-complete.

2. *Undirected Two-Commodity Flow*

2.1 DEFINITIONS. An undirected two-commodity flow network consists of the following:

(1) An undirected finite graph $G = (V, E)$ with no parallel edges and self-loops. An edge between u and v is denoted $[u, v]$.

(2) A capacity function of $c: E \to R^+$, where R^+ is the set of nonnegative real numbers. (We denote the capacity of the edge $[u, v]$ by $c[u, v]$.)

(3) Vertices s_1 and s_2 (not necessarily distinct) which are called *sources.*

(4) Vertices t_1 and t_2 (not necessarily distinct) which are called *terminals.*

Since the graph is undirected, $[u, v] = [v, u]$. However, flows and paths are directed. We shall use the notation $(\cdot\ ,\ )$ for ordered pairs. Abusing the notation, $(u, v) \in E$ means that the edge $[u, v] \in E$.

The problem is to find two feasible flow functions $f_i: V \times V \to R$, $i = 1, 2$; $f_i(u, v) \neq 0$ only if $[u, v] \in E$. The ith flow function indicates the amount of commodity i which passes through the edge. If the flow passes in direction from u to v, then $f_i(u, v) > 0$ and $f_i(v, u) = -f_i(u, v) < 0$.

The flow functions are *feasible* if they satisfy:

(a) Capacity constraint: For every $(u, v) \in E$, $|f_1(u, v)| + |f_2(u, v)| \leq c[u, v]$, indicating that the total flow in both directions along an edge is bounded from above by the capacity of the edge.

(b) Conservation of flow: For each commodity and each vertex $v \in V - \{s_i, t_i\}$,

$\sum_{u \in V} f_i(u, v) = 0$. The amount of flow of each commodity which enters a vertex equals the flow which emanates from it.

For each commodity let the total flow be defined as $F_i = \sum_{v \in V} f_i(s_i, v)$. Our aim is to maximize the total flow, i.e. to maximize the function $F_1 + F_2$.

2.2 THE UNDIRECTED TWO-COMMODITY FLOW ALGORITHM. We follow Hu's algorithm to some extent. To maximize $F_1 + F_2$ we first maximize F_1 while $f_2 = 0$, then increase F_2 without changing F_1 (although f_1 may change on some edges). It will be proved that such a scheme maximizes $F_1 + F_2$. The crux of the method is the way F_2 is increased. Towards this end, pairs of augmenting paths are found. Let π be a path; then $(u, v) \in \pi$ if v immediately succeeds u in π.

For every $(u, v) \in E$ define $\alpha(u, v) = \frac{1}{2}(c[u, v] - f_1(u, v) - f_2(u, v))$. $2\alpha(u, v)$ is the upper bound on the additional flow that can be pushed from u to v through the edge $[u, v]$.

A *forward path* is a simple (s_2, t_2) path (a path from s_2 to t_2 which does not cross itself), $\pi_\alpha = (s_2 = v_0, \ldots, v_r = t_2)$ where $\alpha(v_i, v_{i+1}) > 0$, $i = 0, \ldots, r - 1$. The *residual capacity* of a forward path is $\alpha(\pi_\alpha) = \min\{\alpha(u, v) | (u, v) \in \pi_\alpha\}$.

For every $(u, v) \in E$ define $\beta(u, v) = \frac{1}{2}(c[u, v] - f_1(u, v) + f_2(u, v))$. (Note that in general $\alpha(u, v) \neq \beta(u, v)$.) $2\beta(u, v)$ is the upper bound on the additional flow that can be pushed through $(u, v) \in E$; where the additional first commodity is pushed from u to v and the additional second commodity is pushed from v to u.

A *backward path* is a simple (t_2, s_2) path $\pi_\beta = (t_2 = v_0, \cdots, v_q = s_2)$ where $\beta(v_i, v_{i+1}) > 0$, $i = 0, \ldots, q - 1$. The residual capacity of a backward path is $\beta(\pi_\beta) = \min\{\beta(u, v) | (u, v) \in \pi_\beta\}$.

A *pair of augmenting paths* (π_α, π_β) consists of a forward path π_α and a backward path π_β. The residual capacity of a pair of augmenting paths $\gamma(\pi_\alpha, \pi_\beta) = \min\{\alpha(\pi_\alpha), \beta(\pi_\beta)\}$.

A maximum flow of the first commodity may block the flow of the second commodity. Possibly, a diversion of the flow of the first commodity would allow an increase of the flow of the second commodity. The algorithm finds pairs of augmenting paths. γ units of the first commodity are recirculated through the paths (π_α, π_β), thus enabling the increase of the second commodity by γ units from s_2 to t_2 along each path.

TWO-COMMODITY FLOW ALGORITHM

Step 1 Let the initial f_2 be zero. Find a maximum flow f_1 (from s_1 to t_1).
Step 2. Find a pair of augmenting paths (π_α, π_β) If no such pair exists, stop maximum flow has been reached
Step 3 For every $(u, v) \in \pi_\alpha$, increase the flow of each commodity by γ (in the direction of the path)
Step 4 For every $(u, v) \in \pi_\beta$, increase the flow of the first commodity by γ and decrease the flow of the second commodity by the same amount
Step 5 Go to step 2

2.3 CORRECTNESS OF THE ALGORITHM. To show that the algorithm indeed finds maximum feasible flow, we first show that the resultant flow is feasible (satisfies the capacity and the conservation constraints).

LEMMA 2.1. *Starting with a feasible flow, at the end of each iteration the flow is feasible.*

PROOF. To show conservation of flow, observe that the flow is changed only along augmenting paths. For any vertex other than s_2 and t_2, any path entering the vertex must emanate. Hence the amount of additional flow equals the amount of flow which is subtracted. At s_2 the backward path enters with γ units of f_1 and the forward path emanates with γ units of f_1, so that f_1 is conserved at s_2. The conservation of f_1 at t_2 follows similarly.

To show that the capacity constraints are fulfilled, a detailed case analysis is conducted on the edges for which the flow has been changed.

Four exclusive cases are considered:

Case 1. $(u, v) \in \pi_\alpha$; $(u, v), (v, u) \notin \pi_\beta$.
Case 2. $(v, u), (u, v) \notin \pi_\alpha$; $(u, v) \in \pi_\beta$.
Case 3. $(u, v) \in \pi_\alpha$; $(u, v) \in \pi_\beta$.
Case 4. $(u, v) \in \pi_\alpha$; $(v, u) \in \pi_\beta$.

Reversing the roles of u and v yields a total of eight exhaustive cases. Because of symmetry only the above four cases need be considered.

Let $f_i(u, v)$ be the flow before the augmentation, $f_i'(u, v)$ the flow after augmenting along π_α and π_β (f_i stands for $f_i(u, v)$ when no confusion arises; α, α', β, β' are defined similarly). We discuss only Case 1; the other cases follow similarly.

Case 1. $(u, v) \in \pi_\alpha$; $(u, v), (v, u) \notin \pi_\beta$. $\gamma(\pi_\alpha, \pi_\beta) \leqslant \alpha(\pi_\alpha) \leqslant \alpha(u, v) = \frac{1}{2}(c - f_1 - f_2)$; $f_i' = f_i + \gamma$, $i = 1, 2$.

Proceed to subcases depending on the signs f_1' and f_2'.

(i) $f_1' \geqslant 0, f_2' \geqslant 0$:

$$|f_1'| + |f_2'| = f_1' + f_2' = (f_1 + \gamma) + (f_2 + \gamma)$$
$$\leqslant f_1 + f_2 + 2\alpha \leqslant f_1 + f_2 + (c - f_1 - f_2) = c.$$

(ii) $f_1' \geqslant 0, f_2' < 0$:

$$|f_1'| + |f_2'| = f_1' - f_2' = (f_1 + \gamma) - (f_2 + \gamma) = f_1 - f_2 \leqslant |f_1| + |f_2| \leqslant c.$$

(iii) $f_1' < 0, f_2' \geqslant 0$ as (ii) by symmetry.

(iv) $f_1' < 0, f_2' < 0$:

$$|f_1'| + |f_2'| = -f_1' - f_2' = -(f_1 + \gamma) - (f_2 + \gamma) < -f_1 - f_2 \leqslant c.$$

Q.E.D.

Note that it does not suffice to check only worst subcases (e.g. for Case 1, $f_1', f_2' \geqslant 0$), since α, β and γ depend on the sign and magnitude of f_1 and f_2.

Assuming that the algorithm halts, by Lemma 2.1 the flow is feasible at termination. To show that it is maximum, we digress and recall a few facts concerning flow network theory.

Let $X \subseteq V$, $\bar{X} = V - X$. The set of ordered pairs $(X, \bar{X}) = \{(u, v) | u \in X, v \in \bar{X}, (u, v) \in E\}$ is a *cut*. Its value is $c(X, \bar{X}) = \sum_{(u,v)\in(X,\bar{X})} c[u, v]$. A cut $(X, \bar{X})$ separates u from v if $u \in X$ and $v \in \bar{X}$ or $u \in \bar{X}$ and $v \in X$.

Let $\nu(u_1, \ldots, u_m; v_1, \ldots, v_m)$ denote the value of a minimum cut which separates u_i from v_i ($i = 1, \ldots, m$). Ford and Fulkerson's [6] max-flow min-cut theorem states that for a single commodity the value of a maximum flow is equal to the value of a minimum cut, i.e. $F_1 = \nu(s_1; t_2)$. Let $\tau(u_1, u_2; v_1, v_2)$ denote the value of a minimum cut $(X, \bar{X})$ such that $u_1, u_2 \in X$ and $v_1, v_2 \in \bar{X}$. (Note that in general $\tau(u_1, u_2; v_1, v_2) \geqslant \nu(u_1, u_2; v_i, v_2)$ since $\nu(u_1, u_2; v_1, v_2)$ may be achieved in a cut $(X, \bar{X})$ for which $u_1, v_2 \in X$ and $u_2, v_1 \in \bar{X}$.)

LEMMA 2.2. (Hu). *In an undirected graph,*

$$\nu(s_1, s_2; t_1, t_2) = \min\{\tau(s_1, s_2; t_1, t_2), \tau(s_1, t_2; s_2, t_1)\}.$$

LEMMA 2.3. *If for any feasible flow (f_1, f_2) found by the algorithm there exists no pair of augmenting paths, then*

$$F_1 + F_2 = \nu(s_1, s_2; t_1, t_2).$$

PROOF. Consider two cases:

(a) There exists no forward path. Define $X = \{u | u \in V$ there exists a path $(s_2 = u_0, \ldots, u_r = u)$ from s_2 to u such that $\alpha(u_{i-1}, u_i) > 0, i = 1, \ldots, r\}$. From the definition, $s_2 \in X$, and since there is no forward path then $t_2 \in \bar{X}$.

For all $(u, v) \in (X, \bar{X})$, $\alpha(u, v) = \frac{1}{2}(c[u, v] - f_1(u, v) - f_2(u, v)) = 0$. Thus $c[u, v] = f_1(u, v) + f_2(u, v)$. Since $c[u, v] \geqslant |f_1(u, v)| + |f_2(u, v)|$, then $f_1(u, v), f_2(u, v) \geqslant 0$.

If $f_1(u, v) = 0$ for all $(u, v) \in (X, \bar{X})$, then $f_2(u, v) = c[u, v]$. The total f_2 flow is equal to $c(X, \bar{X}) \geqslant \nu(s_1; t_2)$. Hence, from the single-commodity max-flow min-cut theorem f_2 is a maximum flow (regardless of f_1) and $F_2 = \nu(s_2; t_2)$. Since the algorithm started with F_1 maximized and it was left unchanged throughout the algorithm, $F_1 = \nu(s_1; t_2)$.

$$F_1 + F_2 = \nu(s_1; t_1) + \nu(s_2; t_2) \geqslant \tau(s_1, s_2; t_1, t_2).$$

Otherwise, there exists a pair $(u, v) \in (X, \bar{X})$ such that $f_1(u, v) > 0$; then $s_1 \in X$,

$t_1 \in \bar{X}$ (otherwise to conserve flow there would be a pair $(w, y) \in (X, \bar{X})$ such that $f_1(w, y) < 0$). Therefore,

$$F_1 + F_2 = c(X, \bar{X}) \geq \tau(s_1, s_2; t_1, t_2).$$

(b) There exists no backward path.

We can conclude similarly that $F_1 + F_2 \geq \tau(s_1, t_2; s_2, t_1)$. Consequently,

$$F_1 + F_2 \geq \min\{\tau(s_1, s_2; t_1, t_2), \tau(s_1, t_2; s_2, t_1)\} = \nu(s_1, s_2; t_1, t_2).$$

Since also $F_1 + F_2 \leq \nu(s_1, s_2; t_1, t_2)$, equality has been proved. Q.E.D.

We have just shown the following corollary.

COROLLARY 2.4. *If the algorithm stops, a maximum flow has been found.*

The ordered pair $(u, v) \in E$ is a *bottleneck* (with respect to the paths π_α and π_β and the flows f_1, f_2) if $\alpha(u, v) = \gamma(\pi_\alpha, \pi_\beta)$ and $(u, v) \in \pi_\alpha$ or $\beta(u, v) = \gamma(\pi_\alpha, \pi_\beta)$ and $(u, v) \in \pi_\beta$.

LEMMA 2.5. *If (u, v) is a bottleneck, then after the augmentation one of the paths π_α or π_β can no longer be used.*

PROOF. The proof proceeds by analyzing the cases of Lemma 2.1:

Case 1. $(u, v) \in \pi_\alpha$; $(u, v), (v, u) \notin \pi_\beta$, $\alpha = \gamma$. The new flows are nonnegative:

$$f_1' = f_1 + \alpha = f_1 + \tfrac{1}{2}(c - f_1 - f_2) = \tfrac{1}{2}(c + f_1 - f_2) \geq \tfrac{1}{2}(c - |f_1| - |f_2|) \geq 0$$
$$f_2' \geq 0 \quad \text{similarly.}$$

The new flow saturates the edge:

$$f_1' + f_2' = f_1' + f_2' = (f_1 + \alpha) + (f_2 + \alpha) = f_1 + f_2 + (c - f_1 - f_2) = c.$$

After the augmentation, a forward path cannot pass through (u, v) since

$$\alpha'(u, v) = \tfrac{1}{2}(c - f_1' - f_2') = 0.$$

The other cases follow similarly. Q.E.D.

2.4 FINDING AUGMENTING PATHS. We have not yet specified how to find the pairs of augmenting paths at step 2; if they are not chosen properly, the number of iterations for some networks is exponential. Here we can follow Edmonds and Karp [3], choosing shortest paths. However, following Dinic [2] and Even and Tarjan [5] yields a better time bound.

The algorithm works in phases. In each phase, pairs of augmenting paths with equal lengths are found, i.e. in the course of a phase, two numbers l_α and l_β are determined, such that for all pairs (π_α, π_β) found in the phase, the length of π_α is l_α and that of π_β is l_β.

The flow in the network is increased using the augmenting paths until there exists no forward paths of length l_α or no backward paths of length l_β. At this point, the phase terminates. In the next phase, l_α and l_β do not decrease and at least one of them strictly increases. Consequently, there may be at most $2(|V| - 1)$ such phases. We now describe a single phase, first constructing the forward auxiliary graph G_α.

CONSTRUCTING G_α

1 Perform a breadth-first search from s_2 considering only ordered pairs $(u, v) \in E$ for which $\alpha(u, v) > 0$
2. If the search does not reach t_2, the entire two-commodity flow algorithm terminates.
3 The vertices of the flow network are divided by the search into levels Let the level of s_2 be numbered zero and let l_α be the level of t_2
4 A vertex $v \in V$ belongs to G_α if either its level is less than l_α or $v = t_2$
5 An ordered pair $(u, v) \in E$ is a directed edge of G_α if $\alpha(u, v) > 0$ and the level of v is greater by 1 than that of u

The paths from s_2 to t_2 in G_α correspond to forward paths of length l_α in the current flow network.

The backward auxiliary graph G_β is constructed similarly; this time we start with t_2, end with s_2, and β replaces α. The paths from t_2 to s_2 in G_β correspond to backward paths of length l_β in the current flow network.

Here and in the sequel, statements concerning α, G_α, π_α, and l_α may be stated and proven also for β, G_β, π_β, and l_β.

Let (π_α, π_β) be a pair of augmenting paths (of length l_α and l_β). The two-commodity flow algorithm uses these paths to increase f_2 by $2\gamma(\pi_\alpha, \pi_\beta)$. The following lemma shows how the residual capacity changes as a result of the augmentation: $\alpha(u, v)$ changes only if (u, v) or (v, u) belongs to π_α.

LEMMA 2.6. *Suppose (π_α, π_β) is a pair of augmenting paths, and $(u, v) \in E$. Then*

$$\alpha'(u, v) = \begin{cases} \alpha(u, v) - \gamma, & (u, v) \in \pi_\alpha, \\ \alpha(u, v) + \gamma, & (u, v) \in \pi_\beta, \\ \alpha(u, v), & \textit{otherwise.} \end{cases}$$

PROOF.

Case 1. $(u, v) \in \pi_\alpha$.

Subcase 1.1. $(u, v) \in \pi_\beta$. The additional flow through π_α increases both $f_1(u, v)$ and $f_2(u, v)$ by γ, while the additional flow through π_β increases $f_1(u, v)$ but decreases $f_2(u, v)$. Therefore, $f_1(u, v)$ increases by 2γ, while $f_2(u, v)$ remains unchanged.

$$\begin{aligned} \alpha'(u, v) &= \tfrac{1}{2}(c[u, v] - f_1'(u, v) - f_2'(u, v)) \\ &= \tfrac{1}{2}(c[u, v] - (f_1(u, v) + 2\gamma) - f_2(u, v)) \\ &= \tfrac{1}{2}(c[u, v] - f_1(u, v) - f_2(u, v)) - \gamma = \alpha(u, v) - \gamma. \end{aligned}$$

Subcase 1.2. $(v, u) \in \pi_\beta$. $f_1(u, v)$ remains unchanged since the increases cancel each other. $f_2(u, v)$ increases by 2γ.

$$\begin{aligned} \alpha'(u, v) &= \tfrac{1}{2}(c[u, v] - f_1'(u, v) - f_2'(u, v)) \\ &= \tfrac{1}{2}(c[u, v] - f_1(u, v) - (f_2(u, v) + 2\gamma)) \\ &= \alpha(u, v) - \gamma. \end{aligned}$$

Subcase 1.3. $(u, v), (v, u) \notin \pi_\beta$. Both $f_1(u, v)$ and $f_2(u, v)$ increase by γ.

$$\begin{aligned} \alpha'(u, v) &= \tfrac{1}{2}(c[u, v] - f_1'(u, v) - f_2'(u, v)) \\ &= \tfrac{1}{2}(c[u, v] - (f_1(u, v) + \gamma) - (f_2(u, v) + \gamma)) \\ &= \alpha(u, v) - \gamma. \end{aligned}$$

The remaining cases are proven similarly. Q.E.D.

COROLLARY 2.7. *No forward paths of length less than or equal to l_α are introduced as a result of the augmentation process.*

PROOF. Let π be a new forward path; then π contains an edge $(u, v) \notin G_\alpha$. If $(v, u) \in G_\alpha$, then level(u) = level(v) + 1. Otherwise, (u, v), $(v, u) \notin G_\alpha$ and the value of $\alpha(u, v)$ has not changed. Since $(u, v) \in \pi$, then $\alpha(u, v) > 0$ after and before the augmentation. If level(v) > level(u), then $(u, v) \in G_\alpha$—a contradiction. Therefore, level(u) $\leq$ level(v).

The edges of $\pi \cap G_\alpha$ lead from one level to the following one, and the remaining edges of π do not lead to a higher level. Therefore, the length of π is strictly greater than l_α. Q.E.D.

Consequently, to find all forward paths of length l_α it suffices to find (s_2, t_2) paths in G_α.

A SINGLE PHASE OF THE TWO-COMMODITY FLOW ALGORITHM

1 Construct the auxiliary graphs G_α and G_β (Let l_α be the distance from s_2 to t_2 in G_α, and l_β the distance from t_2 to s_2 in G_β)

2 **While** G_α contains an (s_2, t_2) path $-$ π_α, and G_β contains a (t_2, s_2) path $-$ π_β, **do**

(1) Increase the flow in the network by $\gamma(\pi_\alpha, \pi_\beta)$

(2) Update α and β according to Lemma 2.6

(3) Delete the saturated edges from the appropriate auxiliary graph (i.e., if $\alpha(u, v)$ has been decreased to zero then delete (u, v) from G_α, and if $\beta(u, v) = 0$ then delete (u, v) from G_β) **end**

Implementation Note. By definition, $\gamma(\pi_\alpha, \pi_\beta) = \min\{\alpha(\pi_\alpha), \beta(\pi_\beta)\}$. If $\gamma(\pi_\alpha, \pi_\beta) = \alpha(\pi_\alpha) < \beta(\pi_\beta)$, then we may continue to use π_β in the next iteration. Furthermore, there

is no point in increasing the flow through π_β in this iteration; we may wait until the increase is equal to $\beta(\pi_\beta)$ (or until the end of the phase). This optimization ensures that in each phase all paths used are found and updated at most once. Corollary 2.7 implies that the paths in the updated auxiliary graph also correspond to the forward paths of length l_α in the current flow network.

To find the paths in the auxiliary graphs, a depth-first search is conducted as follows: For G_α we start tracing from s_2 moving through an edge of G_α to a vertex of level 1, from there to a vertex of level 2, etc. If we reach t_2, we have found a forward path. If the depth-first search reaches a deadend, namely, a vertex v from which no edge emanates, we backtrack to the vertex preceding v on the path and erase the last edge of the path from G_α. The search is continued from there. If we cannot proceed from s_2 then the phase is over.

Since G_α has $l_\alpha + 1$ levels, at most l_α edges may be traced until either t_2 is reached or an edge is deleted. In either case at most l_α edges are scanned until an edge is deleted. The next path is found by continuing scanning from the edge nearest s_2 which was deleted from the previous path.

If G_α contains $|E_\alpha|$ edges, then finding the paths of G_α requires $O(|E_\alpha| l_\alpha)$ time. Since by the implementation note each path is updated at most once and there may be at most $|E_\alpha|$ (s_2, t_2) paths in G_α, the entire updating also requires at most $O(|E_\alpha| l_\alpha)$ time.

A similar process is applied to G_β. Thus each phase requires at most $O(|E_\alpha| l_\alpha) + O(|E_\beta| l_\beta) = O(|V||E|)$ time. Since there are at most $O(|V|)$ phases, the entire two-commodity flow algorithm requires $O(|V|^2|E|)$ time.

The previous discussion and Corollary 2.4 are summarized in the following theorem.

THEOREM 2.1. *The two-commodity flow algorithm finds a maximum flow in at most* $O(|V|^2|E|)$ *time.*

Zadeh [14] exhibited a network in which $|V|^3$ augmenting paths are found by Edmonds and Karp's single-commodity flow algorithm. As noted in [5], for this network Dinic's algorithm requires $O(|V|^2|E|)$ time. The network can be modified to show that $O(|V|^2|E|)$ is also a lower bound to steps 2 through 5 of the undirected two-commodity flow algorithm.

2.5. PROPERTIES OF THE ALGORITHM. We present some properties of the algorithm.

(1) Maximum two-commodity flow is achieved with the first commodity having maximum flow.

(2) If the capacities are integers, the maximum flow obtained is in units of one-half. This follows since in this case α,β are integers throughout the algorithm. In this property the algorithm follows Hu's original algorithm. However, the finiteness of our algorithm does not depend on this fact.

(3) The two-commodity max-flow min-cut theorem [7] states that in an undirected two-commodity flow network the maximum flow is equal to the value of the minimum cut: $\nu(s_1, s_2; t_1, t_2)$.

The proof is based on the fact that after a finite number of steps there exists no pair of augmenting paths, and by Lemma 2.3 the flow achieved is equal to the minimum cut. Hu's algorithm is not necessarily finite for arbitrary real capacities. Moreover, one could construct an example, similar to that of Ford and Fulkerson [6, p. 21] in which an infinite series of flows converges to a value smaller than the minimum cut. Therefore, Hu's proof of the two-commodity min-cut max-flow theorem is valid only for networks with integer or rational capacities. The existence of a finite algorithm for networks with arbitrary real capacities removes this difficulty.

(4) We can use the algorithm to solve a related problem: undirected two-commodity real flow with requirements. This problem is similar to the previous one except that two constants R_1, R_2 are given, and it is required to find feasible flow such that $F_1 \geq R_1$ and $F_2 \geq R_2$.

We augment the network with two new sources $\bar{s}_1$, $\bar{s}_2$ and edges $[\bar{s}_1, s_1]$, $[\bar{s}_2, s_2]$ of capacities R_1, R_2, respectively (s_1, s_2 are now ordinary vertices). The maximum flow in the new network is equal to $R_1 + R_2$ if and only if there exists feasible flow in the original network such that $F_1 \geq R_1$ and $F_2 \geq R_2$.

2.6 AN IMPROVED ALGORITHM. Recently Karzanov [10] discovered a more efficient algorithm for finding a maximum single-commodity flow. The method is quite involved and will not be explained in detail. It takes advantage of the fact that the flow added in a single phase of Dinic's algorithm blocks all the augmenting paths in an auxiliary graph. Karzanov finds in $O(|V|^2)$ time a flow that blocks all augmenting paths in that auxiliary graph. This flow is used to augment the flow in the network instead of finding and updating each path separately.

This approach is applicable to two commodities as well. In each phase the additional flow blocks all the augmenting paths in one of the auxiliary graphs. Hence we may first find the additional flow in the phase, then use this flow to augment the flow in the network.

Each phase is conducted as follows: First two auxiliary graphs G_α and G_β are constructed. Then flows f_α in G_α and f_β in G_β which block all the augmenting paths are found. If, for instance, $F_\alpha < F_\beta$ then f_β is replaced by a flow the value of which is equal to F_α. (This flow may be obtained by first adding to G_β a new vertex $\bar{t}_2$ and a new edge $(\bar{t}_2, t_2)$ of capacity F_α and then finding a blocking flow from $\bar{t}_2$ to s_2 in the new G_β.)

Finding f_α and f_β of the same value involves at most three applications of Karzanov's method, hence $O(|V|^2)$ time. Constructing the two auxiliary graphs and updating the flow requires $O(|E|)$ time. Since there are at most $O(|V|)$ phases, the entire algorithm requires $O(|V|^3)$ time.

3. *Directed Two-Commodity Flow*

3.1 DEFINITION AND BASIC PROPERTIES. In directed flow networks the graph is directed and the commodities flow only in the direction of edges. (A directed edge from u to v is denoted (u, v).)

(a) The capacity constraint is $f_1(u, v) + f_2(u, v) \leq c(u, v)$, $(u, v) \in E$.

(b) The conservation law states that for all $i = 1, 2$ and for all vertices $v \in V - \{s_i, t_i\}$, $\sum_{(u,v)\in E} f_i(u, v) = \sum_{(v,w)\in E} f_i(v, w)$.

Apparently, the directed case is more difficult than the undirected one.

The properties of single-commodity flow do not carry on to the two-commodity directed case:

(i) The max-flow min-cut theorem does not hold. In Figure 1 the maximum flow is obtained when $F_1 = \frac{1}{2}$, $F_2 = 1$; $F_1 + F_2 < 2$, the value of a minimum cut.

(ii) Even if the capacities are integer, the maximum flow may be rational (not necessarily in units of one-half). In Figure 2 all the capacities are equal to 1 and the maximum flow is depicted on each edge.

(iii) The maximum cannot always be achieved when the flow of the first commodity is maximum (in Figure 1 if $F_1 = 1$, then $F_2 = 0$, $F_1 + F_2 = 1$, which is less than the maximum possible flow).

However, all multicommodity flow problems may be considered as special cases of linear programming [6] (Letting $f_i(u, v)$ be the variables, all the constraints and the target function are linear with coefficients $\{-1, 0, 1\}$.)

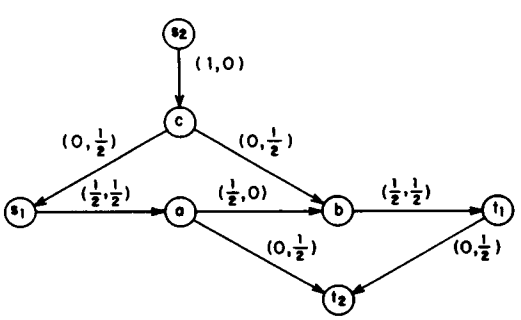

FIG 1

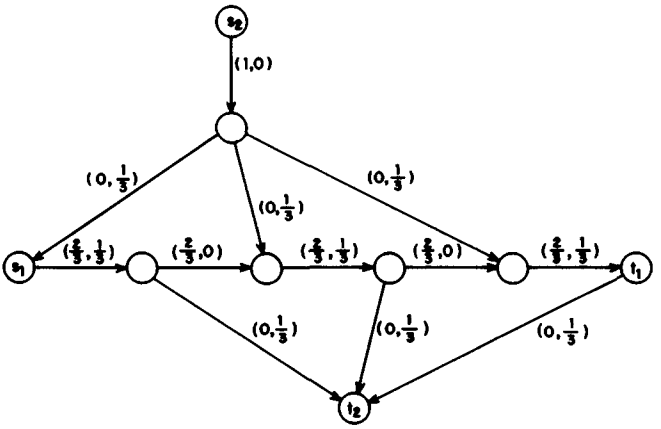

FIG 2

The worst case behavior of linear programming is not well understood: No upper bound less than exponential has been proved and in many cases exponential worst case behavior has been demonstrated [11, 14]. On the other hand, it is not known whether linear programming is NP-complete [9]. Furthermore, there seems to be some evidence to the contrary: The complement of linear programming (all sets of linear inequalities which do not have a feasible solution) is a member of the class NP. None of the NP-complete problems are known to have this property; if the complement of any NP-complete problem belongs to NP, then the complements of all NP problems also belong to NP, which seems quite unlikely. Ladner [12] has shown that if P $\neq$ NP there is an infinite hierarchy of "polynomial equivalent" degrees (equivalence classes) within NP. The lowest degree is P: all the problems solvable in polynomial time; the highest is the NP-complete degree. Possibly, linear programming belongs to an intermediate degree (i.e. it cannot be solved by a polynomially time bounded deterministic Turing machine, but on the other hand not all the problems of NP are reducible to it). We show that directed two-commodity real flow is polynomially equivalent to linear programming. Thus any polynomial algorithm for the flow problem would yield such an algorithm for linear programming.

While proving this, we obtain some auxiliary results which are interesting for their own merit.

3.2 SOME SIMPLIFIED LINEAR PROGRAMMING PROBLEMS. The following problems are polynomially equivalent.

(1) LP: Linear Programming. Given a matrix A, vectors b, c (a_{ij}, b_i, $c_j \in Z$, the integers) and an integer K. Determine whether there exists a nonnegative rational vector x ($x_j \in Q^+$) such that $Ax \leqslant b$ and $cx \geqslant K$.

(2) LI: Linear Inequalities. Given A, b as in LP. Determine whether there exists a nonnegative rational vector x such that $Ax \leqslant b$.

(3) LE: Linear Equalities. As LI except that "=" substitutes for "$\leqslant$". From the classical theory of linear programming LP, LI and LE are polynomially equivalent [6]. Without loss of generality we assume that the vector b, the right-hand side (r.h.s.), is nonnegative.

(4) $[l, u]$ LE: LE with Bounded Coefficients. An LE problem in which a_{ij}, b_i are integers between l and u.

LEMMA 3.1. *LE* $\propto$ [−2, 2] *LE*. ($\propto$ *denotes "polynomially reducible"* [9].)

PROOF. The proof is based on bitwise decomposition of each equation of an LE problem. For example,

(1) Let $5x_1 + 3x_2 - 7x_3 = 6$ be one of the equations. Then it may be rewritten:

$$(1x_1 + 1x_2 - 1x_3)2^0 + (0x_1 + 1x_2 - 1x_3)2^1 + (1x_1 + 0x_2 - 1x_3)2^2 = 0\ 2^0 + 1\ 2^1 + 1 \cdot 2^2.$$

Consider the computation of each bit, thus obtaining an equation for each power of 2.

(ii) $1x_1 + 0x_2 - 1x_3 = 0,$
$0x_1 + 1x_2 - 1x_3 = 1,$
$1x_1 + 0x_2 - 1x_3 = 1.$

However, the system of equations (ii) has no solution. The reason is that in (i) there is a carry from the computation of one bit to that of the higher bit. To overcome this difficulty, we introduce the following system:

(iii) $1x_1 + 0x_2 - 1x_3 - 2(c_0 - d_0) = 0,$
$0x_1 + 1x_2 - 1x_3 + (c_0 - d_0) - 2(c_1 - d_1) = 1,$
$1x_1 + 0x_2 - 1x_3 + (c_1 - d_1) = 1.$

The variables c_k, d_k pass the carry from one equation to the next. (The carry may be positive or negative; since the variables are required to be nonnegative, two such variables are introduced so that their difference can obtain any real value.)

From any solution to (i) it is easy to derive a solution to (iii). For example, let $x_1 = 1.6$, $x_2 = 4$, $x_3 = 3$ be a solution to (i). Then x and $c_0 = 0$, $c_1 = 0.4$, $d_0 = 0.2$, $d_1 = 0$ satisfies (iii).

In the other direction, if (x, c, d) solves (iii) then x solves (i).

The above process is performed on all the n equations, obtaining a system of nK equations in $(n + 1)K - 1$ variables ($K = 1 + \lceil \log \max\{|a_{ij}|: i = 1, \ldots, n; j = 1, \ldots, m\} \cup \{|b_j|: j = 1, \ldots, m\} \rceil$). This reduction is polynomial in the size of the input.

The coefficients and the r.h.s. of the new system are integers in the region $[-2, 2]$. The new system is solvable if and only if the original system is. Q.E.D.

LEMMA 3.2. $[-2, 2]\ LE \propto [-1, 1]\ LE$.

The construction which transforms any $[-2, 2]$ LE problem to an equivalent $[-1, 1]$ LE problem is sketched as follows.

For every occurrence of the variable x_j with coefficients $a_{ij} = \pm 2$ put $\pm(x_j + x_j')$ and add the equation $x_j - x_j' = 0$ If $b_i = 2$, change $\sum_{j=1}^{m} a_{ij}x_j = b_i$ to

$$\sum_{j=1}^{m} a_{ij}x_j - z_i = 1; \qquad z_i = 1.$$

3.3 HOMOLOGOUS FLOW. Following Berge and Ghouila-Houri [1] we introduce homologous flow problems and show that they are polynomially equivalent to the problems of the previous section.

First, consider a generalization of the capacity rule: An (l, u) *flow network* is a flow network in which every edge (v, w) has a lower bound, $l(v, w)$, and an upper bound, $u(v, w)$. $((l(v, w), u(v, w))$ is a pair of real numbers, called the *generalized capacity*. An edge is nonrestricted if its generalized capacity is $(0, \infty)$.) A flow f is *feasible* if the generalized capacity rule holds, i.e. $\forall (v, w) \in E$, $l(v, w) \leq f(v, w) \leq u(v, w)$. We wish to determine whether an (l, u) flow network has a feasible flow.

Two edges (v, w) and (v', w') are *homologous* if it is required that $f(v, w) = f(v', w')$. A *homologous flow network* consists of a single commodity (l, u) flow network with pairs of homologous edges. We wish to determine whether there exists a feasible flow which satisfies the homologous requirements.

Homologous flow is polynomially equivalent to LP, as seen by the following lemma.

LEMMA 3.3. $[-1, 1]\ LE \propto$ *homologous flow.*

PROOF. Let

(i) $\sum_{j=1}^{m} a_{ij}x_j = b_i$, $i = 1, \ldots, n$ be an instance of $[-1, 1]$ LE. For $\sigma = -1, 0, 1$, let $J_\sigma^i = \{j | a_{ij} = \sigma\}$. Then (i) is equivalent to

(ii) $\sum_{j \in J_1^i} x_j - \sum_{j \in J_{-1}^i} x_j = b_i$.

The homologous flow network constructed below has a feasible flow (which also satisfies the homologous requirements) if and only if (i) (or (ii)) has a solution.

The homologous flow network consists of n sections. Each section contains $m + 5$ vertices $\{v_1^i, \ldots, v_m^i, y^i, z^i, J_{-1}^i, J_0^i, J_1^i\}$. The edges in the ith section depend on the ith

equation. For $\sigma = -1, 0, 1$, if $j \in J_\sigma^i$ then v_j^i is connected to J_σ^i by a nonrestricted edge. J_1^i is connected to z^i by an edge of generalized capacity (b_i, b_i). J_1^i and J_{-1}^i are connected to y^i by nonrestricted homologous edges (i.e. $0 \leqslant f(J_1^i, y^i) = f(J_{-1}^i, y^i) < \infty$). J_0^i and y^i are connected to z^i by nonrestricted edges. (See Figure 3.)

The network contains an additional vertex $s = z^0$ which is the source. The terminal is $t = z^n$. For each j ($j = 1, \ldots, m$) the network contains the nonrestricted pairwise homologous edges $(z^0, v_j^1), (z^1, v_j^2), \ldots, (z^{n-1}, v_j^n)$.

Given a solution x to (i) we define a feasible flow as follows: $x_j = f(z^0, v_j^1) = \cdots = f(z^{n-1}, v_j^{n-1})$. The edges are homologous so that the value of the variable x_j stays the same in all the equations.

$$f(J_1^i, y^i) = f(J_{-1}^i, y^i) = \sum_{j \in J_{-1}^i} x_j = \sum_{j \in J_1^i} x_j - b_i.$$

The flow on the remaining edges is defined so that the conservation rule is preserved.

Given a feasible flow it is easy to construct a solution to (i). Therefore, the equalities are satisfiable if and only if there exists a feasible flow. Q E.D.

3.4. SELECTIVE FLOW. In this section we define additional flow networks and show that the existence of feasible flow in these networks is polynomially equivalent to solving the problems of the last two sections.

An edge is *selective* if only a specific commodity may pass through it. In such networks we specify the commodities which pass through each edge. A *selective* (l, u) $2CF$ is a two-commodity directed flow network with lower and upper bounds on the sum of the flows on each edge. We get rid of the homologous requirement by introducing a second commodity.

LEMMA 3.4. *Homologous flow* $\propto$ *selective* (l, u) $2CF$.

PROOF. Without loss of generality, we may assume that each edge (v, w) is homologous to at most one other edge. (If (v, w) is homologous to (y, z) and (y', z') then replace (v, w) by two edges (v, x) and (x, w) such that x is a new vertex incident only to these two edges and (v, x) is homologous to (y, z) and (x, w) is homologous to (y', z').)

Let (v, w) and (y, z) be homologous edges both with generalized capacity (l, u). Replace these edges by the construction of Figure 4.

The vertices vw, vw', yz, and yz' are new. s_2 and t_2 are the source and the terminal of the second commodity. The permissible subsets appear above the edges. C is a large constant.

Clearly, $f_1(vw, vw') + f_2(vw, vw') = C$, $f_1(yz, yz') + f_2(yz, yz') = C$.

Since $f_2(vw, vw') = f_2(yz, yz')$, then $f_1(vw, vw') = f_1(yz, yz')$, and the effect of homologous edges is achieved.

There remains a subtle point: The constant C must be as large as the largest flow on any edge. Since we allowed nonrestricted edges it is not clear whether a uniform bound can be found a priori. Moreover, the reduction is polynomial only if the number of bits in the representation of C is bounded by a polynomial of the number of bits of the input.

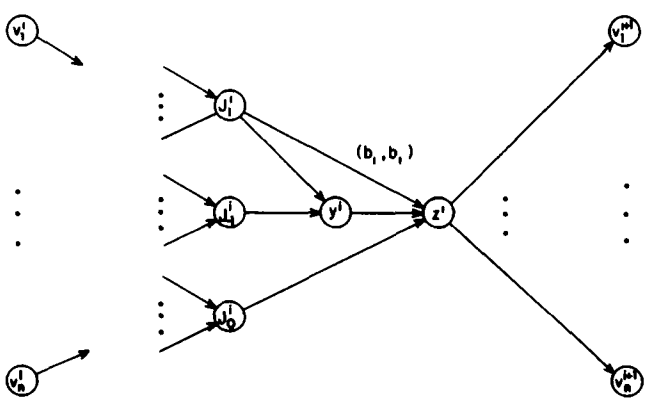

FIG 3

As already mentioned, every flow problem is a special case of linear programming. For homologous flow the entries of the resultant matrix are $\{-1, 0, +1\}$. The entries of the r.h.s. are bounded by the maximum capacity. From the theory of linear programming we learn that the existence of a solution implies the existence of a basic solution. Let A_{basic} be the submatrix corresponding to a basic solution. A^i_{basic} is A_{basic} except that the ith column is replaced by the r.h.s. By Cramer's rule,

$$|x_i| = |\det(A^i_{\text{basic}})| / |\det(A_{\text{basic}})| \leq |\det(A^i_{\text{basic}})| .$$

Let M be a bound on the finite capacities; then the r.h.s. is also bounded by M. By Hadamar's inequality, $|x_i| \leq M\ n^{n/2}$. Therefore, $C \leq M\ n^{n/2}$, and its representation requires at most $\log M\ n^{n/2} = \log M + \frac{1}{2}n \log n$ bits. Hence the reduction is polynomial. Q.E.D.

We can drop the requirement for selective edges if we maintain the requirement for lower and upper bounds.

LEMMA 3.5. *Selective* (l, u) $2CF \propto (l, u)$ $2CF$.

PROOF. We simulate the selective edges by changing each selective edge (v, w) of capacity (l, u) which accepts only commodity i into the structure of Figure 5 (vw and vw' are new vertices).

Without loss of generality we may assume that no edge enters s_i or emanates from t_i. The capacity requirements are fulfilled if and only if $l \leq f_i(v, w) \leq u$ and the flow of the other commodity is zero. Q.E.D.

3.5. FLOW WITH REQUIREMENTS AND MAXIMUM FLOW. A two-commodity flow network with requirements (2CFR) consists of two real numbers R_1 and R_2 and a two-commodity directed flow network with only upper bounds on the edges (all lower bounds are equal to zero). A flow is feasible if it satisfies the capacity and conservation rules and $F_i \geq R_i$, $i = 1, 2$

LEMMA 3.6. (l, u) $2CF \propto 2CFR$.

PROOF. Given an instance of (l, u) 2CF, an equivalent instance of 2CFR is constructed by changing the graph G into $\bar{G}$ as follows:

(i) The sources and the terminals of $\bar{G}$ are the new vertices $\bar{s}_1$, $\bar{s}_2$, $\bar{t}_1$, $\bar{t}_2$, respectively. (The vertices s_1, s_2, t_1, and t_2 become ordinary vertices which satisfy the conservation rule.)

(ii) Every edge (x, y) of capacity (l, u) is replaced by the construction of Figure 6.

(iii) Let M be the sum of all the upper bounds of all the edges of G. For $i = 1, 2$, let z_i, z_i' be new vertices and add the construction of Figure 7.

The requirements are $R_1 = R_2 = 2M$. We show that the two flow problems are equivalent.

(a) If there exists a legal flow in the $\bar{G}$ then there exists a legal flow in G.

Let $\bar{f}$ be a flow in $\bar{G}$ which satisfies the requirements, then $\bar{f}$ saturates all the edges

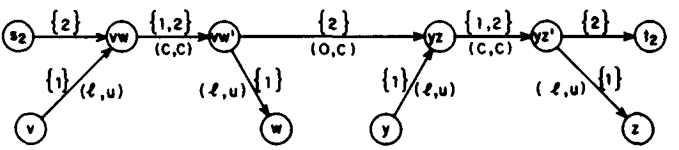

FIG 4

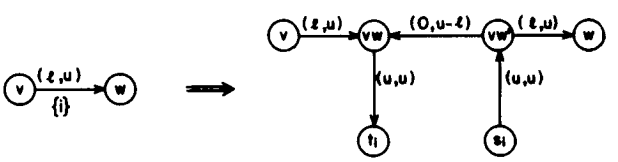

FIG 5

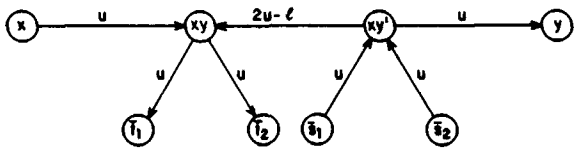

FIG. 6

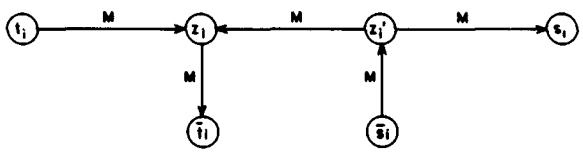

FIG 7

incident with $\bar{s}_i$ and $\bar{t}_i$ ($i = 1, 2$). For $\bar{t}_i$ this follows because the sum of the capacities of all the edges entering $\bar{t}_i$ is $\sum^{(x,y)\in E} c(xy, \bar{t}_i) + c(z_i, \bar{t}_i) = 2M = R_i$.

The flow f in G is defined as $f_i(x, y) = \bar{f}_i(x, xy)$ $(= \bar{f}_i(xy', y))$. Obviously, this flow fulfills the conservation rule and is less than the upper bound. It remains to demonstrate that the lower bounds are satisfied.

Because of the conservation of $\bar{f}_i$ at xy, $\bar{f}_i(x, xy) + \bar{f}_i(xy', xy) = \bar{f}_i(xy, \bar{t}_i) = u$. Therefore, $\sum_i f_i(x, y) = \sum_i \bar{f}_i(x, xy) = 2u - \sum_i \bar{f}(xy', xy) \geq 2u - (2u - l) = l$.

(b) If there exists a legal flow f in the original graph G, then there exists a legal flow $\bar{f}$ in the new graph $\bar{G}$.

Define:

(i) $\bar{f}_i(x, xy) = \bar{f}_i(xy', y) = f_i(x, y), \bar{f}_i(xy', xy) = u - f_i(x, y), \bar{f}_i(xy, \bar{t}_i) = \bar{f}_i(\bar{s}_i, xy') = u$.

(ii) $\bar{f}_i(t_i, z_i) = \bar{f}_i(z_i, s_i) = F_i, \bar{f}_i(z_i', z_i) = M - F_i, \bar{f}_i(z_i, \bar{t}_i) = \bar{f}_i(\bar{s}_i, \bar{z}_i') = M$.

For these edges the flow of the other commodity is zero.

By definition the requirements are fulfilled. Flow is conserved at the vertices of $\bar{G}$ since it is conserved in G. To complete the proof we show that the capacity constraints are fulfilled. For the edges (x, xy), (xy', y), $(z_i, \bar{t}_i)$, $(\bar{s}_i, z_i')$, this follows from the construction of the flow. For (xy', xy) the following holds.

Since $\sum_i f_i(x, y) \geq l(x, y)$, then $\bar{f}_i(xy', xy) = u(x, y) - f_i(x, y)$ and $\sum_i \bar{f}_i(xy', xy) = 2u(x, y) - \sum_i f(x, y) \leq 2u(x, y) - l(x, y) = c(xy', xy)$. Q.E.D.

Note. This construction is easily generalized to m-commodity flow. In fact, this reduction continues Ford and Fulkerson's [6] reduction for the single-commodity case. (Their original construction does not work for $m \geq 2$.)

Let 2CF be the problem of maximizing the sum of the flow $(\max(F_1 + F_2))$.

LEMMA 3.7. $2CFR \propto 2CF$.

The proof is similar to that of property (4) of Section 2.5.

3.6 FROM LINEAR PROGRAMMING TO TWO-COMMODITY FLOW. Since all the previous problems are special cases of linear programming, we may summarize the previous lemmas as follows.

THEOREM 3.1. All the following problems are polynomially equivalent: LP, LI, LE, [−1, 1]LE, homologous flow, (l, u) 2CF, 2CFR, 2CF.

Notes. (1) We can generalize all the previous two-commodity flow problems to $m \geq 2$ commodities. Therefore, all these problems are polynomially equivalent to linear programming.

(2) The reductions increase the size of the problem linearly except that of Lemma 3.4 where the size might grow by $n \log n$.

(3) Starting from integer programming [9] instead of from linear programming, the reductions would still carry through while requiring all variables to be nonnegative integers. Thereby, we have given another proof to the fact that two-commodity integer flow is NP-complete [4].

4. *Undirected Flow Networks with Lower Bounds*

For directed flow networks, Lemmas 3.6 and 3.7 showed the computational equivalence of networks with lower bounds and networks without lower bounds. In this section we show that for undirected networks a similar construction is quite improbable, since for undirected networks the former problem is NP-complete and the latter is polynomially solvable [2, 3, 5, 6, 10]. To this end we define three auxiliary network flow problems and show that they are all NP-complete. For all three, it is required to determine whether there exists feasible flow.

P1. An undirected single commodity flow network with lower and upper bounds on the edges. The vertices s and t are each incident with a single edge $([s, s''], [t'', t])$ for which the lower and upper bounds are equal to F.

P2. A single-commodity mixed flow network with lower and upper bounds (some edges are directed and some are undirected). All flow enters t (emanates from s) through a single directed edge (t', t) $((s, s'))$ of capacity (F, F).

P3. Undirected single-commodity flow network with lower and upper bounds.

Let equal-occurrence SAT be the satisfiability problem of Boolean expressions in conjunctive normal form for which each literal appears exactly k times, for some integer k [9].

LEMMA 4.1. *Equal occurrence SAT is NP-complete.*

PROOF. We show that SATISFIABILITY $\propto$ equal occurrence SAT.

Let φ be a Boolean expression in conjunctive normal form, in which x_i ($\bar{x}_i$) occurs k_i ($\bar{k}_i$) times. Let $k = 1 + \max\{k_i, \bar{k}_i | i = 1, \ldots, n\}$. Construct the Boolean expression $\psi = \varphi \cdot D_i$ D_n, where $D_i = (x_i + \quad + x_i + \bar{x}_i + \quad \cdot \quad + \bar{x}_i)$, and x_i ($\bar{x}_i$) occurs $k - k_i \geq 1$ ($k - \bar{k}_i \geq 1$) times in D_i.

Each literal appears exactly k times in ψ. Furthermore, ψ is true exactly when φ is. Obviously, ψ can be computed from φ in polynomial time. Q.E.D

LEMMA 4.2. *P2 is NP-complete.*

PROOF. We show that equal occurrence SAT $\propto$ P2.

Let ψ be an instance of equal occurrence SAT with p clauses $C_1, \ldots, C_p$ and n variables $x_1, \ldots, x_n$ such that each literal occurs exactly k times. We construct the network depicted in Figure 8. In addition to the edges explicitly drawn, if x_i ($\bar{x}_i$) occurs q times in the clause C_j, there is an edge of capacity $(0, q)$ between the vertex x_i and the vertex C_j.

The resultant network is an instance of P2 and can be constructed from ψ in polynomial time. It remains to show that the network contains feasible flow if and only if ψ is satisfiable.

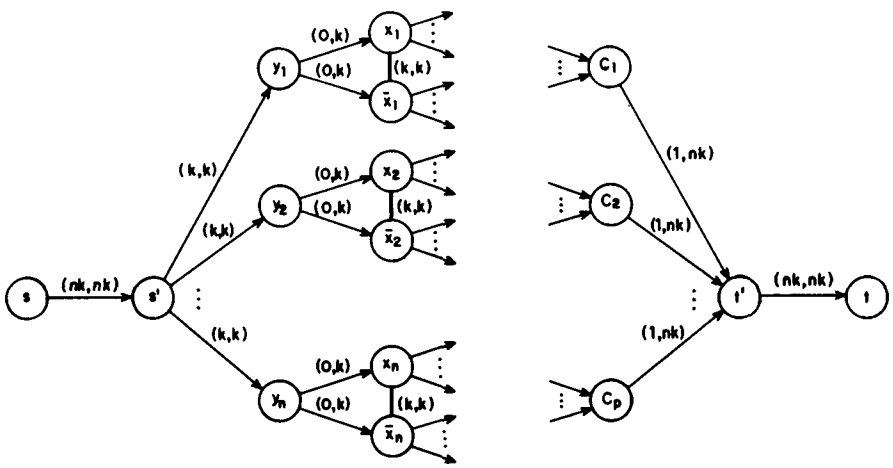

FIG 8

If there exists feasible flow, then the undirected edge $[x_i, \bar{x}_i]$ must be saturated. Therefore, all flow in the triangle $(y_i, x_i, \bar{x}_i)$ must pass either from y_i to $\bar{x}_i$ and then to x_i (in which case x_i is designated true) or from y_i to x_i and then to $\bar{x}_i$ (in which case x_i is false). In the former case, to conserve flow at x_i the edge (x_i, C_j) must be saturated; otherwise $(\bar{x}_i, C_r)$ is saturated. Since $f(C_j, t) \geq l(C_j, t) = 1$, to conserve flow at C_j, there exists an edge (z, C_j) with positive flow. The vertex z corresponds to a literal which belongs to C_j. Since flow emanates from z, z has been designated true, and it causes the clause C_j to be satisfied. Since this applies for all C_j, $j = 1, \ldots, p$, the expression ψ is satisfiable.

The other direction follows immediately. Q.E D.

LEMMA 4.3. *P1 is NP-complete.*

PROOF. We show that P2 $\propto$ P1.

We shall simulate the directed edges by structures of undirected edges. The edge (t', t) $((s, s'))$ is replaced by $[t'', t]$ and $[t'', t']$ ($[s'', s']$ and $[s, s']$) of capacities (F, F) and $(F + U, F + U)$, respectively (U is the sum of the upper bounds over all the directed edges except (t', t) and (s, s')). See Figure 9.

Let (a, b) $(\neq(s, s'), (t', t))$ be a directed edge of capacity (l, u); it is replaced by the structure of undirected edges (Figure 10). (The vertices ab and ab' get introduced only once in the entire construction.)

Without loss of generality, $f(s, s'')$, $f(t'', t) \geq 0$ (otherwise, reverse the direction of the flow.) Since t is connected only to t'', $f(t', t'') = F$ and $f(ab, t'') = u$. To satisfy the conservation rule at ab, $f(a, ab) \geq 0$. Consequently, $l \leq f(a, ab) \leq u$ and $0 \leq f(ab', ab) = u - f(a, ab) \leq u - l$. Conclude similarly that $f(s'', ab') = u$ and $f(ab', b) = f(a, ab) \geq l$ and the total flow from a to b is nonnegative.

A flow on the structure which simulates the flow on the directed edge is easily constructed. Q.E.D.

Since P1 is a special case of P3 we have proven the following theorem.

THEOREM 4.1. *The problem of determining whether there exists a feasible flow in an undirected single commodity network with lower and upper bounds on the edges is NP-complete.*

5. *Conclusions*

When comparing directed and undirected network flow problems we see that some

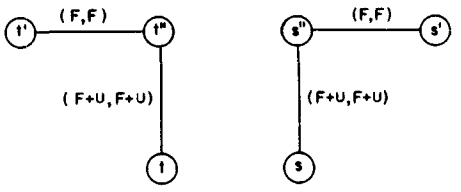

FIG 9

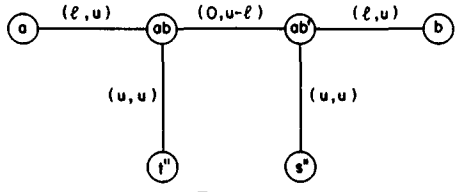

FIG 10

TABLE I SUMMARY OF RESULTS

The problem	Number of commodities		
	1	2	*m*
Maximum flow			
undirected	$\|V\|^3$	$\|V\|^3$	?
directed	$\|V\|^3$	LP	LP
Maximum integer flow			
undirected and directed	$\|V\|^3$	NPC	NPC
Lower and upper bounds			
undirected	NPC	NPC	NPC
directed	$\|V\|^3$	LP	LP

Notes $|V|^3$ There exists a $O(|V|^3)$ time algorithm LP Polynomially equivalent to linear programming NPC NP-complete

problems are easier for undirected networks; whereas other problems are easily solvable in the directed case, while notoriously difficult for undirected networks. Table I summarizes the results.

ACKNOWLEDGMENTS. I wish to thank Prof. S. Even who supervised the research, Dr. M. Rodeh, and an anonymous referee whose numerous comments helped bring the paper to its present form.

REFERENCES

1 BERGE, C., AND GHOUILA-HOURI, A. *Programming, Games and Transportation Networks.* Methuen, London, England, 1965
2 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
3 EDMONDS, J., AND KARP, R. M. Theoretical improvements in algorithm efficiency for network flow problems. *J. ACM 19* (1972), 248–264
4 EVEN, S., ITAI, A., AND SHAMIR, A. On the complexity of timetable and multi-commodity flow problems. *SIAM J. Compting 5* (1976), 691–703
5 EVEN, S., AND TARJAN, R. E. Network flow and testing graph connectivity. *SIAM J. Compting. 4* (1975), 507–518
6 FORD, L. R. JR., AND FULKERSON, D. R. *Flows in Networks.* Princeton U. Press, Princeton, N. J., 1962
7 HU, T. C. Multi-commodity network flows. *J. ORSA 11* (1963), 344–360, also in *Integer Programming and Network Flows*, Addison-Wesley, Reading, Mass., 1969
8 ITAI, A. Multi-commodity flow. Ph. D. Diss., Feinberg Graduate School, Weizmann Inst. Sci., Rehovot, Israel, 1976
9 KARP, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. N. Miller and J. W. Thatcher, Eds., Plenum Press, New York, 1972, pp. 85–104
10 KARZANOV, A. V. Determining the maximal flow in a network by the method of preflow. *Soviet Math. Dokl. 15* (1974), 434–437
11 KLEE, V. L., AND MINTY, G. J. How good is the simplex algorithm. In *Inequalities III*, O. Shisha, Ed., Academic Press, New York, 1972, pp. 159–175
12 LADNER, R. E. On the structure of polynomial time reducibility. *J. ACM 22* (1975), 155–171
13 ZADEH, N. A bad network problem for the simplex method and other minimum cost flow algorithms. *Math. Programming 5* (1975), 255–266
14 ZADEH, N. Theoretical efficiency of the Edmonds-Karp algorithm for computing maximal flows. *J. ACM 19* (1972), 184–192

RECEIVED JANUARY 1976, REVISED NOVEMBER 1977