A POLYNOMIAL-TIME PARAMETRIC SIMPLEX ALGORITHM FOR THE MINIMUM COST NETWORK FLOW PROBLEM

James B. Orlin

Sloan W.P, No. 1484-83 September 1983

A Polynomial-Time Parametric Simplex Algorithm for the Minimum Cost Network Flow Problem

by James B. Orlin

Sloan School of Management

MIT

Abstract

In this paper we consider the minimum cost network flow problem: $\min(cx : Ax = b, x \geq 0)$, where A is an m x n vertex-edge incidence matrix. We show how to solve this problem as a parametric linear program with $O(m\, b^*)$ pivots, where b^* is the number of 1's in the binary representation of b. The parametric formulation is non-linear and is based on Edmonds-Karp scaling technique.

In this paper we consider the minimum cost network flow problem (1).

$$\begin{aligned} &\text{Minimize} && cx \\ &\text{Subject to} && Ax = b \\ & && x \geq 0 , \end{aligned} \tag{1}$$

where A is a full row rank $m \times n$ matrix in which there is at most one 1 and at most one -1 in each column and the remaining entries are all 0. Moreover, b is an integral m-vector, and c is a real n-vector.

Zadeh (1978) showed that the simplex procedure as applied to (1) may take an exponentially large number of pivots. Edmonds and Karp (1972) developed a scaling procedure in conjunction with the out-of-kilter method that solves the minimum-cost network flow problem in a polynomial number of steps. The purpose of this note is to provide a parametric simplex procedure that solves the minimum-cost network flow problem in polynomial time.

Recently, Ikura and Nemhauser (1983) have independently developed a similar parametric procedure for the transshipment problem.

2. Preliminaries: Strongly Feasible Bases

As before, we assime that A is the full row rank constraint matrix of the minimum cost network flow problem (1). We assume without loss of generality that no two columns of A are the same. We associate with matrix A the directed graph $G = (V, E)$ where $V = \{0, \ldots, m\}$, and the edge set E is constructed as follows.

(2.1) If a column of A has a 1 in component i and a -1 in component j, then we associate an edge (i, j) of E.

(2.2) If a column of A has a 1 [resp, -1] in component i and a 0 in the remaining components then we associate an edge $(i, 0)$ [resp., $(0, i)$] of E.

As is well known, each basis B of A induces a subgraph G_B of G that is a tree. We assume henceforth that the tree is rooted at vertex 0.

For each tree T of G, we say that the edge e = (i, j) of T is a downward edge of T if (i, j) is on the unique simple path from to j. Otherwise, we say that e is an upward edge of T. Suppose that i and i' are two vertices of T and that e and e' are two edges of T. We let $P_T(i, i')$ be the simple path in T whose initial vertex is i and whose terminal vertex is i'. Similarly $P_T(e, e')$ is the simple path in T whose initial edge is e and whose terminal edge is e'. We define $P_T(i, e)$ and $P_T(e, i)$ analogously.

We say that e is an ancestor of e' in T if $e \in P_T(0, e')$. We also say that e' is a descendent of e. We say that e is the father of e' (and that e' is the son of e) if e is the second to last edge on $P_T(0, e')$. For each vertex $v \in V$ and for each edge e_i of the basic spanning tree T, we associate the colors red, green and yellow as follows.

(3.1) If $x_i > 0$, then we color edge e_i "green".

(3.2) If $x_i = 0$ and if e_i is an upward edge of T then e_i is colored "red".

(3.3) If $x_i = 0$ and if e_i is a downward edge of T then e_i is colored "yellow".

(3.4) For each vertex $v \in V$, if $P_T(0, v)$ has at least one red edge then v is colored "red".

(3.5) For each vertex $v \in V$, $P_T(0, v)$ solely of green and yellow edges then v is colored "green".

We say that a red edge e = (i, j) in T is maximum in T if i is red and j is green. Thus the ancestors of e in T are green.

As per Cunningham (1976), we say that a basic feasible flow is strongly

feasible if no edge of the corresponding tree is red. (Orlin (1983) showed that a basic feasible flow is strongly feasible if and only if it is lexico-positive.)

In general, the dual simplex pivot rule that we adopt will pivot out red edges. For each red (and hence upward) edge $e = (i, j)$, let T_e be the vertices of V that are in the same connected component of $T - e$ as vertex 0. Let $F(T, e)$ be the fundamental cutset of T induced by e and defined as follows.

$$F(T, e) = \{e' = (i', j') \in G - T : i' \in T_e \text{ and } j' \in T - T_e\}.$$

We illustrate these terms in Figure 1 and in Tables 1 and 2. In Figure 1, the only maximal red edge is $e = (4, 2)$. The fundamental cutset induced by e is $F(T, e) = \{(1, 6)\}$

Edge	Tree Edge	Reduced Cost	Flow	Color
(0, 1)	Yes	.	3	green
(0, 2)	Yes	.	0	yellow
(1, 6)	No	5	0	.
(2, 1)	No	2	0	.
(2, 3)	Yes	.	4	green
(4, 2)	Yes	.	0	red
(4, 5)	Yes	.	2	green
(4, 6)	Yes	.	6	green
(5, 3)	No	6	0	.

Table 1. The data for the edges of the graph of Figure 1.

Vertex	Supply/Demand	Color
1	-3	green
2	4	green
3	-4	green
4	8	red
5	-2	red
6	-6	red

Table 2. The data for the vertices of the graph in Figure 1.

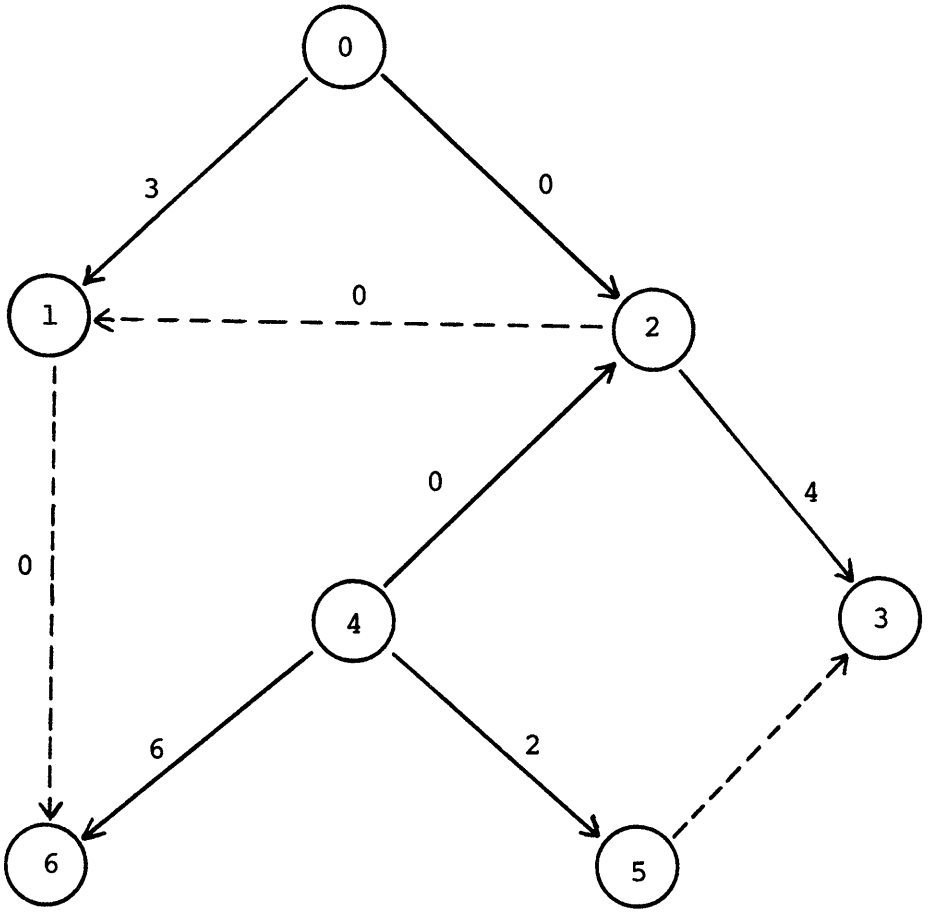

Figure 1. A basic feasible solution that is not strongly feasible. Edge (4,2) is red. The edge numbers refer to the edge flows.

3. Dual Pivoting to Reach Strong Feasibility

In Section 4 we will develop a parametric procedure in which the parameter decreases whenever strong feasibility is obtained. Our parametric algorithm relies on the efficient procedure developed below for moving from feasibility to strong feasibility using dual pivots.

The dual simplex pivoting procedure is well-known; see for example Dantzig (1963). Here we describe it using the terminology developed in the previous section. Moreover, we specialize it for our purposes.

Dual Simplex Pivoting

(4.0). Start with an optimal basic solution $x = (x_1)$ that is not strongly feasible. Let $\bar{c}_k$ denote the reduced cost of variable $\bar{x}_k$.

(4.1). Select a maximum red edge e in the basis tree T. If no such edge e exists, then quit because T is strongly feasible. Otherwise, continue.

(4.2). Select an edge e' in $F(T, e)$ whose reduced cost is minimum. If $F(T, e) = \emptyset$, then quit as there is no strongly feasible basis. Otherwise, let the new basic tree consist of $T + e' - e$. Return to (4.1).

REMARK. A special case of the dual simplex pivot rule (4) is the rule of pivoting out the lexicographically most infeasible edge. See Orlin (1983) for more details.

We illustrate the dual simplex pivot rule in Figure 2. The variable pivoted out corresponds to edge (2, 3), the only red edge. The variable pivoted in corresponds to the edge (1, 6). The spanning tree obtained subsequent to the pivot is given in Figure 2.

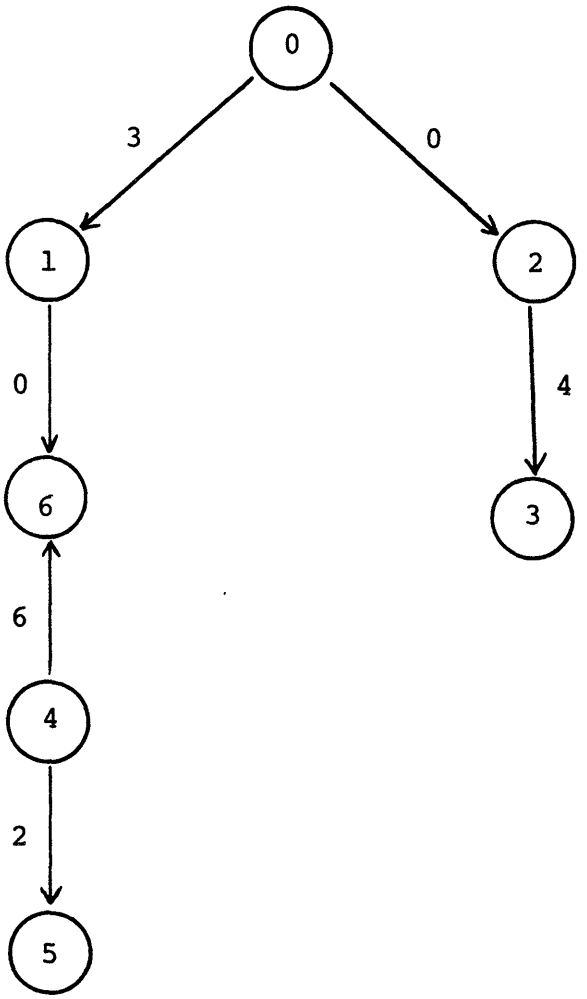

Figure 2. A strongly feasible spanning tree solution obtained from Figure 1 by a dual pivot.

We say that $P_T(i, j)$ is a red-green chain in T if i is an ancestor of j and if $P_T(i, j)$ has no yellow edges. The red-green chains in T induce a partial order S_T as follows. We write that $e \succ e'$ in S_T if edges e and e' are red and if $P_T(e, e')$ is a red-green chain in T. We illustrate these concepts in Figures 3 and 4.

THEOREM 1. Suppose that B is an optimal basis for the network flow problem (1) but that B is not strongly feasible. Let T be the corresponding spanning tree, and let S_T be the partial order on the red edges induced by the red-green chains. Let t be the number of minimal elements in S_T. Then the number of dual pivots using procedure (4) to create a strongly feasible basis (or prove that none exist) is at least t and at most tm.

PROOF. Let $T^0, T^1, \ldots, T^r$ be the sequence of trees obtained by dual pivots, where T^0 is the tree corresponding to the original basis. In addition, we define the following notation.

V^k = set of green vertices in T^k.

E^k = set of maximum red edges in T^k.

S^k = partial order on the red edges induced by T^k.

t^k = number of minimal red edges in S^k.

s^k = dilworth number of S^k, i.e., the minimum number of red-green chains to cover the red edges of T^k.

$P^k(i, j) = P_{Tk}(i, j)$ = the path on T^k from i to j.

To complete the proof, we will prove the following relations, all of which are valid for $k \in [0..r-1]$. Here the notation [j..k] refers to the set of integers $j, j+1, \ldots, k$.

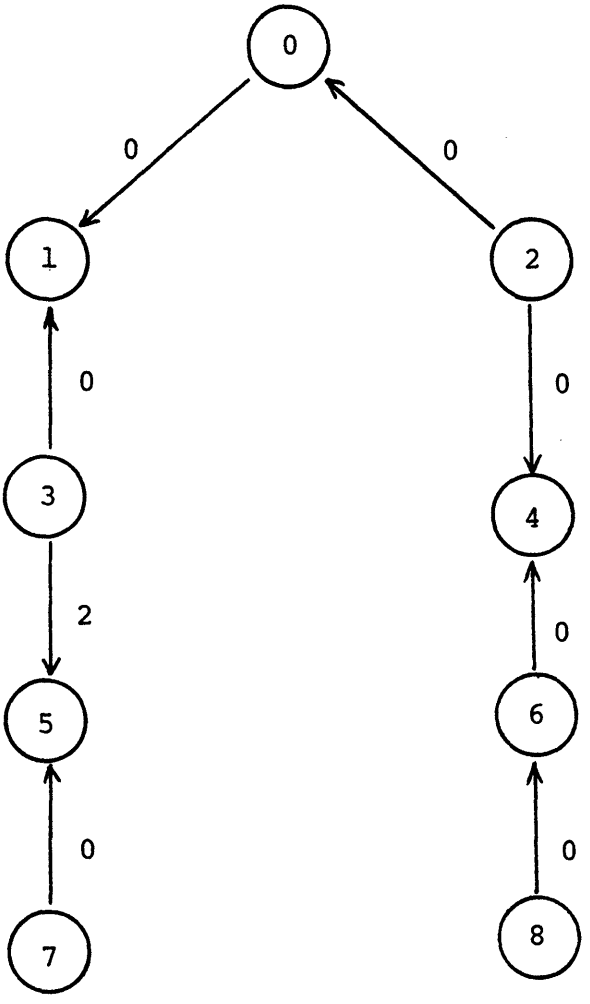

Figure 3. A tree with its corresponding feasible flow.

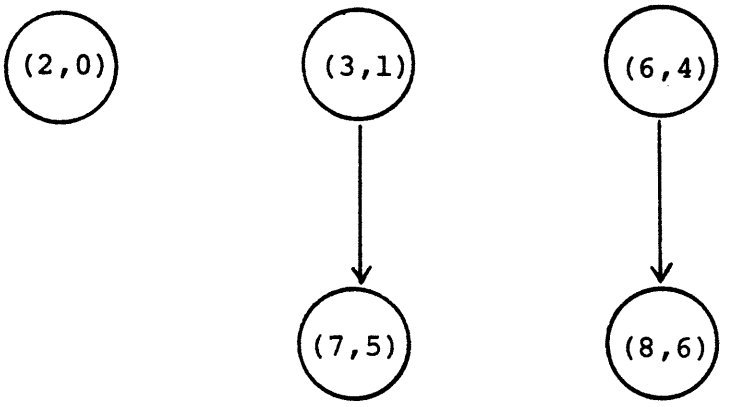

Figure 4. The partial order induced by the red-green chains of the tree of Figure 3. Elements (2,0) and (3,1) are maximum. Elements (2,0), (7,5) and (8,6) are minimal.

$$V^k \subseteq V^{k+1}. \tag{5}$$

$$\text{If } V^{k+1} = V^k \text{ then } |E^{k+1}| < |E^k|. \tag{6}$$

$$|E^k| \leq s^k. \tag{7}$$

$$s^k = t^k. \tag{8}$$

$$t^k - 1 \leq t^{k+1} \leq t^k. \tag{9}$$

We will prove the relations (5) - (9) below. First we will investigate the consequences of these relations.

The fact that the number of pivots is at least t^0 follows from (9) since $t^r = 0$.

From (7), (8) and (9) we see that $|E^k| \leq t^0$ for $k \in [0..r]$. Thus the number of consecutive iterations for which $V^{k+1} = V^k$ is at most $t^0 - 1$ by (6) (If $V^k \neq \emptyset$ then $|E^k| \geq 1$.). Since the number of iterations for which $V^{k+1} \neq V^k$ is at most m by (5), the total number of iterations is at most $t^0 m = tm$.

To prove (5) - (9) we consider tree T^k. Suppose that $e' = (i_2, i_3)$ is the edge that is pivoted in at the k-th pivot, and thus $e' \in T^{k+1} - T^k$. Suppose that edge $e = (i_5, i_4)$ is the edge pivoted out of tree T^k. Now let us consider Figure 5, which represents the circuit created upon adding edge (i_2, i_3) to T^k. In Figure 5, i_0 is the source vertex 0, and the "edges" in Figure 5 represent paths in T^k. Since we will often refer to the paths in Figure 5 we will sometimes drop the superscript k.

We do not assume that the vertices in Figure 5 are distinct. If, for example, $i = i_4$ then $P(i_1, i_4)$ is the trivial path consisting of a single vertex.

The colors of the edges in T^{k+1} may be summarized as follows.

(10.1) Edge $e' = (i_2, i_3)$ is yellow in T^{k+1}.

(10.2) If edge $e^* \notin P(i_5, i_3)$ and $e^* \in T - e$, then e^* is colored the same in T^k and in T^{k+1}.

(10.3) If edge $e^* \in P(i_5, i_3)$ then e^* has different orientations in T^k and T^{k+1}. (Switching orientation changes red edges to yellow and changes yellow edges to red.)

We are now ready to prove (5). Suppose that vertex v is green in T^k. Then $i_5 \notin P^k(0, v)$. Thus $P^k(0, v) = P^{k+1}(0, v)$ and v is green in T^{k+1}. Thus (5) is true.

Suppose now that $V^{k+1} = V^k$. We first claim that i_2 is red in T^k. Otherwise $i_3 \in V^{k+1} - V^k$, contrary to assumption. We will now prove that $E^{k+1} \subseteq E^k$. Since $e \in E^k - E^{k+1}$, this will complete the proof of (6).

Let e^* be a maximum red edge in T^{k+1}. Let $P^* = P^{k+1}(0, e^*)$. Since e^* is maximum, $i_2 \notin P^*$. In fact $P^* \cap P(i_5, i_3) = \emptyset$. Thus $P^* = P^k(0, e^*)$, and e^* is a maximal red edge in T^k. We have thus shown that $E^{k+1} \subseteq E^k$ and thus (6) is true.

Relation (7) is immediate since the set of maximum red edges forms an anti-chain in S^k. (An anti-chain is a set of unrelated elements in S^k).

To see that relation (8) is valid, we first note that $s^k \geq t^k$ since the minimal red edges are an anti-chain in S^k. For each red edge e_i let R_i denote the maximal red-green chain of T^k containing e_i. We claim that each red edge e' of T^k is in some chain R_i, and this will show that $s^k \leq t^k$ and complete the proof. Suppose that e' is not minimal in P^k. Then $e' \succ e_i$ for some red edge e_i in S^k. Then $P^k(e', e_i) \subseteq R_i$, and thus (8) is true.

We now consider relation (9). Let M^k denote the set of minimal red edges of T^k. Our proof first takes into consideration the set $M^k \cap M^{k+1}$. We state the result as Lemma 1.

LEMMA 1. Suppose that $e^* \in T^k \cup T^{k+1}$ and that $e^* \notin P(i_4, i_3)$. Then $e^* \in M^k$ if and only if $e^* \in M^{k+1}$.

PROOF. Suppose that $e^* \in M^k$ and $e^* \notin P(i_4, i_3)$. Since (i_5, i_4) is a maximum red edge in T^k, $e^* \notin P(i_0, i_4)$. We suppose that $e^* \notin M^{k+1}$ and we will derive a contradiction. Let $P^* = P(e^*, e')$ be a red-green chain in T^{k+1} terminating at the red edge e'. Then $(i_2, i_3) \notin P^*$ since (i_2, i_3) is yellow. Therefore, the chain P^* does not contain any edge of $P(i_5, i_3)$ and by (10) it follows that $P^* \in T^k$. This contradicts that $e^* \in M^k$.

Suppose now that $e^* \in M^{k+1}$ and $e^* \notin P(i_3, i_5)$. Suppose further that $e^* \notin M^k$ and that $P^* = P(e^*, e')$ is a red-green chain in T^k that terminates at the red edge e'. Since (i_4, i_5) is maximum, we know that $e^* \notin P(i_0, i_4)$. Thus no edge of P^* is in (i_5, i_3) and thus P^* is also a path in T^{k+1}. Thus contradicts that $e^* \in M^{k+1}$, completing the proof of the Lemma. □

To complete the proof of (9) we will define a 1:1 mapping of M^{k+1} into (but not necessarily onto) M^k. However, we will show that there is at most one element of M^k that is not in the range of the mapping, and this will complete the proof.

Let $f : M^{k+1} \to M^k$ be defined as follows. For $e^* \in M^k \cap M^{k+1}$, we let $f(e^*) = e^*$. By Lemma 1, we are left with edges on the path $P(i_3, i_5)$. For $e^* \in P(i_3, i_5) \cap M^{k+1}$, let $f(e^*)$ be the unique yellow edge e' in $P(i_3, i_5)$ that is closest to e^*. If $P^{k+1}(e^*, i_5)$ has no yellow edges, then let $f(e^*) = (i_5, i_4)$.

We first note that $f(e^*)$ is red in T^k by relation (10). We also note that the mapping is 1:1. Otherwise there are two edges e^* and e' such that $f(e^*) = f(e')$. In such a case, one can show that $P^{k+1}(e^*, e')$ or $P^{k+1}(e', e^*)$ is a red-green chain and thus at most one of e^*, e' is in M^{k+1}.

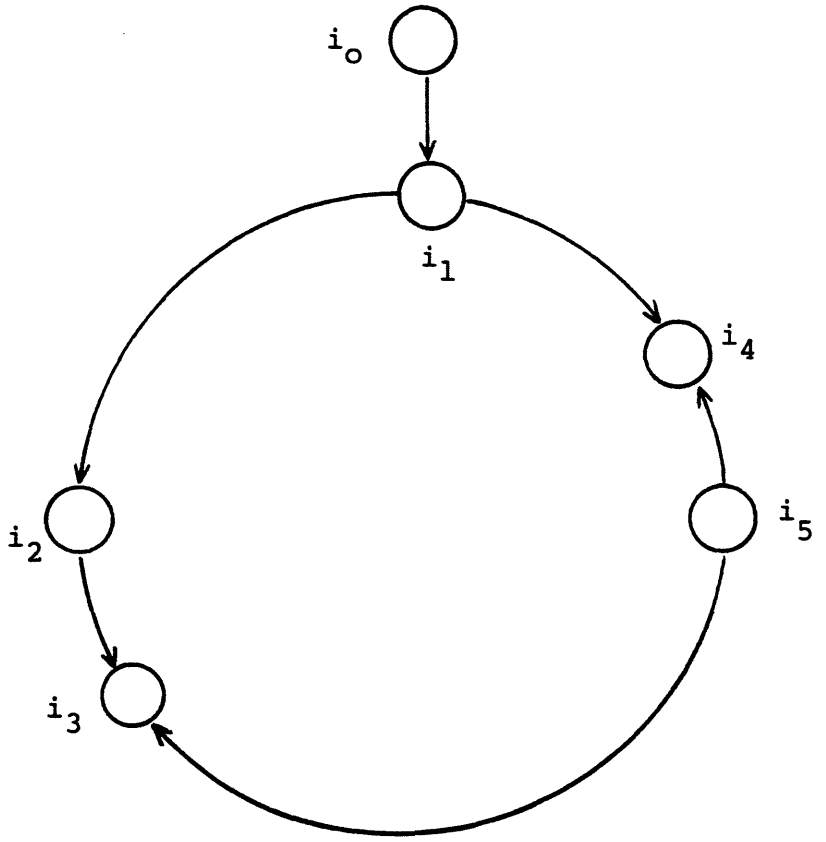

Figure 5. The circuit created by adding edge (i_2, i_3) to T^k.

We now prove via a contradiction that $f(e^*)$ is a minimal red edge in T^k. Suppose that $e' = f(e^*)$ and that $P^* = P^k(e', e'')$ is a red-green chain in T^k terminating at the red edge e''. Let i' be the last vertex of P^* that is also on the path $P(i_5, i_3)$. Thus $P^* = P^k(e', i') \cup P^k(i', e'')$. We claim that $P^{k+1}(e^*, i') \cup P^{k+1}(i', e'')$ is a red-green chain terminating at the red edge e'', and this will complete the contradiction. We note that $e'' \notin P(i_5, i_3)$. (Otherwise, e'' is yellow in T^{k+1}. Moreover, e'' must be on the path $P(e^*, i_5)$ since e^* is yellow in T^k and P^* is a red-green chain. This now contradicts our definition of $f(e^*)$ since e'' is closer to e^* than is e''.) Since $e'' \notin P(i_5, i_3)$, $P^{k+1}(i', e'')$ is a red-green chain in T^{k+1} terminating at the red edge e''. We now claim that $P^{k+1}(e^*, i')$ is a red-green chain. Otherwise, there is a yellow edge $\hat{e}$ on the path. But then $\hat{e}$ is closer to e^* than is e', contradicting our definition of $f(e^*)$. This completes the proof of the relation "$t^{k+1} \leq t^k$".

To complete the proof of the relation "$t^{k+1} \geq t^k - 1$", we show that there is at most one edge in $M^k - f(M^{k+1})$. By Lemma 1, we may restrict attention to the path $P(i_5, i_3)$. Let i^* be the vertex on $P(i_3, i_5)$ such that $P^k(i^*, i_3)$ is a red-green chain and such that i^* is closest to i_5. We claim first that if $e^* \in M^k \cap P(i^*, i_4)$, then $e^* \in f(M^{k+1})$. To see this, we let e' be the yellow edge in $P^k(e^*, i_3)$ that is closest to e^*. A symmetric argument to the one given above shows that $e' \in M^{k+1}$. Finally, there is at most one edge in $P^k(i^*, i^3) \cap M^k$. This completes the proof of Theorem 1. □

THEOREM 2. Suppose that B is a feasible basis that is not strongly feasible. Let t be the number of minimal red edges in the partial order induced by the tree corresponding to basis B. Then the number of pivots

required to reach strong feasibility is at least t. Moreover, if the directed graph is complete and if we are permitted to pivot in any edge, then the minimum number of pivots to reach strong feasibility is exactly t.

PROOF. An examination of the proof in Theorem 1 that "$t^{k+1} \geq t^k - 1$", will show that the proof does not rely on the fact that the edge pivoted out is a maximum red edge. Thus the number of dual pivots required to reach strong feasibility is at least t. To prove that there is a sequence of exactly t pivots to reach strong feasibility, we prove the equivalent result that there is a single pivot that reduces the number of minimal red edges from t to $t - 1$.

Let $P(e^*, e')$ be a red-green chain from a maximum red edge e^* to a minimal red edge e'. Let i and j be the initial and terminal vertices of $P(e^*, e')$. We claim that if we pivot in edge (i, j) and pivot out edge e^*, then the resulting tree T' has at most $t - 1$ minimal red edges.

If we consider Figure 5 once again, we note first that $i_4 = i_1 = i_2 = i$ and that $i_3 = j$. Thus $P(i_4, i_3)$ is a red-green chain in T. Moreover, any minimal red edge in T' does not lie on $P(i_3, i_5)$ and thus is a minimal red edge in T. Since $e' \in T - T'$, it follows that the number of minimal red edges in T' is at most $t - 1$, completing the proof. □

4. The Parametric Network Flow Algorithm

Consider the parametric linear program $L_g(\theta)$

$$\begin{aligned} &\text{Minimize} && cx && L_g(\theta) \\ &\text{Subject to} && Ax = b + g(\theta) \\ & && x \geq 0. \end{aligned}$$

We will soon specify a vector-valued function g for which we can solve $L_g(\theta)$ very efficiently. Our choice of g is based on ideas from Edmonds-Karp scaling technique and on the following Lemma concerning network flows.

LEMMA 2. Suppose that B is a strongly feasible basis for the network flow problem $\min(cx : Ax = b, x \geq 0)$. Then the following are true.

(1) If $b' \leq b$, b is integral and $\sum_{i=1}^{n} (b_i - b'_i) \leq 1$, then B is also feasible for the network flow problem $\min(cx : Ax = b', x \geq 0)$.

(2) If $b' \leq b$ and if B is feasible for the network flow problem $\min(cx : Ax = b', x \geq 0)$, then B is also feasible for the network flow problem $\min(cx : Ax = b^*, x \geq 0)$ for all $b' \leq b^* \leq b$.

PROOF. Statement (2) is proved as part of Theorem 1 of Orlin (1983). We next consider (1). Let T be the spanning tree corresponding to the basis B. By hypothesis, T has no red edges and thus the flow in any upward edge of T is strictly positive. By the unimodularity of B and the integrality of b, the flow in any upward edge of T is at least one.

To obtain the basic flow of b' from the basic feasible flow of b, one sends $b_i - b'_i$ units of flow along the path $P_T(0, i)$. Since $\Sigma(b_i - b'_i : i \in [1..m]) \leq 1$, the resulting flow is non-negative and hence feasible. □

Let $t = \max(\lceil \log(|b_i| + 1) \rceil : 1 \leq i \leq m)$.

Let us now rewrite b_i as

$$b_i = b_{ti}2^t - \sum_{k=0}^{t-1} b_{ki}2^k, \quad \text{for } i \in [1..m] \tag{11}$$

where b_{ki} is 0 or 1 for all i. We now define $g = (g_i)$ as follows. For each triple i, j, ℓ of integers with $i \in [1..m]$, $j \in [0..m-1]$, and $\ell \in [0..t-1]$, let

$$g_i(\ell m + j) = \begin{cases} \sum_{k=0}^{\ell} b_{ki}2^k & \text{if } j \in [i..m] \\ \sum_{k=0}^{\ell-1} b_{ki}2^k & \text{if } j \in [0..i-1]. \end{cases} \tag{12}$$

The Parametric Network Flow Algorithm

(13.0) Obtain an optimal strongly feasible basis B for $L_g(\theta)$ for $\theta = tm$. (We will discuss this step in more detail below).

(13.1) Determine the minimum integral value θ' such that B is feasible for $L_g(\theta')$. If $\theta' \leq 0$, then quit with the optimum basic feasible solution. Otherwise, continue.

(13.2) Perform dual pivots using procedure (4) until either (a) a strongly feasible basis B is obtained or (b) $F(T, e) = \emptyset$ for some T. In case (a), return to step (13.1). In case (b), partition $L_g(\theta)$ into two network flow problems as described by Cunningham (1976).

THEOREM 3. Suppose that the parametric algorithm (13) is used to solve $L_g(\theta)$ with g defined as in (12). Then the total number of pivots is at most mb^*, where b^* is the total number of ones in the binary representation (11) of b.

PROOF. We will henceforth abbreviate $L_g(\theta)$ as $L(\theta)$. Suppose that the basis B is optimal and strongly feasible for $L(\theta^*)$ for the integer θ^* and that θ' is the minimal integral value of θ chosen in (13.1) such that B is feasible for $L(\theta')$. We claim that

(14) $\theta' < \theta^*$.

and (15) The number of dual pivots to create a strongly feasible basis for $L(\theta')$ is at most $m(\theta^* - \theta')$.

Relation (14) is a consequence of (1) of Lemma 2 and our construction of g. (In order to apply Lemma 2 it will be necessary to divide the right hand side coefficients $b - g(\theta)$ by the least common divisor, which is 2^k for a suitable choice of k.)

We now consider (15). Let T be the optimal spanning tree for

$P_g(\theta^*)$. Let $S = \{i \in V : g_i(\theta') < g_i(\theta^*)\}$. By our construction of g in (13), $|S| \leq \theta^* - \theta'$. Moreover, since we are sending flow along each path $P_T(0, i)$ for $i \in S$, it follows that no edge of any of these paths is yellow with respect to the flow for $L(\theta')$. Thus the set $\{P_T(0, i) : i \in S\}$ is a set of $|S|$ red-green chains that covers all of the red-edges of T with respect to $L(\theta')$. (No other edge is red since T is strongly feasible for $L(\theta^*)$.) Thus the number of minimal red edges in T for $L(\theta^*)$ is at most $|S|$. By Theorem 1, the number of pivots to reach strong feasibility is at most $m|S| \leq m(\theta^* - \theta')$. □

Initialization of the Algorithm

In step (13.0), we assume that we start with a strongly feasible basis for the network flow problem $P_g(tm)$. We rewrite this problem as

$$\begin{aligned} &\text{Mimimize} && cx \\ &\text{Subject to} && -Ax = 2^t b' \\ & && x \geq 0 \end{aligned} \qquad (16)$$

where $b'_i = 0$ or -1 according as b_i is non-positive or positive. Then (16) is a shortest path and may be solved via a number of different procedures. If the optimum basis for the shortest path problem is not strongly feasible, then we may obtain a strongly feasible optimum basis with at most m^2 additional pivots, as proved in Theorem 1. (In fact, the simplex algorithm with Dantzig's pivot rule solves the shortest path problem in a number of pivots that is bounded by a polynomial in $|V|$, as proved by Orlin (1983)).

On the Binary Representation of b

The algorithm given above also works if the function $g(\theta)$ is monotonically decreasing rather than increasing. (It does rely on g being monotonic). In this case we could write b_i as $-b_{t_i}2^t + \Sigma(b_{k_i}2^k : k \in [0..t-1])$. In this case, the binary representation

of b_i is the ordinary one if $b_i \geq 0$. The previous representation was the ordinary representation of negative integers b_i.

If g is monotonically decreasing, the algorithm works essentially the same way as before except that the roles of yellow edges and red edges are reversed.

A Speed-up Technique

In (13) we defined the function g in a specific way. However, the computational bound would be valid if we chose any parametrization such that $g(\theta)$ differs from $g(\theta + 1)$ in at most one component and only in the last bit of accuracy. It may be worthwhile in practice to consider a heuristic procedure that determines which component should vary at the value θ. In this way one may be able to decrease the number of pivots.

We also note that the pivoting technique consists of dual pivots. Unfortunately performing dual pivots requires $O(|E|)$ steps per iteration as opposed to the faster methods for performing primal pivots.

Some Concluding Remarks

Although the above procedure was developed primarily for theoretical reasons, we hope that the algorithm will prove to be efficient in practice. As of this time, we have insufficient computational experience with the algorithm to assess its efficacy.

The above parametric network simplex algorithm runs in polynomial time, i.e., time that is polynomially bounded in the size of the data. It is an interesting open question as to whether the number of pivots is bounded by a polynomial in $|V|$. It appears that the "equivalence" arguments of Lemma 3 of this author (1983) are not strong enough to prove such a bound for the parametric algorithm.

REFERENCES

Cunningham, W. (1976). A network simplex method. Mathematical Programming 11, 105-116.

Dantzig (1963). Linear Programming and Extensions. Princeton University Press. Princeton, N.J.

Edmonds, J. and R. Karp (1972). Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM 19, 248-264.

Ikura, Y. and G. Nemhauser (1983). Unpublished manuscript.

Orlin, J. (1983). On the simplex algorithm for networks and generalized networks. Working Paper. Sloan School of Management, MIT.

Zadeh, N. (1973). A bad network flow problem for the simplex method and other minimum cost flow algorithms. Mathematical Programming 5, 255-266.