MANAGEMENT SCIENCE
Vol. 17, No. 9, May, 1971
Printed in U.S.A.

FLOWS IN ARBORESCENCES*†

FRED GLOVER

University of Colorado

This paper gives efficient methods for solving four specially structured network problems that arise in connection with certain integer programming methods developed by Cook and Cooper, Hillier, and Glover. Such problems have also independently been studied in inventory theory by Ignall and Veinott, who have developed extensive qualitative (nonalgorithmic) implications of their structures. In the integer programming context, special subclasses of these network problems are generated and solved as part of a strategy for solving more general integer linear programs. By providing particularly efficient methods for accommodating somewhat broader network structures, our results enable the development of related integer programming solution strategies that generate more complex subproblems.

We show that the first of the four network problems can be solved by a procedure that assigns each variable a value exactly once, without subsequent revision. Properties of optimal solutions for the remaining problems are developed that enable the algorithm for the first to be extended to the second and then, by suitable transformation of variables, to the third and last problems as well. Moreover, while the last three problems involve an additional linear constraint that nullifies the "integer extreme point property", we show that optimal integer solutions to these problems can nevertheless easily be obtained from the optimal continuous solutions.

1. Introduction

Special algorithms are given for four closely related network flow problems that arise in solving integer programming problems by truncated enumeration [4]. The constraints for these problems consist of nested inequalities on partial sums of variables and a single linear inequality over all variables. In a network formulation these constraints can be represented by a capacitated arborescence with the linear inequality restricting the flows from the source nodes.

It is shown that the structure of the first problem yields to a very efficient algorithm in which each variable is assigned a value exactly once, without subsequent revision. Properties of optimal solutions for the remaining problems are developed that enables the algorithm for the first to be extended to the second, and then, by suitable transformations of variables, to be extended also to the third and last problems. In addition, the solution properties are shown to provide an alternative algorithm for the first problem and hence, by extension, for the other problems in turn. Finally, it is shown that optimal integer solutions for the last three problems can be easily obtained from the optimal continuous solutions.

2. Notation and Problem Statement

Let S_k, $k = 1, \cdots, m$, denote a collection of distinct nonempty sets that contain some or all of the indices j, $j = 1, \cdots, n$, and satisfy the *nesting property*:[1]

$$(p \neq q \quad \text{and} \quad S_p \cap S_q \neq \emptyset) \Rightarrow (S_p \subset S_q \quad \text{or} \quad S_q \subset S_p).$$

* Received April 1968; revised March 1970.

† An abbreviated version of the first portion of this paper appears in the *IFORS Proceedings*, June 1969.

[1] We use the symbol $\subseteq$ to denote set inclusion and $\subset$ to denote proper set inclusion.

For convenience we assume $S_m = \{1, 2, \cdots, n\}$ (hence $S_k \subset S_m$ for all $k < m$), and $S_j = \{j\}$ for $j = 1, \cdots, n$ (hence $m > n$). Consistent with this, we also assume $S_p \subset S_q \Rightarrow p < q$. Let $x = (x_1, x_2, \cdots, x_n)^T$ denote a column vector of variables, $A = (a_1, a_2, \cdots, a_n)$ a row vector of constants such that $a_1 \geqq a_2 \geqq \cdots \geqq a_n$, and define $f_k = \sum_{j \in S_k} x_j$. With each set S_k we associate finite constants U_k, L_k and the inequality

$$(1) \qquad U_k \geqq f_k \geqq L_k, \qquad k = 1, \cdots, m,$$

where $L_k \geqq 0$ for all k.

The problems we wish to solve are:

I. Maximize Ax ($\sum_{j=1}^{n} a_j x_j$) subject to (1).

II. Maximize (alternatively Minimize) f_m subject to (1) and

$$(2) \qquad Ax \geqq a_0$$

where a_0 is a given scalar constant.

III. Maximize (alternately Minimize) x_p for any p, $1 \leqq p \leqq n$, subject to (1) and (2).

IV. Maximize (alternately Minimize) f_p for any p, $n < p < m$, subject to (1) and (2).

Three recently proposed truncated enumeration algorithms for integer programming concern themselves with solving problems that are special cases of Problems I–IV as part of a strategy for solving the more general integer programming problem. Thus the methods developed here may be used with these and related algorithms as a means for implementing such a strategy efficiently. The most recent of these algorithms, due to Fred Hillier [7], seeks to impose restrictions on a convex combination of extreme points of a simplex consistent with the nonnegativity and integrality of feasible lattice points. The second algorithm, due to Cook and Cooper [2] (as modified by a proposal in [4]), exploits a restricted form of (1) and (2) to supplement information obtained from the Fourier elimination method. The third algorithm [5], for the 0–1 problem, provides the chief source of motivation for this paper and specifies solutions for Problems I–IV when the inequalities of (1) simplify to $U_m \geqq f_m \geqq L_m$ and $1 \geqq x_j \geqq 0$ for $j = 1, \cdots, n$. The solutions proposed in [5] can be modified to accommodate constraints $U_k \geqq f_k \geqq L_k$ for $n < k < m$ provided the S_k are disjoint. However, to solve more general integer programs by this approach, it is desirable to solve Problems I–IV rapidly when the x_j are not 0–1 variables, and the sets S_k are not only disjoint but assume other structures as well.

We now derive the general results upon which such applications can be based.

3. A Network Representation[2]

The inequalities of (1) can be translated into those of a network flow problem over a capacitated arborescence by means of a *Hasse diagram* (see Berge [1, p. 12]). We create a network whose nodes are the sets S_k, $k = 1, \cdots, m$, an arc going from node p to node q if:

(i) $S_p \subset S_q$, and

(ii) no S_k exists such that $S_p \subset S_k \subset S_q$.

To complete the network we also adjoin a sink node t and an arc from m to t (where, say, $t = m + 1$).

We note that the nesting property defined in §2 immediately implies that each node

[2] I am indebted to R. Van Slyke for suggesting this representation.

k, except for t, has exactly one arc issuing from it,[3] and we will identify each arc by the name of its initial node. The constraints (1) may thus be preresented by attaching the lower bound L_k and upper bound U_k to arc k and requiring a flow to the sink node t that satisfies all arc capacities with unlimited supply available at the source nodes and all other nodes $k \leqq m$ satisfying the usual conservation equations. The source nodes are of course the minimal sets S_j, $j = 1, \cdots, n$, and the value of the flow across each arc k, $k \leqq m$, is identified as the value assigned to f_k (hence to the variables x_j for $j = 1, \cdots, n$).[4]

Problem I then arises by assigning a weight a_j to each unit of flow across arc j for $j \leqq n$ and then seeking a feasible flow that maximizes total weight. Each of the other problems consists of maximizing or minimizing the flow f_p over some arc p, subject to the restriction that the total weight of the solution be at least a_0.

4. An Algorithm for Problem I

For a given arc p, let P_p denote the set consisting of p and all the arcs succeeding p along the unique path from p to t; that is, $P_p = \{k: S_p \subseteq S_k\}$.[5] Let B_p denote the set of arcs that are immediate predecessors of p, that is, $B_p = \{k: S_k \subset S_p \text{ and } \nexists\, S_r \ni S_k \subset S_r \subset S_p\}$. Finally, let $B_{p:k}$ be the set of all immediate predecessors of p except for the one in P_k (i.e., $B_{p:k} = B_p - P_k$); and for $P_p \subset P_k$, let $P_{k:p}$ be the set of all successors of k in P_k except for those that follow p (hence $P_{k:p} = \{h: S_k \subset S_h \subseteq S_p\}$). Note that the indexing assumed earlier for the sets S_k implies that arc k is a predecessor of arc p (equivalently, $P_p \subset P_k$) only if $k < p$, hence $p \in P_k$ implies $k \leqq p$ and $k \in B_p$ implies $k < p$. Also, $P_{k:p} = \{h: h \in P_k \text{ and } k < h \leqq p\}$. These definitions are illustrated in the following diagram, where the nodes are represented by circles with the node index appearing inside, and each arc receives the same index as node to its left.

Some of the relevant arc sets for this diagram are as follows:

$$\begin{array}{rlcrl}
S_j &= \{j\} \text{ for } j = 1, \cdots, 6 & \qquad & P_7 &= \{7, 9, 10\} \\
S_7 &= \{4, 6\} & & B_{10} &= \{3, 8, 9\} \\
S_8 &= \{2, 5\} & & B_{10:7} &= \{3, 8\} \\
S_9 &= \{1, 4, 6\} & & P_{4:9} &= \{7, 9\} \\
S_{10} &= \{1, 2, 3, 4, 5, 6\} & & &
\end{array}$$

The strategy of the algorithm for Problem I is to obtain values L_k^* and U_k^*, the latter representing "true" upper bounds on the flows f_k for $k \leqq n$, and values Δ_k which represent increments to cumulative lower bounds across each arc k for $k > n$. Then the variables x_j are assigned values one at a time, beginning first with x_1 and then, given the assignment to x_1, applying the same rule to determine the value for x_2, and so on.

[3] This, and the fact that there are no cycles, characterizes the network as an arborescence. See Berge [1, p. 160]. For fundamental theory of flows in more general networks, see also Ford and Fulkerson [3].

[4] The proof of these remarks follows by observing their validity for the initial arcs $j \leqq n$, then for all arcs each of whose predecessors has already been validated, and hence eventually for all arcs of the network.

[5] Using this definition, we note that Problem I is no less general than the one in which a weight a'_k is assigned to every arc k, since the two problems become equivalent by letting $a_j = \sum_{k \in P_j} a'_k$ for $j = 1, \cdots, n$. Also we remark in passing that Problem I has the same set of feasible and optimal solutions if the orientation of every arc is reversed. These comments of course also apply to Problems II, III, and IV.

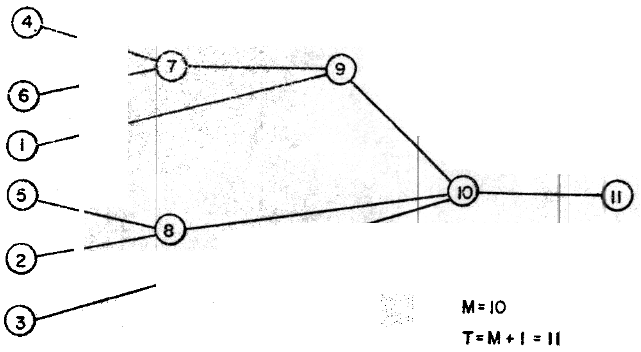

FIGURE 1. Example Arborescence (with arcs directed from left to right)

We determine the incremental and revised upper and lower bounds as follows:

For $p = 1, \cdots, n$, let $L_p^* = L_p$. Then, for $p = n + 1, \cdots, m$, let $L'_p = \sum_{k \in B_p} L_k^*$, $L_p^* = \text{Max}\ (L_p, L'_p)$, and $\Delta_p = L_p^* - L'_p$.

Beginning at the terminal arc m, we also define $U'_m = U_m^* = U_m$. Then working backwards, if U_r^* has been determined, consider any arc $p \in B_r$ and define $U'_p = U_r^* - \sum_{k \in B_r : p} L_k^*$ (equivalently, $U'_p = U_r^* - L'_r + L_p^*$) and $U_p^* = \text{Min}\ \{U_p, U'_p\}$.

Clearly, $U_p^* \geqq f_p \geqq L_p^*$ is a necessary requirement for feasibility. However, it is possible that no feasible flow exists in which f_p attains the bound U_p^* or L_p^* for a given arc p. We will give other bounds that f_p must satisfy in §8 and identify bounds that provide both necessary and sufficient feasibility criteria in §9. Nevertheless, the bounds U_p^*, L_p^* and the numbers Δ_p contain all the information needed to determine an optimal solution to Problem I, if such a solution exists. We make this assertion precise in Theorem 1 and its corollary to follow.

THEOREM 1.[6] *If Problem* I *has an optimal solution, then it has an optimal solution such that*

$$x_1 = U_1^* \qquad \text{if } a_1 > 0$$

and

$$x_1 = \text{Min}\ \{U_1^*, L_1 + \textstyle\sum_{k \in P_1 : m} \Delta_k\} \qquad \text{if } a_1 \leqq 0.$$

COROLLARY 1. *If Problem* I *has a feasible solution, then it has a feasible solution* x^0 *in which, for an arbitrary single index* j, x_j^0 *assumes any value satisfying* $U_j^* \geqq x_j^0 \geqq \text{Min}\ \{U_j^*, L_j + \sum_{k \in P_j : m} \Delta_j\}$.

We observe that Theorem 1 provides an algorithm for Problem I as follows. Let x_1^0 denote the value of x_1 in an optimal solution to Problem I. Setting $x_1 = x_1^0$ yields a new problem with exactly the same form as Problem I, but in which, for all $k \in P_1$, S_k is replaced by $S_k - \{1\}$ and L_k and U_k are respectively replaced by $L_k - x_1^0$ and $U_k - x_1^0$. Thus in the new problem x_2 can be treated as the "new" x_1 and consequently assigned an optimal value by Theorem 1.

In following this procedure it is useful to determine the new values for L_k', L_k^*, Δ_k, U'_k and U_k^* by reference to the old ones, to avoid starting from scratch at each step. We now indicate a set of short cut rules for accomplishing this.[7]

[6] For proofs of the assertions of this paper, see Glover [6].

[7] Computational savings can sometimes (but not always) result by initially replacing each x_j with $x_j - L_j$ to yield new variables with zero lower bounds. In particular, the preliminary computation for such a replacement will be worthwhile when the sets S_k have a nonempty intersection for $k > n$.

If $x_1^0 > L_1 + \sum_{k \in P_{1:m}} \Delta_k$, let $w = t$. Otherwise, let

$$w = \operatorname{Min}\{p : p \in P_1 \quad \text{and} \quad L_1 + \textstyle\sum_{k \in P_{1:p}} \Delta_k \geqq x_1^0\}.$$

(Recall $P_{1:p} = \{k : k \in P_1 \text{ and } 1 \leqq k \leqq p\}$.)

We will denote the new values of the bounds after x_1 is set equal to x_1^0 by $\bar{L}'_k$, $\bar{L}_k^*$, $\bar{\Delta}_k$, etc.

Evidently, if $p \notin P_1$, then $\bar{L}'_p = L'_p$, $\bar{L}_p^* = L_p^*$, and $\bar{\Delta}_p = \Delta_p$. On the other hand, if $p \in P_1$, then it may readily be verified that

$$\begin{aligned}
\bar{L}_p^* &= L_p^* - (L_1 + \textstyle\sum_{k \in P_{1:p}} \Delta_k) && \text{if } p < w,\\
&= L_p^* - x_1^0 && \text{if } p \geqq w.\\
\bar{\Delta}_p &= 0 && \text{if } p < w,\\
&= L_1 + \textstyle\sum_{k \in P_{1:w}} \Delta_k - x_1^0 && \text{if } p = w,\\
&\quad \Delta_p && \text{if } p > w.
\end{aligned}$$

Then of course

$$\bar{L}'_p = \bar{L}_p^* - \bar{\Delta}_p \qquad \text{for } p \in P_1.$$

The new values for U'_p and U_p^* when $p \in P_1$ are given simply by

$$\bar{U}'_p = U'_p - x_1^0 \quad \text{and} \quad \bar{U}_p^* = U_p^* - x_1^0.$$

The values of $\bar{U}'_p$ and $\bar{U}_p^*$ are not so easily found when $p \notin P_1$, but we note that it is possible originally to compute U'_p and U_p^* only for $p \in P_1$, without bothering to compute them for $p \notin P_1$. Thus, $\bar{U}'_p$ and $\bar{U}_p^*$ can be computed for $p \in P_2$ (when x_2 becomes the "new" x_1) by the original rule for computing U'_p and U_p^*.

5. Properties of Optimal Solutions to Problems II, III, and IV

To develop methods for solving Problems II, III, and IV, we will first state certain relationships between optimal solutions to these problems and optimal solutions to Problem I.

For the material to follow we define Problem I^1 to be the same as Problem I except for the values of the bounds L_k and U_k (denoted by L_k^1 and U_k^1 for Problem I^1).

THEOREM 2.[8] *Assume that $L_p^1 \geqq L_p$, $U_p^1 \geqq U_p$, and for all $k \neq p$, $L_k^1 = L_k$ and $U_k^1 = U_k$. Assume also that Problems* I *and* I^1 *have feasible solutions. Then if x^0 is an optimal solution to Problem* I, *there is an optimal solution x^1 to Problem* I^1 *such that* (i) $x_j^1 \geqq x_j^0$ *for all $j \in S_p$,* (ii) $x_j^1 \leqq x_j^0$ *for all $j \notin S_p$, and* (iii) $f_k^1 \geqq f_k^0$ *for all $k \in P_p$. Likewise, if x^1 is an optimal solution to Problem* I^1, *then there is an optimal solution x^0 to Problem* I *such that* (i), (ii), *and* (iii) *are true.*

The following two corollaries to Theorem 2 make it possible to obtain information from an optimal solution to Problem IV for one value of p to be used in solving Problem IV for another value of p.

COROLLARY 2.1. *Let Problems* I *and* I^1 *be the same except that $L_p^1 \geqq L_p$ and $L_q^1 \leqq L_q$, where $S_p \subset S_q$. Let L_q assume the largest value such that Problem* I *has a feasible solution satisfying* (2) *and let L_p^1 assume the largest value such that Problem* I^1 *has a feasible solu-*

[8] Arthur F. Veinott, Jr. and Edward Ignall have independently obtained Theorem 2 in their very interesting paper [8] when Ax is replaced by a strictly convex function.

tion satisfying (2). *If x^0 is an optimal solution to Problem* I, *then there is an optimal solution x^1 for Problem* I^1 *such that $x_j^1 \leqq x_j^0$ for $j \in S_q - S_p$; $x_j^1 \geqq x_j^0$ for $j \notin S_q - S_p$; and $f_k^1 \leqq f_k^0$ for all $k \in P_q$. Likewise if x^1 is an optimal solution to Problem* I^1, *then there is an optimal solution x^0 for Problem* I *such that x^0 and x^1 satisfy the foregoing properties.*

COROLLARY 2.2. *Let Problems* I *and* I^1 *be the same except that $U_p^1 \leqq U_p$ and $U_q^1 \geqq U_q$, where $S_p \subset S_q$. Let U_q assume the smallest value such that Problem* I *has a feasible solution satisfying* (2) *and U_p^1 assume the smallest value such that Problem* I^1 *has a feasible solution satisfying* (2). *If x^0 is an optimal solution to Problem* I, *then there is an optimal solution x^1 to Problem* I^1 *such that $x_j^1 \geqq x_j^0$ for $j \in S_q - S_p$, $x_j^1 \leqq x_j^0$ for $j \notin S_q - S_p$, and $f_k^1 \geqq f_k^0$ for $k \in P_q$. Likewise, if x^1 is an optimal solution to Problem* I^1, *then there is an optimal solution x^0 to Problem* I *such that x^0 and x^1 satisfy the foregoing properties.*

Our next two theorems, together with Theorem 2, give the main results of this section that make it possible to develop efficient methods for Problems II, III and IV.

THEOREM 3.1. *Let Problem* V *be to maximize Ax subject to* (1), (2) *and $f_p = f_p^*$, where f_p^* is the largest value of f_p for which a feasible solution to Problem* V *exists. Let the indices $j \in S_p$ be denoted $1', 2', \cdots, \theta'$, in ascending order, where θ is the cardinality of S_p. Define the sequence of Problems $\mathrm{I}^{h'}$, $h = 0, 1, \cdots, \theta$, so that Problem $\mathrm{I}^{0'}$ is the same as Problem* I *except that $L_j^{0'} = x_j^0$ for $j \in S_p$ (where x^0 is optimal for Problem* I) *and in general Problem $\mathrm{I}^{h'}$ is the same as Problem $\mathrm{I}^{(h-1)'}$ except that $L_{h'}^{h'}$ assumes the largest value for which there is a feasible solution to Problem $\mathrm{I}^{(h-1)'}$ satisfying* (2). *Then there is a set $\{x^{h'}\}$ of optimal solutions to the Problems $\mathrm{I}^{h'}$ such that*: (i) *$x_j^{h'} \leqq x_j^{(h-1)'}$ for all $j \notin S_p$ and* (ii) *$x_j^{h'} = x_j^{(h-1)'}$ for all $j \in S_p, j \neq h'$ (in particular, $x_{j'}^{h'} = x_{j'}^{j'} = L_{j'}^{j'}$ for $1 \leqq j \leqq h$ and $x_{j'}^{h'} = x_{j'}^0$ for $h < j \leqq \theta$). Moreover, $x^{\theta'}$ is an optimal solution to Problem* V.

THEOREM 3.2. *Let Problem* VI *be to maximize Ax subject to* (1), (2) *and $f_p = f_p^*$, where f_p^* is the smallest value of f_p for which a feasible solution to Problem* VI *exists. Let the indices $j \in S_p$ be given as in Theorem* 3.1 *and define the Problems $\mathrm{I}^{h'}$ so that Problem $\mathrm{I}^{0'}$ is the same as Problem* I *except that $U_p^{0'} = x_j^0$ for $j \in S_p$ (where x^0 is optimal for Problem* I) *and, in general, Problem $\mathrm{I}^{h'}$ is the same as Problem $\mathrm{I}^{(h-1)'}$ except that, for $k = \theta - h + 1$, $U_{k'}^{h'}$ assumes the smallest value for which there is a feasible solution to Problem $\mathrm{I}^{(h-1)'}$ satisfying* (2). *Then there is a set $\{x^{h'}\}$ of optimal solutions to the Problems $\mathrm{I}^{h'}$ such that* (i) *$x_j^{h'} \geqq x_j^{(h-1)'}$ for all $j \in S_p$; and* (ii) *$x_j^{h'} = x_j^{(h-1)'}$ for all $j \in S_p, j \neq h'$. Moreover, $x^{\theta'}$ is an optimal solution to Problem* VI.

Theorem 4 provides a computational shortcut in solving Problems III and IV.

THEOREM 4. *Let Problems* I *and* I^1 *and the solutions x^0 and x^1 be given as in Theorem* 2. *Then, if $x_r^1 > x_r^0$ for $r \in S_p$ it follows that $x_j^1 = x_j^0$ for all $j \notin S_p$ such that $a_j > a_r$.*

We now show how to exploit these results algorithmically.

6. Algorithms for Problems II, III, and IV

6.1 *An Algorithm for Problem* II.

The Algorithm we give for Problem II is also fundamental to the algorithms for Problems III and IV. Moreover, the procedural ideas developed for all of these problems will be combined in a highly efficient composite algorithm after the requisite foundations have been laid below. We first consider the case in which the objective is to maximize f_m subject to (1) and (2). Let x^0 denote an optimal solution to Problem I (determined, for example, by the method of §4). Then an optimal solution x^1 to Problem II is obtained as follows:

To maximize f_m :

1. Let $x_j^1 = x_j^0$ for all j such that $a_j > 0$.

2. Suppose that x_j^1 has been determined for all $j < v$ (where $a_v \leqq 0$). Let x^2 denote the current solution (feasible for Problem I) defined by $x_j^2 = x_j^1$ for $j < v$ and $x_j^2 = x_j^0$ for $j \geqq v$.

3. Define $\beta_v = \operatorname{Min}_{k \in P_v} \{U_k - f_k^2\}$ and $\gamma_v = (a_0 - \sum_{j \neq v} a_j x_j^2)/a_v$ ($\gamma_v = \infty$ if $a_v = 0$).

4. Let $x_v^1 = \operatorname{Min} \{\gamma_v, x_v^0 + \beta_v\}$. If $x_v^1 = \gamma_v$, also let $x_j^1 = x_j^0$ for all $j > v$. Otherwise, return to instruction 2 with v at its next larger value (until $v = m$).[9]

In reality, the solution x^1 given by the foregoing algorithm not only maximizes f_m subject to (1) and (2) but also maximizes Ax subject to (1) and $f_m = f_m^1$; i.e., x^1 is an optimal solution to Problem V of Theorem 3.1 for $p = m$. To verify this using Theorem 3.1, we need only show that the value specified for x_v^1 in instructions 1 and 4 is the largest possible value satisfying (1), (2), and $x_j = x_j^2$ for $j \neq v$. This is obviously true for instruction 4 by the definitions of γ_v and β_v. To see that it is also true for instruction 1, let h be the least j for which there exists a solution x^2 feasible for Problem I such that $x_j^2 > x_j^0$ and $x^2 \geqq x^0$. Then, the solution x^3 defined by $x_h^3 = x_h^2$ and $x_j^3 = x_j^0$ for $j \neq h$ is feasible for Problem I and $Ax^0 \geqq Ax^3$ (from the optimality of x^0) implies $a_h \leqq 0$. Consequently, the largest possible value for x_v^1 for all $a_v > 0$ is just x_v^0.

The algorithm for Problem II when the objective is to minimize f_m subject to (1) and (2) corresponds to the preceding algorithm in the same way that Theorem 3.2 corresponds to Theorem 3.1. Thus, we first seek the smallest value of x_n subject to (1), (2), and $x_j = x_j^0$ for $j < n$; then seek the smallest value of x_{n-1} subject to (1), (2), $x_n = x_n^1$ and $x_j = x_j^0$ for $j < n - 1$, and so on. The precise form of the algorithm follows :

To minimize f_m :

1. Let $x_j^1 = x_j^0$ for all j such that $a_j < 0$.

2. Suppose that x_j^1 has been determined for all $j > v$ (where $a_v \geqq 0$). Let x^2 denote the current solution (feasible for Problem I) defined by $x_j^2 = x_j^1$ for $j > v$ and $x_j^2 = x_j^0$ for $j \leqq v$.

3. Define $\beta_v' = \operatorname{Min}_{k \in P_v} \{ f_k^2 - L_k\}$, and

$$\gamma_v' = (a_0 - \textstyle\sum_{j \neq v} a_j x_j^2)/a_v \qquad (\gamma_v' = -\infty \text{ if } a_v = 0).$$

4. Let $x_v^1 = \operatorname{Max} \{\gamma_v', x_v^0 - \beta_v'\}$. If $x_v' = \gamma_v'$, also let $x_j^1 = x_j^0$ for all $j < v$. Otherwise, return to instruction 2 with v at its next smaller value (until $v = 1$).[10]

The justification of this algorithm is analogous to the justification of the algorithm for maximizing f_m, noting that the solution x^1 in this case solves Problem VI of Theorem 3.2 for $p = m$.

We observe that the algorithms we have given for Problem II are even more efficient than the algorithm for Problem I, provided an optimal solution to Problem I has already been obtained.

Also, when minimizing f_m, if x^0 has been obtained by the method of §4, then instruc-

[9] An apparent minor change that can improve the efficiency of this method is to define $u = \operatorname{Max} (h{:}h \in P_v \text{ and } U_h - f_h^2 = \beta_v)$ at instruction 3. Then set $x_j = x_j^0$ for all $j \in S_u$, $j > v$, at instruction 4 and exclude these j as candidates for v at instruction 2.

[10] To shortcut computation, define $u = \operatorname{Max} (h{:}h \in P_v \text{ and } f_h^2 - L_h = \beta_v^1)$ at instruction 3. Then set $x_j^1 = x_j^0$ for all $j \in S_u$, $j < v$, at instruction 4, and exclude these j as candidates for v at instruction 2.

tion 1 may be changed to specify $x_j^1 = x_j^0$ for all j such that $a_j \leqq 0$. The reason for this is that the rule prescribed by Theorem 1 treats $a_j = 0$ as though $a_j < 0$. It may also be noted that the procedure of §4 treats the a_j as though $a_1 > a_2 > \cdots > a_n$. In particular, the use of the indexing of the a_j in this procedure (and also in the procedures of this section) imposes a ranking that corresponds to replacing a_j by $a_j - j\epsilon$, where $\epsilon > 0$ is sufficiently small that $a_j > 0$ implies $a_j - j\epsilon > 0$. For Problems II–IV ϵ must also be small enough that $Ax^0 > a_0 + \epsilon \sum jx_j^0$ for all x^0 satisfying (1). This implicit perturbation clearly admits only one optimal solution to Problem I and extends the applicability of Theorem 4.

6.2 *An Algorithm for Problem* III.

We first consider the problem of maximizing x_p subject to (1) and (2). As before, we actually solve Problem V of Theorem 3.1 (in this case for $f_p = x_p$).

Define a new vector z so that $z_j = x^0 - x_j$ for $j \neq p$ and $z_p = f_m - f_m^0 = \sum_{j=1}^{n} (x_j - x_j^0)$, where x^0 is optimal for Problem I. Also define $T_k = S_k$ if $p \notin S_k$ and $T_k = (S_m - S_k) \cup \{p\}$ if $p \in S_k$. Finally, let $g_k = \sum_{j \in T_k} z_j$.

By reference to these definitions, (1) can alternately be stated $f_k^0 - L_k \geqq g_k \geqq f_k^0 - U_k$ if $p \notin T_k$ and $U_k - f_k^0 \geqq g_k \geqq L_k - f_k^0$ if $p \in T_k$. Applying Theorem 2, there is an optimal solution to Problem III (and, more particularly, Problem V) such that $x_p \geqq x_p^0$, $x_j \leqq x_j^0$ for $j \neq p$, and $f_m \geqq f_m^0$. From the definition of the z_j, this yields $z_j \geqq 0$ for all j and hence $g_k \geqq 0$ for all k. Thus, letting $M_k = f_j^0 - L_k$ if $p \notin T_k$ and $M_k = U_k - f_k^0$ if $p \in T_k$, Problem III can be restated:

Maximize $g_p = \sum_{j=1}^{n} z_j + x_p^0$ subject to

$$M_k \geqq g_k \geqq 0, \qquad k = 1, \cdots, m, \tag{3}$$

and

$$\textstyle\sum_{j=1}^{n} a_j' z_j \geqq a_0 - Ax^0 \tag{4}$$

where

$$a_p' = a_p \quad \text{and} \quad a_j' = a_p - a_j \qquad \text{for } j \neq p.$$

Correspondingly, the associated restatement of Problem I is to maximize $A'z$ subject to (3), (4), and $g_p = g_p^*$, where g_p^* is optimal value for g_p in Problem III. This latter problem has exactly the form of Problem V when $p = m$, since the fact that the S_k satisfy the nesting property implies that the T_k satisfy this property, also. Consequently, to maximize x_p subject to (1) and (2) (i.e., solve Problem V for $x_p = f_p$), it suffices to solve the above problem with the method given for solving Problem II (solving Problem V for $p = m$) in §6.1. However, an optimal solution is first required for the corresponding Problem I (i.e., maximize $A'z$ subject to (3)). For the special form of this problem, the following algorithm may be used.

Algorithm for Problem I *when* $L_k = 0$ *for all* k

1. Let $x_j^0 = 0$ for all j such that $a_j \leqq 0$.
2. Suppose x_j^0 has been determined for all $j < v$ and $a_v > 0$. Define $x_j^2 - x_j^0$ for $j < v$ and $x_j^2 = 0$ for $j \geqq v$.
3. Let $x_v^0 = \text{Min}_{k \in r_v} \{U_k - f_k^2\}$. Repeat instruction 2 for the next larger value of v unless none remain such that $a_v > 0$.

This algorithm clearly applies to the form of Problem I associated with maximizing g_p above. Its validity is established by reference to Theorem 1, since $L_k = 0$ for all k implies $\Delta_k = 0$ for all $k > n$ and hence $U_j^* = \text{Min}_{k \in P_j} (U_k)$.

Given the assignment $x_j = x_j^0$ for $j < v$, the current value of U_v^* is just the value assigned x_v^0 by instruction 3, which is optimal by Theorem 1 for $a_v > 0$.[11] The fact that $L_v + \sum_{k \in P_{v:m}} \Delta_k = 0$ justifies the $x_j^0 = 0$ for $a_j \leqq 0$ in instruction 1.

We note that solving Problems I and II when the above algorithm can be used for Problem I corresponds computationally to solving only Problem II when $a_j \leqq 0$ for all j, since the above algorithm does no work to assign values to x_j^0 for $a_j \leqq 0$ and the algorithm for Problem II does not change the values of the x_j^0 for $a_j > 0$.

Another shortcut in computation is afforded by Theorem 4, which implies that in an optimal solution to Problem III (Problem V for $x_p \neq f_p$) we may immediately set $x_j = x_j^0$ (hence $z_j = 0$) for all j such that $a_j < a_p$. Also, if $a_p > 0$, then it is obvious that one may similarly set $z_p = 0$.

Now, we turn to solving Problem III when the objective is to minimize x_p subject to (1) and (2). For this case, define $z_j = x_j - x_j^0$ for $j \neq p$ and $z_p = f_m^0 - f_m$. Then by Theorem 2 there is an optimal solution x to Problem III (Problem VI for $f_p = x_p$) that implies $z_j \geqq 0$ for all j. Given T_k and g_k as above, (1) can be restated

$$U_k - f_k^0 \geqq g_k \geqq 0 \qquad \text{for all } k \text{ such that } p \notin T_k,$$

$$f_k^0 - L_k \geqq g_k \geqq 0 \qquad \text{for all } k \text{ such that } p \in T_k.$$

Also, (2) becomes

$$\sum a_j' z_j \geqq a_0 - Ax^0$$

where $a_p' = -a_p$ and $a_j' = a_j - a_p$ for $j \neq p$. Finally, the objective Minimize x_p becomes

$$\text{Minimize} \quad -\sum_{j=1}^{n} z_j + x_p^0,$$

which is equivalent to the objective

$$\text{Maximize} \quad \sum_{j=1}^{n} z_j - x_p^0.$$

Thus, when minimizing x_p, Problem III reduces to the same form of Problem II encountered when maximizing x_p (although with different constants) and can be solved in accordance with our previous remarks. It should be noted that Theorem 4 applies in this case to prescribe $x_j = x_j^0$ (hence $z_j = 0$) for all j such that $a_j > a_p$. Also, if $a_p < 0$ then one may permissibly set $z_p = 0$.

6.3 *An Algorithm for Problem* IV.

The algorithm for Problem III stipulated in §6.2 immediately gives an algorithm for Problem IV via Theorems 3.1 and 3.2.

To maximize f_p, first solve Problem $\mathrm{I}^{1'}$ (in the notation of Theorem 3.1) by the procedure of §6.2 (maximizing $x^{1'}$ subject to the stated conditions). Since x^0 is optimal for Problem $\mathrm{I}^{0'}$ and Problem $\mathrm{I}^{1'}$ corresponds to Problem $\mathrm{I}^{0'}$ as Problem III does to Problem I, the starting solution x^0 for the procedure of §6.2 is already given.

Next, we solve Problem $\mathrm{I}^{2'}$ by the procedure of §6.2, where in this case the starting solution "x^0" is the solution to Problem $\mathrm{I}^{1'}$. Solving each Problem $\mathrm{I}^{h'}$ in turn by this form of successive postoptimization eventually yields the solution $x^{\theta'}$ to Problem $\mathrm{I}^{\theta'}$, which is also optimal for Problem IV (Problem V for $n < p < m$). The fact that Problem $\mathrm{I}^{h'}$ is more restricted than Problem $\mathrm{I}^{(h-1)'}$ and $x^{h'}$ is an extension of $x^{(h-1)'}$ suggests

[11] One can alternately compute the values of the U_j^* as in §4, noting in this case that $k \in B_h$ implies $U_k^* = \text{Min}\ \{U_k, U_h^*\}$. This remark also applies to the determination of β_v in the algorithm for Problem II, which corresponds to a current U_v^* for the incremental value of x_v.

that the procedure should be relatively efficient. We note also that, if the value for $x_{h'}^{h'}$ in Problem $I^{h'}$ is restrained from being larger by (2), but not (1), then $x^{h'}$ is itself optimal for Problem IV and it is unnecessary to solve Problems $I^{k'}$ for $k > h$.

Corollaries 2.1 and 2.2 can also be used in this context to provide tighter bounds for the x_j and Theorem 4 can be used to fix some of the x_j at constant values, thereby speeding the computation.

The procedure for minimizing f_p is precisely analogous to that of maximizing f_p, in this case solving the problems $I^{h'}$ of Theorem 3.2.

6.4 *A Composite Algorithm for Problems* II, III *and* IV.

Using the results of the proceding sections, we will give an algorithm that can be applied to solve Problems II, III or IV without defining new variables z_j by the rules of §6.2 (although these definitions are implicitly relied upon).

The composite algorithm is sufficiently detailed to be somewhat difficult to follow conceptually at first examination and for restricted applications the algorithms of the preceding sections (which are easy to program for the computer) are preferable. However, the composite algorithm not only has the advantage of applying to a greater range of problems than the previous algorithms but also can be organized to rival or surpass these methods in efficiency (see §10).

We will assume that x^0 is the (unique) optimal solution to Problem I obtained relative to the perturbation discussed at the end of §6.1 and we will continue to implicitly defer to this perturbation by using the indexing of the a_j to determine rank.

Composite algorithm for maximizing f_p.

1. To begin, let $f_k^1 = f_k^0$ for all k and let $S = S_p$ and $T = S_m - S_p$.
(Optional): Contract S and T by the definitions[12]

$$S = S - \{j:j \in S_k \text{ for some } k \text{ such that } S_k \subseteq S_p \text{ and } f_k^0 = U_k\}$$

$$T = T - \{j:j \in S_k \text{ for some } k \text{ such that } S_k \not\subseteq S_p,\ S_p \not\subseteq S_k,\ \text{and } f_k^0 = L_k\}$$

If $U_k > f_k^0$ for all $k \in P_p$, let $v = t$ $(= m + 1)$. Otherwise, let

$$v = \text{Min}\,\{k:k \in P_p \text{ and } f_k^0 = U_k\}$$

and let $T = T \cap S_v$.

2. If $S = \varnothing$, x^1 is optimal and the process stops. Otherwise, let $r = \text{Min}\,\{j:j \in S\}$.
3. If $T = \varnothing$ and $v \neq t$, stop (x^1 is optimal). Otherwise, let

$$s = \text{Max}\,\{j:j \in T \text{ and } j < r\}.$$

If s does not exist and $a_r > 0$, let $S = S - \{j:a_j > 0 \text{ and } j < \text{Min}\,\{h:h \in T\}\}$, and return to 2.

4. If $v = t$ (hence $U_k > f_k^1$ for all $k \in P_p$), if $a_r \leqq 0$ and either $a_s > 0$ or s does not exist let

$$\delta_r = \text{Min}_{k \in P_p}\,\{U_k \quad f_k^1\},$$

$$\gamma_r = \infty \quad \text{if} \quad a_r = 0, \quad \text{and}$$

$$\gamma_r = (a_0 - Ax^1)/a_r \qquad \text{if } a_r < 0.$$

[12] In this section, we will use instructions of the form $X = X - Y$ to mean that the new X is equal to the old $X - Y$.

For notational convenience, to reduce the number of prescriptions in 7 and 8 below, let $s = 0$ and define $e_0 = 0$, $\delta_0 = \infty$ and $P_0 = \varnothing$. Then go to instruction 7.

5. If s does not exist and instruction 4 is inapplicable, let $S = S - \{r\}$ and return to 2.

6. If instructions 4 and 5 are not applicable (hence s exists), let

$$\delta_r = \text{Min}_{k \in P_r - P_s} \{U_k - f_k^1\},$$

$$\delta_s = \text{Min}_{k \in P_s - P_r} \{f_k^1 - L_k\},$$

$$\gamma_r = (a_0 - Ax^1 \quad (a_r - a_s) \qquad \text{if } a_s > a_r,$$

$$\gamma_r = \infty \quad \text{if} \quad a_s = a_r$$

7. Let $\delta = \text{Min}\{\delta_r, \delta_s, \gamma_r\}$ and redefine x^1 so that $x^1 = x^1 + \delta(e_r - e_s)$.

8. One or more of the following cases hold:

(a) If $\delta = \gamma_r$, let $S = S - \{r\}$. If also $\text{Min}\{\delta_r, \delta_s\} > 0$, then x^1 is optimal and the algorithm stops.

(b) If $\delta = \delta_s$, let $T = T - S_q$, where $q = \text{Max}\{k : k \in P_s - P_p \text{ and } f_k^1 = L_k\}$.

(c) If $\delta = \delta_r$, define $u = \text{Max}\{k : k \in P_r,\ k \leqq p, \text{ and } f_k^1 = U_k\}$. If u exists, let $S = S - S_u$. Also if $f_k^1 < U_k$ for all $k \in P_p - P_s$, let v be unchanged. Otherwise, define $v = \text{Min}\{k : k \in P_p - P_s \text{ and } f_k^1 = U_k\}$ and let $T = T \cap S_v$.

(d) If $\delta > 0$, let $T = T - \{j : j > r\}$.

(e) If $T = \varnothing$, let $S = S - \{j : a_j > 0\}$.

9. If r is removed from S by one of the above instructions, return to 2. Otherwise, return to 3.

Justification for the Composite Algorithm.[13] The algorithm is an application of Theorems 2, 3.1 and 4, and the earlier remarks of §6. The sets S and T respectively consist of those x_j for $j \in S_p$ and $j \notin S_p$ whose optimal values have not yet been determined. The initial contracting of S and T in instruction 1 is justified directly by Theorem 2(i) and (ii).

Selecting r in instruction 2 prepares the method to solve Problem $\text{I}^{h'}$ as defined in Theorem 3.1 for $h' = r$. We observe (as earlier) that Problem $\text{I}^{h'}$ stands in the same relation to Problem $\text{I}^{(h-1)'}$ as Problem III stands to Problem I (more precisely as Problem V stands to Problem I when $f_p = x_p$). The choice of s thus may be seen to correspond to establishing a revised indexing for the coefficients a_j' $(= a_r - a_j)$ of the variables z_j as defined in §6.2, treating Problem $\text{I}^{h'}$ as Problem III (with r taking the role of p). Instruction 4 accommodates the exceptional coefficient $a_r' = a_r$.

Restricting s so that $s < r$ in instruction 3 accords with Theorem 4 (and the implicit perturbation) which implies that $z_j = 0$ for $a_j < a_r$. The inapplicability of instruction 4 when $a_r > 0$ follows from the observation that $a_r > 0$ implies $a_r = 0$. Also, when s does not exist, the only variable currently to be assigned a value is z_r (unless it has already been assigned a value), which justifies the provision for s not existing in instructions 4 and 5. These remarks also justify the stipulation for contracting S in instruction 3 when s does not exist and $a_r > 0$. 8 (a) and (c) assure that 4 cannot be visited twice for the same value of r.

By Theorem 3.1 and the remarks of §§6.1 and 6.2, the optimal value for z_j is its

[13] A less intuitive and more elaborate description of this algorithm is given in §10, where it is used to solve a numerical example problem.

maximum value when values are assigned consecutively in the proper indexing sequence established by instructions 3, 4 and 5. Moreover, by the definitions of §6.2, the value of z_r represents an increment to x_r and the value of z_s represents an increment to x_r and a decrement to x_s. Thus, the maximum value of z_j is precisely δ as specified in instruction 7 (based on instructions 4 and 6).

We now consider the cases of instruction 8. For case (a), we let $S = S - \{r\}$ if if $\delta = \gamma_r$ since x_r cannot be further increased. To see this, note that $\gamma_r < \infty$ implies either $a_s > a_r$ at instruction 6 or $a_r < 0$ at instruction 4, and $a_0 = Ax^1$ for the new x^1 at instruction 7. Suppose r were not removed from S. The fact that the coefficient of z_j is negative in order to yield $\delta = \gamma_r < \infty$ (at instruction 4 for $j = r$ or instruction 6 for $j = s$) implies that coefficients of all subsequent z_j must also be negative, and $\delta = \gamma_r = 0$ will occur for all subsequent δ and γ_r until r changes. Thus, the solution being generated is unaltered by removing r from S immediately. The terminating condition of (a) is similarly justified by observing that once both $\delta = \gamma_r$ and Min $\{\delta_r, \delta_s\} > 0$,[14] then $\delta = 0$ thereafter (both for the current r and all subsequent r).[15]

Instructions 8 (b) and (c) are justified by the fact that in each successive Problem $\text{I}^{h'}$ the optimal value of x_j does not increase for $j \notin S_p$ and does not decrease for $j \in S_p$. Instruction 8 (d) applies Theorem 4, noting that $j < r$ "effectively" corresponds to $a_j > a_r$ by the implicit perturbation. Finally, (e) uses the observation of §6.2 that $T = \emptyset$ implies no x_j for $j \in S_p$ can increase unless $a_j \leqq 0$. Note that either S or T (or both) must be diminished by at least one element in (a), (b), or (c).[16]

Evidently, the process must return to 2 if r is removed from S and return to 3 otherwise, as stipulated in instruction 9. Thus, the algorithm accomplishes the successive solution of the Problems $\text{I}^{h'}$ of Theorem 3.1, taking advantage of tight upper and lower bounds to preassign certain x_j their optimal values by appropriately contracting S and T.

Composite Algorithm for Minimizing f_p. The algorithm for minimizing f_p may be described in terms of the algorithm for maximizing f_p by stipulating the following changes (justified by Theorems 2, 3.2, 4, and our earlier remarks).

Every occurrence of $a_j > 0$ is replaced by $a_j \leqq 0$ and every occurrence of $a_j \leqq 0$ is replaced by $a_j > 0$ (including $j = r$ and $j = s$).

In instructions 1 and 8, L_k is replaced by U_k and U_k by L_k. In instructions 2 and 3, Min and Max replace each other and inequalities in indices ($j < r$, $j <$ Min) are reversed ($j > r$, $j >$ Max).

In instructions 4 and 6, $f_k^1 - L_k$ and $U_k - f_k^1$ replace each other and a_r and a_s are replaced by their negatives in defining γ_r.

In instruction 7, the new x^1 is given by replacing δ with $-\delta$.

In instruction 8 (d), $j < r$ becomes $j > r$. Otherwise the algorithm for minimizing f_p is the same as for maximizing f_p.

We remark that the composite algorithms (like the earlier algorithms for Problems II, III, and IV) not only obtain an optimal value for f_p, but subject to this also maximize Ax.

[14] Min $\{\delta_r, \delta_s\} > 0$ will always hold if the optional contraction of S and T in instruction 1 is executed.

[15] A precise proof of the assertion requires showing that $\delta = 0$ for each possible way that $\gamma_r = \infty$ for subsequent r. We omit a detailed argument to exhaust the possibilities since it involves only a straightforward application of our previous observations.

[16] A possible exception occurs when 8 is visited directly after 4 and only v exists in 8 (c). But then 4 is inapplicable thereafter and the exception cannot be repeated. Note that this implies 8 (d) and 8 (e) may be regarded as optional. In fact, if the optional contraction of S and T by instruction 1 is performed, then 8 (d) and 8 (e) are redundant.

7. A Special Alternative Algorithm for Problem I

Drawing on the results of §§4 and 5, we give an alternative algorithm for Problem I that can be used as a basis for solving Problems II, III and IV by the procedures of §6. In the process we define new modified bounds which lead to feasibility theorems for these problems.

The alternative algorithm for Problem I is a mirror image of the algorithm of §4 and proceeds by assigning a value first to x_n, then to x_{n-1} and so on until a complete solution is obtained. To describe this algorithm, we reverse the roles of upper and lower bounds in §4 in the following definitions. For $j \leqq n$, let $U_j^{**} = U_j$. Then, for $j > n$ (proceeding in the order of indexing), let

$$U_k'' = \sum_{j \in B_k} U_j^{**} \quad \text{and} \quad U_k^{**} = \text{Min}\{U_k, U_k''\}.$$

Corresponding to Δ_k of §4, define $\theta_k = U_k^{**} - U_k''$ (note $\theta_k \leqq 0$). Finally, let $L_m^{**} = L_m'' = L_m$ and, for $k < m$, $k \in B_r$ (proceeding in reverse order of indexing), let $L_K'' = L_r^{**} - \sum_{h \in B_r : k} U_h^{**}$ and $L_k^{**} = \text{Max}\{L_k, L_k''\}$.

The counterpart of Theorem 1 using these definitions is as follows:

Theorem 5. *If Problem* I *has an optimal solution, then it has an optimal solution such that*

$$x_n = L_n^{**} \qquad \text{if } a_n \leqq 0$$

and

$$x_n = \text{Max}\{L_n^{**}, U_n + \sum_{k \in P_{n:m}} \theta_k\} \qquad \text{if } a_n > 0.$$

By replacing the symbol U (as in U_k, U_q^*, etc.) with L, replacing L with U, doubling every asterisk and prime, and reversing all inequalities (except those involving indices) the new values for the bounds L_k'', L_k^{**}, U_k'', U_k^{**}, and θ_k, based on the assignment $x_n = x_n^0$, are exact counterparts of the "updated" bounds specified in §4 and may be determined from them. (For example, the modified definition of w becomes $w = \text{Min}\{p : p \in P_n \text{ and } U_n + \sum_{k \in P_{n:p}} \theta_k \leqq x_n^0\}$.)

In addition, drawing on Corollary 1 and its counterpart, we may state the following results.

Corollary 3. *If Problem* I *has a feasible solution, then it has a feasible solution x^0 in which x_j^0 assumes any value satisfying $U_j^* \geqq x_j^0 \geqq L_j^{**}$.*

8. Feasible Solutions to Problem I

Corollary 1 of §4 and Corollary 3 of §7 provide information about the range of values x_j can take, given the existence of a feasible solution to Problem I. In this section we give necessary and sufficient conditions for the existence of a feasible solution to Problem I and specify upper and lower bounds for each arc that exactly constrain the range of feasible flows across that arc.

We begin by specifying such bounds for the arc m.

Theorem 6. *If Problem* I *has a feasible solution, then it has a feasible solution x^0 such that f_m^0 assumes any value satisfying $L_m^* \leqq f_m^0 \leqq U_m^{**}$.*

We note that a single forward pass through the arborescence suffices to determine the bounds U_m^{**} and L_m^* that exactly constrain the flow across arc m by Theorem 6. We now show that the bounds U_k^{**} and L_k^* generated in this forward pass in fact provide necessary and sufficient criteria for feasibility.

Theorem 7. *Problem* I *has a feasible solution if and only if $U_k^{**} \geqq L_k^*$ for all k.*

The bounds L_k^* and U_k^{**}, while sufficiently limiting to determine the existence or nonexistence of a feasible solution, are not in general the most restrictive *necessary* bounds on f_k^0 (except when $k = m$). Bounds that do exhibit this restrictive property are given in the next theorem.

THEOREM 8. *Let* $L_k^{\#} = \text{Max}\{L_k^*, L_k^{**}\}$ *and* $U_k^{\#} = \text{Min}\{U_k^*, U_k^{**}\}$. *If Problem* I *has a feasible solution, then for any arc p there is a feasible solution* x^0 *such that* f_p^0 *assumes any value satisfying* $L_p^{\#} \leqq f_p^0 \leqq U_p^{\#}$.

It may be observed that the bounds $L_k^{\#}$ and $U_k^{\#}$ of Theorem 8 can be determined by two passes through the arborescence. This is accomplished by replacing the original bounds L_k, U_k with the bounds L_k^*, U_k^{**} obtained on a forward pass and then computing L_k^{**} and U_k^* on a backward pass by reference to these revised bounds. The resulting L_k^{**} and U_k^* must be at least as limiting as $L_k^{\#}$, $U_k^{\#}$ and hence, by Theorem 8, must be equal to them if Problem I has a feasible solution.

9. Integer Solutions to Problems II, III and IV

We assume for this section that all L_k and U_k are integers and that the x_j are required to take integer values. Clearly, the integer requirement for the x_j poses no difficulty for Problem I, since the optimal solution obtained by the implicit perturbation acknowledged in §6.1 will automatically assign integer values to the x_j. On the other hand, Problems II, III and IV do not share this property with Problem I.

The results of §§6 and 7 nevertheless imply that an optimal integer solution to Problems II, III and IV can be obtained simply by rounding the fractional (continuous) solution. It should be stressed that Problems II–IV are among the very few for which such a rounding process is legitimate.

To emphasize the significance of this result we state it as our final theorem.

THEOREM 9. *If* x^1 *is an optimal fractional solution to Problem* V *of Theorem* 3.1 (*Problem* VI *of Theorem* 3.2) *obtained relative to the perturbation of* §6.1, *then at most two of the* x_j^1 *will not be integers. If only* x_r^1 *is not an integer, then* $r \in S_p$; *and if both* x_r^1 *and* x_s^1 *are not integers, then* $r \in S_p$, $s \notin S_p$ *and* $x_r^1 + x_s^1$ *is an integer. Finally, let Problem* V′ *be the same as Problem* V (*Problem* VI′ *the same as Problem* VI) *except that* f_p *is required to assume its largest* (*smallest*) *integer value. Then an optimal* (*continuous*) *solution to Problem* V′ (*Problem* VI′) *assigns each* x_j *an integer value, and is obtained by rounding* x_r^1 *down* (*up*) *for* $r \in S_p$ (*if it is not an integer*), *and rounding* x_s^1 *up* (*down*) *for* $s \notin S_p$ (*if it is not an integer*).

10. A Numerical Example

To illustrate the ideas of the foregoing sections we apply the algorithm of §4 for Problem I and the composite algorithm of §6.4 for Problem IV to the arborescence on the following page.

The nodes of the arborescence are drawn as circles, with the node index inside the circle, which also gives the index of the arc issuing from the right of the node. It may be seen that $n = 11$ and $m = 17$. Also, the sets S_k are given by $S_1 = \{1\}$, $S_2 = \{2\}, \cdots$, $S_{11} = \{11\}$, $S_{12} = \{3, 4\}$, $S_{13} = \{1, 5, 10\}$, $S_{14} = \{8, 11\}$, $S_{15} = \{3, 4, 6, 9\}$, $S_{16} = \{1, 5, 7, 8, 10, 11\}$, $S_{17} = \{1, 2, \cdots, 11\}$.

The numbers to the left of the initial nodes ($j \leqq n$) are the constants a_j of the vector $A = (13, 12, 7, 5, -1, -2, -5, -6, -7, -11)$. We also stipulate that $a_0 = 48$. The pair of numbers above each arc gives the values of the bounds L_k, U_k.

Solving Problem I.

We depict the solution of Problem I be the algorithm of §4 in Table 1. The first column gives the indices of the arcs k, and the second gives the vector (L_k, U_k) for

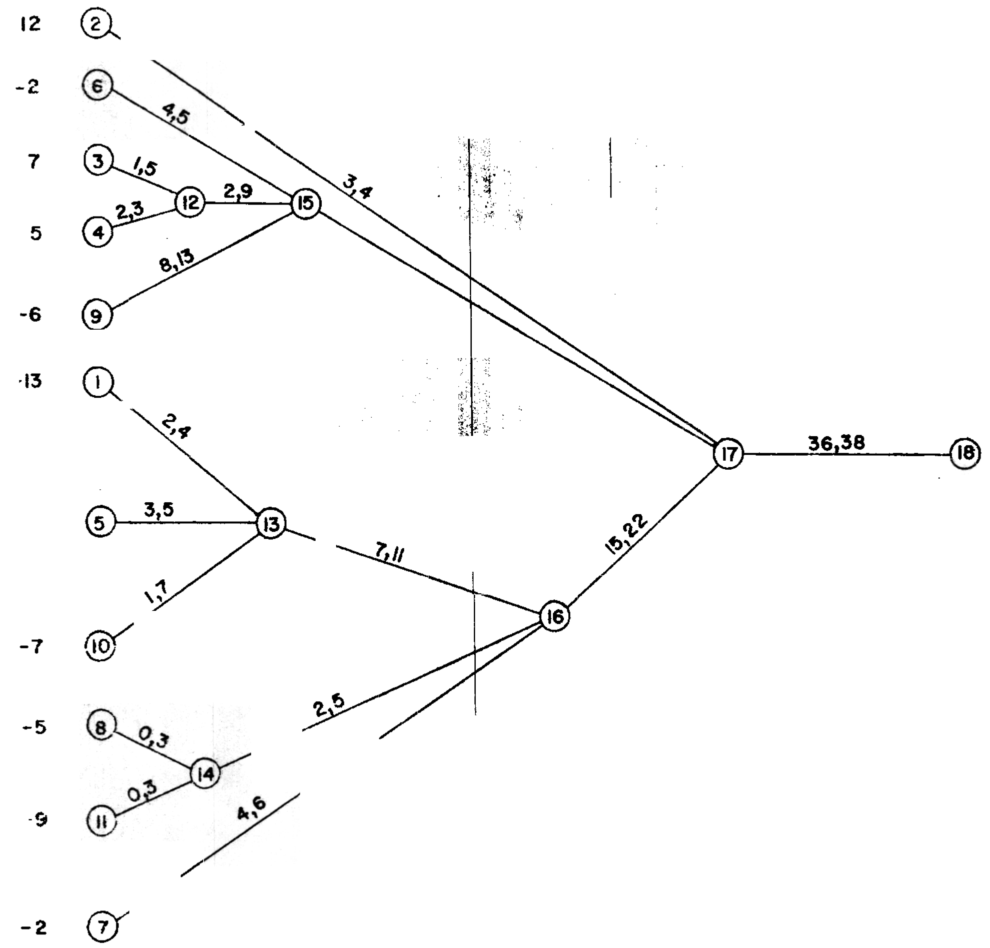

FIGURE 2. Diagram of Example Arborescence

each k, which may be seen to correspond to the bounds indicated in the diagram of the (L_k^*, U_k^*) for $k \leqq n$ $(n = 11)$, and (L_k^*, U_k^*, Δ_k) for $k > n$. An empty space in a vector means that the value of that component is unchanged from the previous vector. (The location of the empty space is emphasized by the use of commas when ambiguity may otherwise result.)

Asterisks in a column identify the arcs $k \in P_j$ where x_j is the variable currently being assigned a value. The value $L_j + \sum_{k \in P_j : m} \Delta_k$ can thus immediately be determined by reference to the asterisks.

The appropriate values underlying the assignment $x_j = x_j^0$ are given by the entries in the column preceding the one headed $x_j = x_j^0$. The column headed $x_j = x_j^0$ gives the new values after the assignment is made.[17]

Some comments about implementing the algorithm of §4 are in order. First, we have not bothered to record the current values of L_k' and U_k' in Table 1 since they are

[17] To verify these values by hand computation it is useful to write them in pencil under the appropriate arcs in the diagram, erasing old values and replacing them by current ones as they are computed.

TABLE 1

k	(L_k, U_k)	(L_k^*, U_k^*) for $k \leq n$ and (L_k^*, U_k^*, Δ_k) for $k > n$.					
		Initial	$x_1^0 = 4$	$x_2^0 = 4$	$x_3^0 = 5$	$x_5^0 = 4$	$x_8^0 = 2$
1	2, 4		0,0*				
2	3, 4			0, 0*			
3	1, 5				0,0*		
4	2, 3				,2		
5	3, 5				,4	0,0*	
6	4, 5				,4		
7	4, 6				,5	,4	
8	0, 3					,2	0,0*
9	8,13			,12	,8		
10	1, 7	, ,6	,4		,2	,1	
11	0, 3					,2	,0
12	2, 9	3, 8,0		, 7,	2, 2, *		
13	7,11	, ,1	4, 7,0*		, 5,	1, 1, *	
14	2, 5	, ,2			, 3,	, 2,	, 0,0*
15	13,20	15, ,0		,19,	14,14, *		
16	15,22	,20,2	11,16,1*	,15,	,11,	7, 7,0*	5, 5, *
17	36,38	, ,3	32,34, *	28,30,2*	25,25,0*	21,21,0*	19,19, *

not used except as a notational convenience to help define the current L_k^*, U_k^*, and Δ_k.[18]

As observed in §4, it is possible to determine U_k^* only for $k \in P_j$, where x_j is the variable currently being assigned a value. We have instead elected in Table 1 to compute updated values of U_k^* (as well as L_k^* and Δ_k) for all arcs k.

To facilitate this computation, note that $\bar{U}_p' \geqq U_p^*$ implies $\bar{U}_k^* = U_k^*$ (and hence U_k^* requires no updating) for all k such that $S_k \subseteq S_p$. Moreover, for w as defined in §4, it is readily verified that $\bar{U}_k^* = U_k^*$ for all $S_k \not\subseteq S_w$.

This last fact makes it possible to "postpone" an explicit assignment of values to some of the x_j and thereby reduce the amount of necessary updating. Specifically, if $L_j^* = U_j^*$, then of course $x_j^0 = L_j^*$ $(= L_j)$ is the only possible value of x_j, and hence $w = j$. But then $\bar{L}_k^* = L_k^* - x_j^0$ and $\bar{U}_k^* = U_k^* - x_j^0$ for $k \in P_j$, while all other bounds remain unchanged (including the new Δ_k values). Consequently, the values to be prescribed for subsequent variables will be the same if x_j^0 is not assigned the value L_j, but is simply bypassed. Thus, it is convenient to skip to the first j remaining such that $L_j^* < U_j^*$ and continue the algorithm from there. When no j $(\leqq n)$ are left, those x_j that were bypassed can all be set equal to L_j to complete the process.

This procedure has been followed in Table 1, as may be seen by noting that the assignment $x_3^0 = 5$ is followed directly by $x_5^0 = 4$, bypassing x_4. Similarly, $x_5^0 = 4$ is followed by $x_8^0 = 2$, bypassing x_6 and x_7. Finally, $U_j^* = L_j^*$ holds for all remaining $j \leqq n$ after the assignment $x_8^0 = 2$, and the algorithm terminates. The optimal solution is $x^0 = (4, 4, 5, 2, 4, 4, 4, 2, 8, 1, 0)$, yielding $Ax^0 = 60$.

[18] Note too that to determine U'_k it is convenient (upon computing U_h^*) to compute $U_h^* - L_h^* + \Delta_h$ and leave this amount unchanged to find $U'_k = (U_h^* - L_h^* + \Delta_h) + L_k^*$ for all $k \in B_h$.

Solving Problem IV

Before illustrating the solution of Problem IV, we will supplement the general description of the composite algorithm of §6.4 by giving explicit rules for implementing the instructions that define δ_r, δ_s, and the sets S and T.

To begin, we assign numbers $\alpha_k = U_k - f_k^0$ ($f_k^0 - L_k$ if minimizing f_p) to all arcs k such that $k \in P_p$ or $S_k \subset S_p$, and assign $\alpha_k = f_k^0 - L_k$ ($U_k - f_k^0$ if minimizing f_p) to all other arcs. Then the rules for determining δ_r, δ_s and for contracting S and T in instructions 1, 3, 4, 6 and 8 of the composite algorithm can be implemented as follows.[19] Instructions 2, 7 and 9 remain unchanged and 5 becomes superfluous. Except as indicated below, our remarks apply both to maximizing and minimizing f_p.

1. As before, initially let $S = S_p$, $T = S_m - S_p$ and $x^1 = x^0$, where x^0 is optimal for Problem I.

(a) Determine new values α_k^* for α_k when $k \in P_{p:m}$ and $p < m$ as follows. (If $p = m$ there is nothing to determine.) Begin with $q = \text{Min}\{k : k \in P_{p:m}\}$ and let $\alpha_q^* = \alpha_q$. Then, if α_h^* has been determined for $h \in P_{p:m}$ and $\alpha_h^* = 0$, let $v = h$ and drop all arcs k of the arborescence such that $S_k \not\subset S_v$, hence contracting T so that $T = T \cap S_v$. (Note that this obviates the determination of α_k^* for $k \in P_{h:m}$.) But if $\alpha_h^* > 0$, $h \in B_k$, and α_k^* has not been determined, then let $\alpha_k^* = \text{Min}\{\alpha_h^*, \alpha_k\}$. If by this process α_m^* is eventually determined and $\alpha_m^* > 0$, then let $v = t$.

(b) Determine new values α_k^* for α_k when $k \notin P_{p:m}$ as follows. Let $\alpha_p^* = \alpha_p$ and, for all $h \in P_{p:v}$ and all $k \in B_{h:p}$, let $\alpha_k^* = \alpha_k$. (For notational convenience, let $P_{p:t} = P_{p:m}$.) Then, if α_h^* has been determined for $h \in P_{p:m}$ and $\alpha_h^* = 0$, eliminate all arcs k from consideration such that $S_k \subseteq S_h$ (as by "disconnecting" h from the arborescence), thus letting $S = S - S_h$ if $S_h \subseteq S_p$ and $T = T - S_h$ otherwise. If, on the other hand, $\alpha_h^* > 0$ and α_k^* has not been determined for some $k \in B_h$, define $\alpha_k^* = \text{Min}\{\alpha_k, \alpha_h^*\}$. Repeat until no more α_k^* can be determined by the rule (hence α_j^* has been determined for all $j \in S \cup T$, for the remaining S and T).

(c) Redefine $\alpha_k = \alpha_k^*$ (i.e., update α_k) for all arcs k remaining for which α_k^* has been determined.

3. In this instruction the definition of s becomes $s = \text{Max}\{j : j \in T\}$ ($\text{Min}\{j : j \in T\}$ if minimizing f_p) and the conditional contraction of S when s does not exist, etc., is given by $S = S - \{j : a_j > 0\}$ ($S - \{j : a_j \leqq 0\}$ if minimizing f_p).

4. The only change in this instruction is to let

$$\delta_r = \text{Min}\{\alpha_r, \alpha_m\}$$

and define $g = m$ (for instruction 8).

6. Instruction 6 remains unchanged except that the definitions of δ_r and δ_s simplify to $\delta_s = \alpha_s$ and $\delta_r = \text{Min}\{\alpha_r, \alpha_g\}$, where $g = \text{Max}\{k : k \in P_p - P_s\}$.

8. This instruction updates the α_k and performs the functions of the old instruction 8 as follows.[20]

(a) If $\delta = \gamma_r$, x^1 is optimal and the algorithm stops. Define $\alpha_k^* = \alpha_k - \delta$ for $k \in P_r - P_s$ and $k \in P_s - P_r$.

[19] This particular statement of the composite algorithm was purposely avoided in §6.4 in favor of the more intuitive description that facilitated the algorithm's justification. Hopefully, the elaboration of details in the present description will thereby be more easily understood.

[20] The divisions of the new instruction 8 do not precisely correspond to those of the old. The use of the optional rules for contracting S and T in instruction 1 account for the elimination of cases 8 (d) and (e) of §6.4 and the changes in instructions 3 and 8 (a).

(b) If $\alpha_g^* > 0$ and $g < m$ (for g given in instruction 6), let

$$Q = \{k : k \in P_g \text{ and } g < k < v\}.$$

If $Q = \emptyset$, then part (b) of instruction 8 may be bypassed. Otherwise, identify $h = \text{Min}\{k : k \in Q \text{ and } \alpha_k = \alpha_g^*\}$. If h does not exist, let $\alpha_k^* = \alpha_g^*$ for $k \in Q$. But if h exists, let $\alpha_k^* = \alpha_k$ for all $k \in Q$ such that $k \geqq h$ and let $\alpha_k^* = \alpha_g^*$ for all $k \in Q$ such that $k < h$.

(c) If $\alpha_g^* = 0$ (for g given in 4 or 6), redefine v so that

$$v = \text{Min}\{k : k \in P_p - P_s \text{ and } \alpha_k^* = 0\},$$

and drop all arcs k such that $S_k \not\subseteq S_v$, hence contracting T so that $T = T \cap S_v$.

(d) If $s \neq 0$, let $q = \text{Max}\{k : k \in P_s - P_r\}$. If $s = 0$ (from instruction 4), for definitional purposes let $q = 0$ and $S_q = \emptyset$. Then for h such that $S_h \subseteq S_p$ or $S_h \subseteq S_q$, if α_h^* is determined and $\alpha_h^* = 0$, remove all arcs k from consideration such that $S_k \subseteq S_h$, thus letting $S = S - S_h$ if $S_h \subseteq S_p$ and $T = T - S_h$ otherwise.

(e) If α_h^* has been determined and $\alpha_h^* = \alpha_h$, then $\alpha_k^* = \alpha_k$ (hence these α_k need not be updated) for all k such that $S_k \subseteq S_h$.

(f) Finally, if $\alpha_h > \alpha_h^* > 0$, then for $k \in B_h$ let $\alpha_k^* = \text{Min}\{\alpha_k, \alpha_h^*\}$.

(g) When no more α_k^* can be determined by these rules, redefine $\alpha_k = \alpha_k^*$ for those k left for which α_k^* was determined (letting the other α_k retain their old values).

The justification for these instructions follows from the justification of §6.4 and the definitions of δ_r, δ_s and the sets S and T. It is to be observed that the α_k represent upper bounds on flow increments and decrements and, except for the arcs $k \in P_{p:m}$,

TABLE 2

	k	α_k			
		Starting	Initial Revision	$r = 5$ $s = 3$ $\delta = 1$	$r = 10$ $s = 3$ $\delta = 2/7$
	1	0	X		
	2				
	3	4		3	
	4	0	X		
	5	1		X	
	6	0	X		
	7	0	X		
	8	2	X		
	9	0	X		
	10	5	2	1	
	11	0	X		
	12	4		3	
	13	2		1	
	14	0	X		
	15	4		3	
	16	7		6	
	17	0			
$a_0 - Ax^1$		-12	-12	-12	-4
$a_r - a_s$				-8	-14
$(\delta_r, \delta_s, \gamma_r)$				1, 4, 3/2	1, 3, 2/7

α_k^* corresponds to α_k as U_k^* corresponds to U_k, taking into account the fact that all lower bounds are zero.

The result of applying the composite algorithm to maximize f_{13} of the example problem, subject to $Ax \geqq 48$, is summarized in Table 2. As before, a blank space indicates that the entry is unchanged from the corresponding entry in the preceding column. The initial elements of S are indicated by the symbol S to the left of the appropriate arc indices in the table (the initial elements of T consist of all remaining $k \leqq 11$). The X's indicate the arcs of the arborescence that are dropped by the revision process, thus in particular indicating for $k \leqq 11$ the arcs eliminated from S and T.

Starting values for the α_k are given by using the original computed values of L_k^* and U_k^* in place of L_k and U_k.

The indices r, s, and the value of δ in each of the last two columns are determined by reference to information in the preceding column. Revised values for the α_k in the final column are omitted since the optimal solution has been obtained.

The optimal solution is seen to be $x_3^1 = x_3^0 - \frac{9}{7} = 3\frac{5}{7}$, $x_5^1 = x_5^0 + 1 = 5$, $x_{10}^1 = x_{10}^0 + \frac{2}{7} = 1\frac{2}{7}$, and $x_j^1 = x_j^0$ for all other j, yielding a maximum value for f_{13} of $10\frac{2}{7}$.

References

1. Berge, Claude, *The Theory of Graphs and Its Applications*, John Wiley & Sons, Inc., New York, 1962.
2. Cook, R. A. and Cooper, L., "An Algorithm for Integer Linear Programming," Report No. AM65-2, School of Engineering and Applied Science, Washington University, St. Louis, Mo., (presented at the 28th National ORSA Meeting, November 1965).
3. Ford, L. R., Jr. and Fulkerson, D. R., *Flows in Networks*, Princeton University Press, Princeton, New Jersey, 1962.
4. Glover, Fred, "Truncated Enumeration Methods for Solving Pure and Mixed Integer Linear Programs," WP-27, Operations Research Center, University of California, Berkeley, (presented at the 29th National ORSA Conference, May 1966).
5. ——, "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," *Journal of the Operations Research Society of America*, (November–December 1965).
6. ——, "Flows in Arborescences," NONR 760(24) NR 047-048, Carnegie Institute of Technology, Pittsburgh, Pennsylvania, (July, 1967).
7. Hillier, Frederick, S., "An Optimal Bound and Scan Algorithm for Integer Linear Programming," Technical Report No. 3, Department of Industrial Engineering, Stanford University, (August 1966).
8. Ignall, Edward and Veinott, Jr., Arthur F., "Optimality of Myopic Inventory Policies for Several Substitute Products," *Management Science*, Vol. 15, No. 5 (January 1969), pp. 284–304.