MEMORANDUM
RM-5456-PR
MARCH 1968

THE NUMERICAL SOLUTION OF NETWORK PROBLEMS USING THE OUT-OF-KILTER ALGORITHM

R. J. Clasen

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-5456-PR
MARCH 1968

THE NUMERICAL SOLUTION OF NETWORK PROBLEMS USING THE OUT-OF-KILTER ALGORITHM

R. J. Clasen

This research is supported by the United States Air Force under Project RAND — Contract No. F44620-67-C-0045 — monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. RAND Memoranda are subject to critical review procedures at the research department and corporate levels. Views and conclusions expressed herein are nevertheless the primary responsibility of the author, and should not be interpreted as representing the official opinion or policy of the United States Air Force or of The RAND Corporation.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA • 90406

Published by The RAND Corporation

PREFACE

This Memorandum is part of RAND's continuing effort in the application of network flow theory. It provides a graphic description of the out-of-kilter algorithm, together with useful computational methods. Network flow problems arise in the solution of transportation and scheduling problems. This work is directed toward the user and programmer of network-solving algorithms.

Portions of this material were presented at the SHARE XXIX meeting, August 1967.

SUMMARY

Network flow problems arise in the solution of transportation and scheduling problems. Divided into four substantially independent sections, this Memorandum: 1) reviews the types of problems that are representable as capacitated network problems; 2) explains (with diagrams) the out-of-kilter algorithm and techniques for implementing it on a computer; 3) describes modification of the algorithm to a two-phase algorithm; 4) presents a method for labeling the nodes by means of a scan list. Tentative conclusions are that the two-phase algorithm is undesirable, and that the labeling procedure shortens computer time at the cost of using more memory.

CONTENTS

FIGURES

SYMBOLS

The subscript notation is equivalent to parentheses or brackets; e.g., $S_j \equiv S(j) \equiv S[j]$.

a_{ij}	Incidence matrix.
b_i	Node flow.
c_j	Unit cost.
$\bar{c}_j$	Reduced cost.
d_j	Total cost, as a function of flow, in one arc.
e_j	Cycle indicators.
g_j	Scrambled source node arc list.
h_j	Scrambled sink node arc list.
i	Node subscript.
j	Arc subscript.
J	Target arc.
k	Node subscript.
k_j	Kilter number.
K	Total kilter number.
ℓ_j	Lower capacity.
L_i	Node label.
m	Number of nodes.
n	Number of arcs.
p	Position of scanner.

q_j	Infeasibility number.
Q	Total infeasibility.
R	Scan list.
refnode	Reference node.
s	Length of scan list.
S_j	Source node.
T_j	Sink node.
u_j	Upper capacity.
U_i	Source node reference list.
V_i	Sink node reference list.
x_j	Amount of flow.
y_j	Alternate designation of flow.
γ_j	Special reduced cost.
Φ	Objective value.
λ_j	Dual slack variable.
π_i	Node price. pi(i) is used in ALGOL programs.
ρ_i	Temporary symbol for constructing the g_j list.
σ_i	Temporary symbol for constructing the U_i list.
τ_i	Temporary symbol for constructing the V_i list.

I. INTRODUCTION

THE CAPACITATED NETWORK PROBLEM

A network is made up of a set of nodes together with a set of directed arcs. Figure 1 is an example of a network with five nodes and eight arcs. The nodes are designated by the letters A, B, C, D, and E, and directed arcs

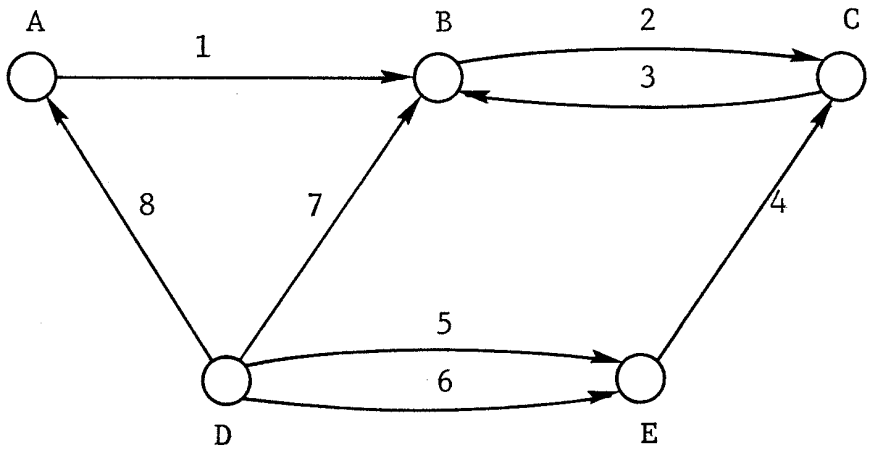

Fig. 1--Example Network

are numbered 1, 2, 3, 4, 5, 6, 7, and 8. An alternative way of designating these arcs is by ordered pairs of nodes; hence, arc 1 is also arc (A,B), arc 2 is arc (B,C), arc 3 is arc (C,B), etc. The first node of the ordered pair, called the source node, is the node from which the indicated arrow is directed; the second node, called the

sink node, is the node toward which the arrow is directed. Note that arcs 5 and 6 are both designated as arc (D,E), so that a subscript on the ordered pair of nodes must be used to distinguish between non-unique ordered pairs. Hence, arc 5 is arc $(D,E)_1$ and arc 6 is arc $(D,E)_2$. Also, the set of arcs is non-exhaustive; e.g., there is no arc (C,D) or (D,C) in Fig. 1.

The capacitated network problem consists of a network, together with the following four quantities for each arc j:

1) c_j: The cost of sending one unit of flow along arc j from its source node to its sink node.

2) u_j: The upper capacity of arc j.

3) ℓ_j: The lower capacity of arc j.

4) x_j^0: The nominal flow along arc j.

Denoting the number of nodes in the network as m and the number of arcs as n, the nominal flows, x_j^0, determine the node constraints

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i=1,2,\ldots,m \ .$$

Here

$$a_{ij} = \begin{cases} +1 & \text{if node } i \text{ is the source node for arc } j; \\ -1 & \text{if node } i \text{ is the sink node for arc } j; \\ 0 & \text{otherwise.} \end{cases}$$

And b_i is found by evaluating

$$b_i = \sum_{j=1}^{n} a_{ij}\, x_j^0 \qquad i=1,2,\ldots,m \ .$$

Note that each column of the (a_{ij}) matrix has only two non-zero entries: +1 and -1. Since

$$\sum_{i=1}^{m} b_i = \sum_{j=1}^{n} \left(x_j^0 \sum_{i=1}^{m} a_{ij} \right) = 0 \ ,$$

at least one node constraint is redundant. Specifying a nominal flow, x_j^0, uniquely determines $b_1, b_2, \ldots, b_m$, such that $b_1 + b_2 + \ldots + b_m = 0$. A <u>nominal solution</u> is any vector $(x_1, x_2, \ldots, x_n)$ that satisfies

$$\sum_{j=1}^{n} a_{ij}\, x_j = b_i \qquad i=1,2,\ldots,m \ .$$

A <u>feasible solution</u> of the capacitated network problem is any vector $(x_1,x_2,\ldots,x_n)$, such that

$$1) \quad \sum_{i=1}^{n} a_{ij}\, x_j = b_i \qquad i=1,2,\ldots,m$$

$$2) \quad x_j \geq \ell_j \qquad j=1,2,\ldots,n$$

$$3) \quad x_j \leq u_j \qquad j=1,2,\ldots,n \ ;$$

that is, a feasible solution is a nominal solution that satisfies $\ell_j \leq x_j \leq u_j$ for all arcs j. If there is at least one feasible solution to the problem, then an <u>optimal solution</u> is a feasible solution that minimizes

$$4) \quad \Phi = \sum_{j=1}^{n} c_j\, x_j \ ,$$

where Φ is called the <u>objective</u> value. That is, a feasible solution $(y_1,y_2,\ldots,y_n)$ is optimal if

$$\sum_{j=1}^{n} c_j\, y_j \leq \sum_{j=1}^{n} c_j\, x_j$$

for all feasible solution vectors $(x_1,x_2,\ldots,x_n)$.

The quantity b_i, called the node i flow, is the net flow out of node i. If b_i is zero, node i is said to be conservative. If all nodes are conservative, then the zero vector is a nominal solution, and the network is conservative. The algorithm for solving the capacitated network problem requires a nominal solution rather than a set of node flows. A simple method of determining a nominal solution is to make the network conservative. This is done by adjoining one more node, called the supernode, to the network, together with an arc from each node that is not conservative to the supernode. The lower capacity and upper capacity of each additional arc are both set equal to the net flow out of the node. The zero vector is then a nominal solution for this augmented network.

Note that there are no sign restrictions on any of the quantities c_j, u_j, ℓ_j, x_j^0, x_j or b_i. However, the problem is trivially infeasible if $u_j < \ell_j$.

A simple capacitated network is a capacitated network with lower capacities (ℓ_j) of zero. A general network may be translated to make it simple. Let

$$\tilde{u}_j = u_j - \ell_j$$

$$\tilde{x}_j^0 = x_j^0 - \ell_j$$

$$\tilde{x}_j = x_j - \ell_j$$

for all j. The node constraints above were

$$\sum_{j=1}^{n} a_{ij}\, x_j = \sum_{j=1}^{n} a_{ij}\, x_j^0 \qquad i=1,2,\ldots,m \ ,$$

where the right sides to these equations are the constants

$$b_i = \sum_{j=1}^{n} a_{ij}\, x_j^0 \qquad i=1,2,\ldots,m \ .$$

Substituting the above quantities having a diacritic tilde into the node constraints, capacity inequalities and the objective function results in

1) $$\sum_{j=1}^{n} a_{ij}(\tilde{x}_j + \ell_j) = \sum_{j=1}^{n} a_{ij}(\tilde{x}_j^0 + \ell_j) \qquad i=1,2,\ldots,m$$

2) $$\tilde{x}_j + \ell_j \geq \ell_j \qquad j=1,2,\ldots,n$$

3) $$\tilde{x}_j + \ell_j \leq u_j + \ell_j \qquad j=1,2,\ldots,n$$

$$4) \quad \Phi = \sum_{j=1}^{n} c_j(\tilde{x}_j + \ell_j) \ .$$

By letting

$$\tilde{b}_i = \sum_{j=1}^{n} a_{ij} \tilde{x}_j^0$$

and

$$\tilde{\Phi} = \Phi - \sum_{j=1}^{n} c_j \ell_j \ ,$$

the above becomes

$$1) \quad \sum_{j=1}^{n} a_{ij} \tilde{x}_j = \tilde{b}_i \qquad i=1,2,\ldots,m$$

$$2) \quad \tilde{x}_j \geq 0 \qquad j=1,2,\ldots,n$$

$$3) \quad \tilde{x}_j \leq \tilde{u}_j \qquad j=1,2,\ldots,n$$

$$4) \quad \tilde{\Phi} = \sum_{j=1}^{n} c_j \tilde{x}_j \ .$$

Since $\tilde{\Phi}$ is at its minimum if and only if Φ is at its minimum, the solution of the translated problem is the translation of the solution of the original problem.

Other linear, and some non-linear, network and transportation problems are representable as capacitated network problems [1].* The classical transportation problem involves a network whose nodes are divided into two classes: source nodes and destination nodes. If there are s source nodes and d destination nodes, then there are sd directed arcs; one arc from each source node to each destination node. These arcs are uncapacitated, which means that $\ell_j = 0$ and $u_j = +\infty$ for each arc j. Each source node has a net flow out of it, called the supply; and each destination node has a net flow into it, called the demand. Each arc has a unit cost which is the same as c_j above. This problem may be solved as a capacitated network problem by making the upper capacities very large, although special algorithms are available that solve specifically this problem. The capacitated transportation problem is as above, except that (some of) the upper capacities are not infinite. If an upper capacity is zero, the corresponding arc need not be included when solving the capacitated transportation problem as a capacitated network problem.

---

*See also Ref. 2, Chap. 14.

The capacitated network algorithm will solve problems with non-linear cost functions if the unit cost function for a given arc is piecewise constant and monotone increasing (non-decreasing). Let

$$d(x) = \int c(x)dx \ ,$$

where $c(x)$ is the unit cost function, and $d(x)$ is the total cost as a function of the flow x. Then $d(x)$ must be piecewise linear and convex. If the cost function has r "pieces"--

$$\begin{array}{lll}
x_0 \le x \le x_1 \ , & c(x) = c_1 \ , & d(x) = c_1 x + c_0 \\
x_1 < x \le x_2 \ , & c(x) = c_2 \ , & d(x) = d(x_1) + c_2(x-x_1) \\
x_2 < x \le x_3 \ , & c(x) = c_3 \ , & d(x) = d(x_2) + c_3(x-x_2) \\
\vdots & \vdots & \vdots \\
x_{r-1} < x \le x_r \ , & c(x) = c_r \ , & d(x) = d(x_{r-1}) + c_r(x-x_{r-1})
\end{array}$$

where c_0 is arbitrary and

$$c_1 \le c_2 \le c_3 \le \ldots \le c_r \ ,$$

--then this arc is represented by r parallel arcs (having the same source node and sink node) with

$$\ell_1 = x_0 \qquad u_1 = x_1$$

$$\ell_2 = 0 \qquad u_2 = x_2 - x_1$$

$$\ell_3 = 0 \qquad u_3 = x_3 - x_2$$

$$\vdots \qquad\qquad \vdots$$

$$\ell_r = 0 \qquad u_r = x_r - x_{r-1}$$

and unit costs $c_1, c_2, c_3, \ldots, c_r$ and nominal flows such that

$$\sum_{i=1}^{r} x_i^0$$

is the nominal flow for the composite arc. The answer (optimum flow) for the composite arc is

$$\sum_{i=1}^{r} x_i \ .$$

This representation of the composite arc is not unique, as can be seen from the converse. Suppose that there are r parallel arcs with lower capacities $\ell_1, \ell_2, \ldots, \ell_r$; upper capacities $u_1, u_2, \ldots, u_r$; unit costs $c_1, c_2, \ldots, c_r$ with

$c_1 \le c_2 \le \ldots \le c_r$; and nominal flows $x_1^0, x_2^0, \ldots, x_r^0$. Then these r arcs are equivalent to the arc with the following composite unit cost:

$$c(x) = \begin{cases} c_1 : \ell_1 + \ell_2 + \ldots + \ell_r \le x \le u_1 + \ell_2 + \ldots + \ell_r \\ c_2 : u_1 + \ell_2 + \ldots + \ell_r < x \le u_1 + u_2 + \ell_3 + \ldots + \ell_r \\ \vdots \\ c_i : \sum_{1}^{i-1} u_j + \sum_{i}^{r} \ell_j < x \le \sum_{1}^{i} u_j + \sum_{i+1}^{r} \ell_j \\ \vdots \\ c_r : u_1 + u_2 + \ldots + u_{r-1} + \ell_r < x \le u_1 + u_2 + \ldots + u_r \end{cases}$$

with lower capacity

$$\sum_{j=1}^{r} \ell_j ,$$

upper capacity

$$\sum_{j=1}^{r} u_j ,$$

and nominal flow

$$\sum_{j=1}^{r} x_j^0 \ .$$

Thus, for example, the composite arc whose total cost is the absolute value of the flow has composite unit cost

$$c(x) = \begin{cases} -1 \ : \ -u \le x \le 0 \\ \\ +1 \ : \ 0 < x < u \end{cases} ;$$

and if the lower bound is the negative of the upper bound u and the nominal flow is x^0, then this arc is equivalent to the two parallel arcs with:

$$\ell_1 = -u + \alpha \qquad u_1 = \alpha \qquad c_1 = -1 \qquad x_1^0 = \beta$$

$$\ell_2 = -\alpha \qquad u_2 = u - \alpha \qquad c_2 = +1 \qquad x_2^0 = x^0 - \beta$$

where α and β are arbitrary. Although α and β are arbitrary as far as the optimal flow $(x_1 + x_2)$ is concerned (if unique), the value for β may affect the speed of solution; while if α is not zero, the objective function (Φ) is decreased by the constant 2α.

A (simple) arc can be reversed in direction by changing the signs of its unit cost and nominal flow and reversing the upper and lower capacities and changing their signs. Denoting a direction reversal by a circumflex, this transformation is

$$\hat{\ell} = -u$$

$$\hat{u} = -\ell$$

$$\hat{c} = -c$$

$$\hat{x}^0 = -x^0 \ .$$

Returning to the capacitated network problem in its translated form (zero lower capacities)--

1) $\sum_{j=1}^{n} a_{ij} x_j = b_i$ $\qquad i=1,2,\ldots,m$

2) $0 \le x_j \le u_j$ $\qquad j=1,2,\ldots,n$

3) minimize $\sum_{j=1}^{n} c_j x_j$

--this problem may be stated as a primal linear programming problem by defining n slack variables $x_{n+1}, x_{n+2}, \ldots, x_{2n}$. The linear programming problem is then:

$$\text{minimize} \sum_{j=1}^{n} c_j x_j \text{ with } x_1, x_2, \ldots, x_{2n} \text{ non-negative, subject to}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i=1,2,\ldots,m$$

$$x_j + x_{n+j} = u_j \qquad j=1,2,\ldots,n \ .$$

Associated with the linear programming problem solution is a set of shadow prices (the negative of the dual variables) $\pi_1, \pi_2, \ldots, \pi_m, \lambda_1, \lambda_2, \ldots, \lambda_n$. A necessary condition for optimality is that there exist shadow prices such that

$$\left.\begin{aligned} c_j + \sum_{i=1}^{m} a_{ij} \pi_i + \lambda_j \geq 0 \\ \lambda_j \geq 0 \end{aligned}\right\} \quad j=1,2,\ldots,n \ .$$

Setting $\bar{c}_j = c_j + \sum_{i=1}^{m} a_{ij}\,\pi_i$:

$$\left.\begin{aligned} \bar{c}_j + \lambda_j &\geq 0 \\ \lambda_j &\geq 0 \end{aligned}\right\} \qquad j=1,2,\ldots,n\ .$$

Moreover, if x_j ($j \leq n$) is positive (then variable j is basic), the equality holds:

$$\bar{c}_j + \lambda_j = 0\ ,$$

$$\text{i.e., } \bar{c}_j \leq 0\ ;$$

and if x_{j+n} is positive (i.e., $x_j < u_j$), then

$$\lambda_j = 0,$$

$$\text{i.e., } \bar{c}_j \geq 0\ .$$

Hence, if

$0 < x_j < u_j$:

$$\bar{c}_j = 0 \ .$$

The quantities $\pi_1, \pi_2, \ldots, \pi_m$ are called the node prices. The quantities $\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_n$ are the marginal or reduced costs. Since only two of the quantities $a_{1j}, a_{2j}, \ldots, a_{mj}$ are non-zero--in particular, if i is the source node for arc j, then $a_{ij} = +1$; and if k is the sink node for arc j, then $a_{kj} = -1$--

$$\bar{c}_j = c_j + \pi_i - \pi_k$$

or in a double subscripted arc notation:

$$\bar{c}_{ik} = c_{ik} + \pi_i - \pi_k \ .$$

The implications

$$\bar{c}_j \leq 0 \quad \text{if} \quad x_j > 0$$

$$\bar{c}_j \geq 0 \quad \text{if} \quad x_j < u_j$$

are also <u>sufficient</u> conditions for $x_1, x_2, \ldots, x_n$ being an optimal solution since

$$\Phi = \sum_{j=1}^{n} c_j \, x_j$$

$$= \sum_{j=1}^{n} \left(\bar{c}_j - \sum_{i=1}^{m} \pi_i \, a_{ij} \right) x_j$$

$$= \sum_{j=1}^{n} \bar{c}_j \, x_j - \sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} a_{ij} \, x_j$$

$$= \sum_{j=1}^{n} \bar{c}_j \, x_j - \sum_{i=1}^{m} \pi_i \, b_i$$

cannot be decreased by making a feasible change in the flow (x_j). That is, a feasible solution $(x_1, x_2, \ldots, x_n)$ has minimum Φ if there exist shadow prices $(\pi_1, \pi_2, \ldots, \pi_m)$ such that $x_j > 0$ implies $\bar{c}_j \leq 0$ and $x_j < u_j$ implies $\bar{c}_j \geq 0$.

II. THE OUT-OF-KILTER ALGORITHM

The method described here is similar to that set forth in Ref. 3. For each arc, j, let S[j] be its source node and T[j] its sink node. Each node, i, has a price, $\pi(i)$ or π_i. The reduced cost, $\bar{c}_j$, of arc j, is related to its unit cost, c_j, by

$$\bar{c}_j = c_j + \pi(S[j]) - \pi(T[j]) \ .$$

A set of flows $x_1, x_2, \ldots, x_n$ in the arcs is called a nominal solution if

$$\sum_{j=1}^{n} a_{ij} \, x_j = b_i \qquad i=1,2,\ldots,m \qquad (1)$$

where the (a_{ij}) matrix is such that for each arc j,

$$a_{ij} = \begin{cases} +1 & \text{if} \quad i = S[j] \\ -1 & \text{if} \quad i = T[j] \\ 0 & \text{otherwise} \end{cases}$$

and the b_i are integers, known as the node flows. A nominal solution is a feasible solution if, in addition to Eq. (1),

$$\ell_j \le x_j \le u_j \qquad j=1,2,\ldots,n \qquad (2)$$

where ℓ_j and u_j are given integers. ℓ_j is known as the lower capacity, and u_j is known as the upper capacity. A feasible solution is an optimal solution if

$$\Phi = \sum_{j=1}^{n} c_j x_j$$

is a minimum over all feasible solutions. A necessary and sufficient condition that a feasible solution be optimal is that there exist $\pi_1,\pi_2,\ldots,\pi_m$ such that the $\bar{c}_j$ calculated from these π_i satisfy:

$$\left.\begin{aligned} \bar{c}_j \ge 0 \quad \text{if} \quad x_j < u_j \\ \bar{c}_j \le 0 \quad \text{if} \quad x_j > \ell_j \end{aligned}\right\} \quad j=1,2,\ldots,n \; . \qquad (3)$$

For any set of nominal x_j and any π_i, let k_j be the kilter number* of arc j, defined as follows:

*This differs from the kilter number defined in Ref. 3.

$$k_j = \begin{cases} |x_j - u_j| & \text{if} \quad \bar{c}_j < 0 \\ \max\,(0,\, x_j - u_j,\, \ell_j - x_j) & \text{if} \quad \bar{c}_j = 0 \\ |x_j - \ell_j| & \text{if} \quad \bar{c}_j > 0\ , \end{cases}$$

or equivalently,

$$k_j = \max(0,\, x_j - u_j,\, \ell_j - x_j,\, \operatorname{sgn}(\bar{c}_j)(x_j - \ell_j),\, \operatorname{sgn}(\bar{c}_j)(x_j - u_j))$$

By comparing k_j with Eqs. (2) and (3), it can be seen that every k_j is zero only if the solution is optimal. The value of k_j, never negative, is the amount that arc j is "out-of-kilter." The amount that the problem is "out-of-kilter" is

$$K = \sum_{j=1}^{n} k_j\ .$$

The network problem could thus be stated as the problem of finding π_i and nominal x_j such that K is a minimum. Let K_0 be the minimum such K. If K_0 is zero (i.e., the problem is feasible), it is possible to reduce each k_j to zero without increasing any of the other kilter numbers in the process. If the problem is infeasible ($K_0 \neq 0$), the algorithm described

in this section may not attain $K = K_0$, because if $K_0 > 0$ a lower value of K is sometimes attainable only at the expense of increasing the kilter numbers of some arcs. In order to attain $K = K_0$, it is (sometimes) necessary to use a "two-phase" algorithm for which the second phase is given here. The first phase of this "two-phase" algorithm is discussed in Sec. III. For most purposes, the algorithm described in this section is sufficient, since one usually deals with a feasible problem or a problem with obvious infeasibilities.

The algorithm for solving a network problem begins with any node prices and any nominal solution, then proceeds as follows. An arc is found with a non-zero kilter number. Then a labeling procedure is initiated that attempts to find a cycle of arcs along which at least one unit of flow can be pushed without increasing the kilter numbers in any arc in the cycle. If such a cycle is found, the indicated flow change is made, thus decreasing the kilter number of at least one arc. If no cycle is found, a change in the node prices is made, such that the labeling procedure now will result in more nodes being labeled. When no more progress with a particular arc can be made, the algorithm looks for another out-of-kilter arc. When all arcs are either in-kilter or in a condition such that no improvement

in the kilter number can be made, the algorithm terminates. Convergence of the algorithm is proved by showing that the number of consecutive labeling procedures that can be attempted without decreasing the total kilter number is bounded.

For example, arcs 5, 4, 3, and 7 in Fig. 1 form a cycle, with arc 7 traversed backwards. To indicate that arc 7 is traversed backwards, it will be denoted as arc -7. Thus, the above cycle is 5, 4, 3, -7. In order to define a cycle in algebraic terms, let $e_1, e_2, \ldots, e_n$ be numbers such that

$$e_j = \begin{cases} +1 & \text{if arc } j \text{ is traversed forward} \\ -1 & \text{if arc } j \text{ is traversed backward} \\ 0 & \text{if arc } j \text{ is not traversed .} \end{cases}$$

Then these arcs which are traversed form a cycle if

$$\sum_{j=1}^{n} a_{ij} e_j = 0 \qquad i=1,2,\ldots,m .$$

Moreover, this cycle is a <u>simple cycle</u> if no node appears as either a source node or a sink node on any arc in the

cycle more than twice,* i.e., if

$$\sum_{j=1}^{n} |a_{ij}\ e_j| \leq 2 \qquad i=1,2,\ldots,m\ .$$

Thus, it can be seen that any amount of flow may be pushed through a cycle without changing the node constraints. That is, nominality of the flow is preserved by replacing x_j by $x_j + Me_j$ for all j and M any constant.

For Fig. 1 the (a_{ij}) matrix has the form

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 1 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 \end{bmatrix} \begin{matrix} \text{node A} \\ \text{node B} \\ \text{node C} \\ \text{node D} \\ \text{node E} \end{matrix}$$

and for the above example the e_j's are 0, 0, 1, 1, 1, 0, -1, and 0.

For any arc j, we may denote its position in a <u>state diagram</u> (assuming $u_j > \ell_j$). Figure 2 shows the 15 possible combinations of x_j and $\bar{c}_j$ in relation to the capacities.

---

*A simple cycle may also be defined as a minimal dependent set of columns of A.

$\bar{c} < 0$ $x > u$ ($\alpha 5$)	$\bar{c} = 0$ $x > u$ ($\beta 5$)	$\bar{c} > 0$ $x > u$ ($\gamma 5$)
$\bar{c} < 0$ $x = u$ ($\alpha 4$)	$\bar{c} = 0$ $x = u$ ($\beta 4$)	$\bar{c} > 0$ $x = u$ ($\gamma 4$)
$\bar{c} < 0$ $\ell < x < u$ ($\alpha 3$)	$\bar{c} = 0$ $\ell < x < u$ ($\beta 3$)	$\bar{c} > 0$ $\ell < x < u$ ($\gamma 3$)
$\bar{c} < 0$ $x = \ell$ ($\alpha 2$)	$\bar{c} = 0$ $x = \ell$ ($\beta 2$)	$\bar{c} > 0$ $x = \ell$ ($\gamma 2$)
$\bar{c} < 0$ $x < \ell$ ($\alpha 1$)	$\bar{c} = 0$ $x < \ell$ ($\beta 1$)	$\bar{c} > 0$ $x < \ell$ ($\gamma 1$)

Fig. 2--State Diagram

Note that states $\alpha 4$, $\beta 4$, $\beta 3$, $\beta 2$, and $\gamma 2$ are "in-kilter," and that reversing an arc (reversing its direction, its upper and lower bounds, and changing the signs of $\bar{c}$, x, u, and ℓ) "reflects" its state through the center box $\beta 3$. The arcs in boxes $\alpha 3$, $\alpha 2$, $\alpha 1$, $\beta 1$, and $\gamma 1$ need to have their flows increased to bring them into kilter (see upward pointing arrows in Fig. 3); and their kilter numbers are precisely

the amount of flow increase needed to bring them into kilter. These arcs will be called sub-kilter arcs. Similarly, the arcs in states $\alpha 5$, $\beta 5$, $\gamma 5$, $\gamma 4$, and $\gamma 3$ must have their flows decreased by an amount equal to their kilter numbers in order to bring them into kilter. These arcs will be called super-kilter arcs.

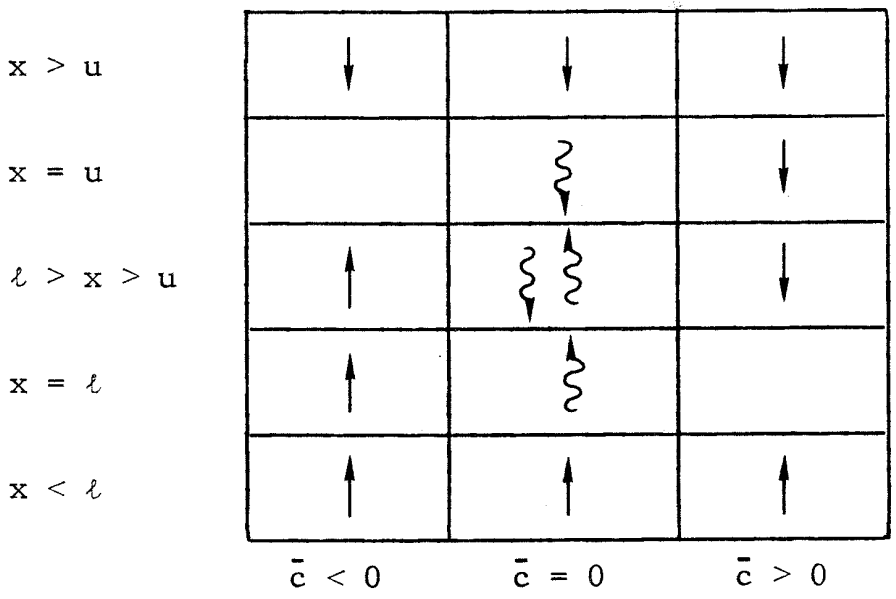

Fig. 3--Flow Changes that Decrease the Kilter Numbers (Indicated by straight arrows. Waved arrows indicate flow changes that do not change the kilter number.)

Moreover, an arc in state $\beta 2$ or $\beta 3$ may increase its flow at least one unit without increasing its kilter number and

without decreasing its kilter number since the kilter number is zero. These possible flow increases, together with possible decreases for the complementary states, are indicated by the waved arrows in Fig. 3.

Because of the symmetry of the state diagram, a rule for increasing the flow in a sub-kilter arc has its complementary rule in a super-kilter arc. For the sake of brevity, we assume that the arcs are directed whichever way is the most convenient for the discussion. Bear in mind that any arc may be reversed if desired.

LABELING PROCEDURE

The nodes are labeled with arc numbers. Begin with all node labels L(i) at zero. A node is unlabeled if its label is zero and it is labeled if its label is non-zero. Find an arc that is out-of-kilter, and assume that it is super-kilter (its flow must be decreased). This arc is the target arc, say arc J. Label its source node S[J] with the label J, i.e., set L(S[J]) = J. Now find any arc, j, where one node is labeled and one is not. Assume its source node, i, is labeled and its sink node, k, is not. If the flow may be increased without increasing the kilter number of the arc, then label the sink node of this arc

with the arc number; i.e., set L(k) = j. Continue looking for nodes to label until one of two occurrences:

1) The sink node T[J] of the target arc is labeled,

or

2) No more nodes can be labeled.

These two possibilities are known respectively as 1) breakthrough and 2) non-breakthrough.

BREAKTHROUGH

Since the sink node of the target arc was labeled, there is a path of arcs from the source node of the target arc to the sink node of the target arc, each of which may have its flow increased without increasing its kilter number. This path may be traced backward from the sink node of the target arc by means of the labels on the nodes to the source node of the target arc. Then calculate the maximum amount that the flow can be increased in the (non-target) arcs in the cycle (or decreased in the target arc) without increasing the kilter numbers of any of these arcs. Let this number be ϵ, and make this change in each arc of the cycle. The kilter number of the problem has now decreased by at least as much as the kilter-number change in the target arc. The kilter numbers for arcs not in the cycle are unchanged, and the kilter numbers of the arcs in the cycle are

$$k_j' = \max(0,\ k_j - \epsilon)$$

where k_j is the old kilter number and k_j' is the new kilter number for the arc j in the cycle. Since we are dealing with integers, ϵ is at least 1.

NON-BREAKTHROUGH

A non-breakthrough occurs if no more nodes can be labeled and the sink node of the target arc is not labeled. The nodes are then divided into two classes, labeled and unlabeled. The set of arcs that have one node labeled and one node unlabeled will be called the cut set for this labeling, since the arcs in the cut set cut off the set of labeled nodes from the set of unlabeled nodes. The target arc is a member of this cut set. Suppose that all arcs in the cut set have their source nodes labeled and their sink nodes unlabeled. (If an arc is labeled conversely, we may reverse it by the transformation mentioned above in Sec. I.) Then all of the arcs in the cut set have the property that their flows cannot be increased without increasing their kilter numbers. As shown in Fig. 3, these arcs must be in states $\alpha 5$, $\beta 5$, $\gamma 5$, $\alpha 4$, $\beta 4$, $\gamma 4$, $\gamma 3$, or $\gamma 2$. That is, the flow in each arc is either at or above its

upper bound or its reduced cost is positive and its flow is at or above its lower bound. These states are those in Fig. 4 for which arrows point to the left.

Note that if the prices of the nodes in the unlabeled set of nodes are increased by any constant, then only the arcs in the cut set will have their reduced cost changed, and this change in the reduced cost will be precisely the negative of the change in node prices of the unlabeled nodes. The leftward pointing arrows in Fig. 4 indicate the direction of state change that may occur by decreasing the reduced cost of the arcs in the cut set. Note that the arrows point toward states that have no greater kilter numbers than they do themselves.

Let Δ be the amount that the unlabeled nodes are going to have their prices raised, i.e., the amount that the reduced costs of that arc in the cut set will be decreased. Denote by Case 1 the situation that exists if there are any arcs in the cut set which are in states $\gamma 2$ or $\gamma 3$, i.e., with $x < u$ (shown shaded in Fig. 4). Case 2 occurs when all arcs in the cut set have flows at least equal to their upper capacities.

Consider first Case 1. It is clear that the value of Δ must not exceed the value of the reduced cost for any arc

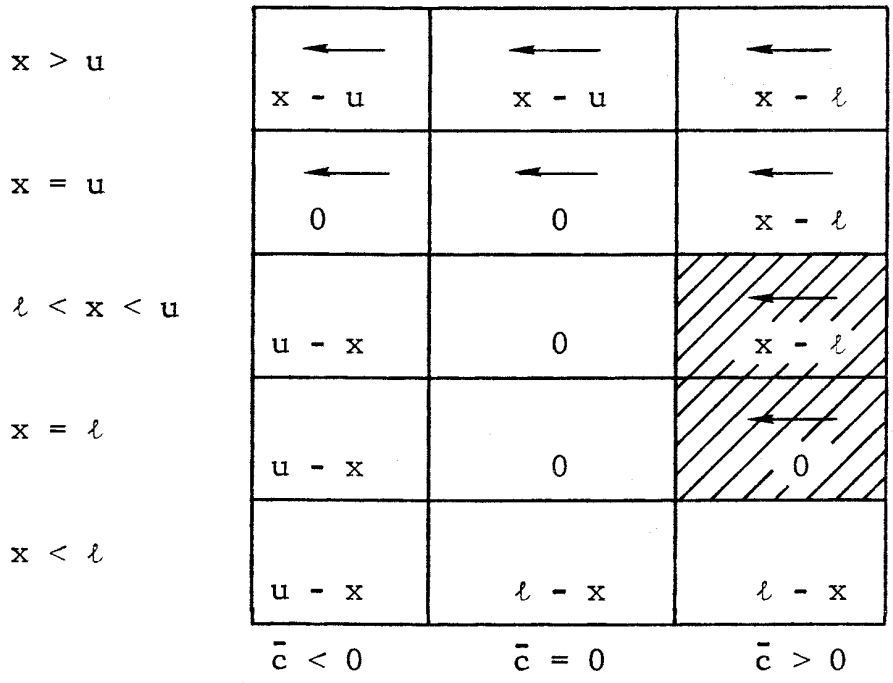

Fig. 4--Kilter Numbers of the Various States (Arrows show the states which may have their reduced costs lowered in a non-breakthrough cut set.)

in the cut set in state $\gamma 2$ or $\gamma 3$, but that the reduced costs of arcs in the cut set in other states may be decreased by any amount without increasing their kilter numbers. Denote as the critical arc that (or one of the) arc(s) in state $\gamma 2$ or $\gamma 3$ with the lowest reduced cost. Let Δ be the reduced cost of the critical arc and increase the node prices of all unlabeled nodes by Δ. If the critical arc is in state $\gamma 2$, its kilter number will remain at zero.

Now consider Case 2. All arcs in the cut set have flows at least equaling their upper capacities. Let the critical arc be the arc with the maximum reduced cost. If the critical arc has a positive reduced cost, let Δ be this number and raise the prices of the unlabeled nodes by Δ. This causes every arc that had a positive reduced cost to now have a non-positive reduced cost and to have its kilter number reduced from $x - \ell$ to $x - u$. On the other hand, if the critical arc has a non-positive reduced cost, then all arcs in the cut set have a non-positive reduced cost; and the kilter number of the target arc cannot be reduced. Hence, the problem is infeasible. The cut set is then a cut in the classical sense in that the target arc has a flow above its upper capacity and the only way to reduce the flow in the target arc is to increase the flow in some other arc(s) in the cut set which has a flow at least equaling its upper capacities. If the flow in the target arc exceeds its upper capacity, the problem is infeasible (in Case 2) regardless of the sign of the reduced cost of the critical arc, but some improvement of the kilter numbers occurs if the reduced cost of the critical arc is positive.

In any event, for Case 2, after the node prices have been changed, the kilter number of the target arc has been minimized (to zero if its flow was at the upper capacity) and no more labelings with this arc as the target arc should be attempted. In Case 1, the labeling procedures should now be continued, keeping the labels intact for the nodes already labeled. At least one more node can be labeled, in particular, the sink node of the critical arc. Hence, since there are m nodes, at most m - 1 consecutive Case 1 non-breakthroughs can occur with the same target arc. If the problem is feasible, the kilter number for the problem will be decreased after at most m labelings. Hence, it is seen that the algorithm converges.

III. RELATED ALGORITHMS

THE INFEASIBILITY ALGORITHM

For each arc, j, of the network, define q_j, the infeasibility number:

$$q_j = \max(0,\ x_j - u_j,\ \ell_j - x_j) \ ,$$

and Q, the total infeasibility:

$$Q = \sum_{j=1}^{n} q_j \ .$$

It is clear that q_j is the amount (if any) that the flow violates the upper or lower capacities imposed on the arc j. It is also clear that the kilter number, K, of the problem can be reduced to the value of Q, and that if Q is at its minimum value, then K can be reduced to its minimum value (which is K = Q) using the out-of-kilter algorithm. If Q is not at its minimum value, and if the problem is infeasible, then K may not be reduced to its minimum value by the out-of-kilter algorithm. For example, the network of Fig. 5 has initially zero flow in each of its three arcs, the upper capacity is equal to the lower

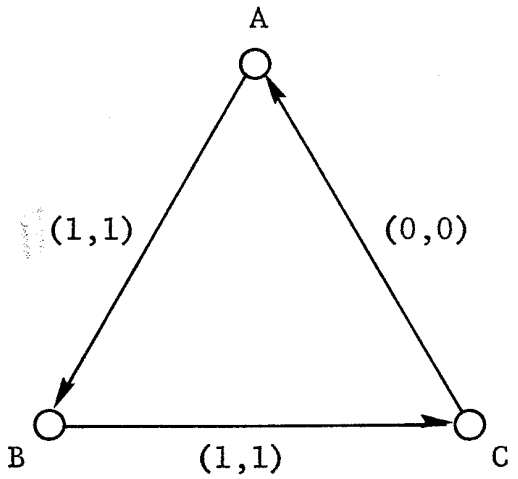

Fig. 5--Example Network (Numbers in parenthesis are the lower and upper capacities.)

capacity for each arc, and the capacity for arc (A,B) is 1, for arc (B,C) is 1, and for arc (C,A) is 0. Arc (C,A) is "in-kilter," while the others are not; but any change in the flow will cause arc (C,A) to go "out-of-kilter," hence, nothing can be done to this network by the out-of-kilter algorithm. Thus, K remains at 2, despite the fact that the minimum K is 1. This minimum is obtained by forcing a flow of 1 into each arc, thereby increasing the kilter number (which in this case is also the infeasibility number) of arc (C,A) to 1 and decreasing the kilter numbers of the other two arcs to zero. The attainment of the

minimum K can also depend on the order in which arcs are chosen to be brought into kilter and on the order in which nodes are labeled. The network in Fig. 6 is an example in which either of these possibilities can occur.

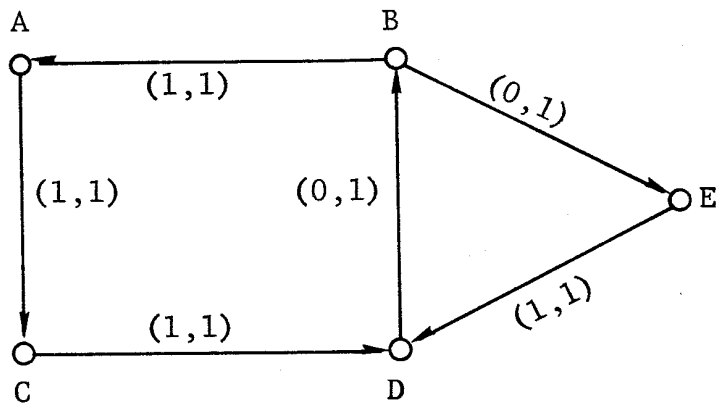

Fig. 6--Another Example of an Infeasible Network

An algorithm will now be developed to solve the problem of minimizing Q for a network. In order to develop this algorithm, we use the artifice of replacing each arc of the network by three new arcs. Suppose an arc has source node A, sink node B, lower capacity ℓ, upper capacity u, and nominal flow x. The three arcs that replace this arc (see Fig. 7), each have source node A and

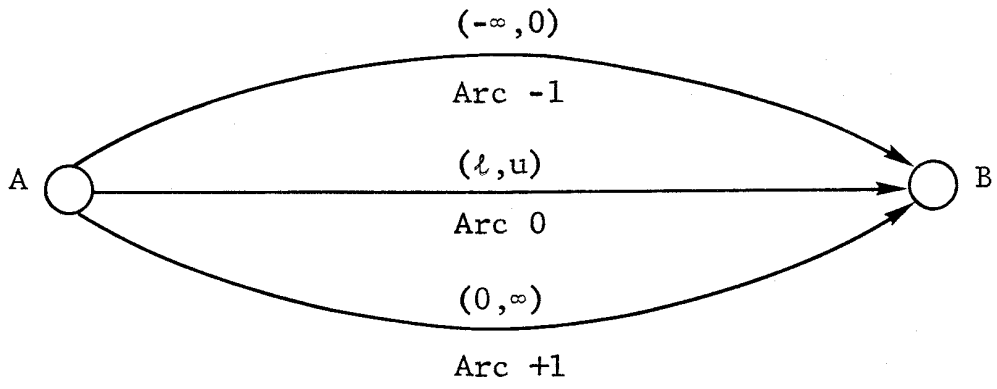

Fig. 7--Artificial Arcs Used in Infeasibility Algorithm

sink node B, and are denoted as arc -1, arc 0, and arc +1. These arcs have the following properties:

Arc	Unit Cost	Lower Capacity	Upper Capacity	Flow
arc -1	$c^- = -1$	$\ell^- = -\infty$	$u^- = 0$	$x^- = \min(0, x - \ell)$
arc 0	$c^0 = 0$	$\ell^0 = \ell$	$u^0 = u$	$x^0 = \max(\ell, \min(u,x))$
arc +1	$c^+ = +1$	$\ell^+ = 0$	$u^+ = +\infty$	$x^+ = \max(0, x - u)$

Note that $x = x^- + x^0 + x^+$, and that the flows are feasible in each arc. Hence, this new problem is feasible and its total cost is precisely Q of the original problem.

Let γ^-, γ, and γ^+ be the reduced costs for arcs -1, 0, and +1, respectively, so that

$$\gamma^- = -1 + \pi_A - \pi_B = \gamma - 1$$

$$\gamma = 0 + \pi_A - \pi_B$$

$$\gamma^+ = +1 + \pi_A - \pi_B = \gamma + 1 \ .$$

It is not necessary to deal with node prices for this algorithm, but only with the reduced costs γ, which for each arc are initially zero. Hence $\gamma^- = -1$ and $\gamma^+ = +1$ initially. γ^+ must remain non-negative, since it could become negative only if the flow in arc +1 were at its upper bound, which would imply that Q is infinite. Similarly, γ^- must remain non-positive. Thus,

$$\gamma^- = \gamma - 1 \leq 0$$

$$\gamma^+ = \gamma + 1 \geq 0,$$

or

$$\gamma \leq 1$$

$$\gamma \geq -1 \ ,$$

i.e., $|\gamma| \leq 1$. But γ will take on only integral values; hence, γ may only be 0 or ± 1. Arc -1 will have non-zero (negative) flow only if $x < \ell$, and its flow may be decreased (made more negative) only if $\gamma^- \geq 0$, i.e., (since γ^- must be non-positive) $\gamma^- = 0$ or $\gamma = 1$. Similarly, the flow may be increased in arc +1 only if $\gamma = -1$; and arc +1 will have non-zero flow only if $x > u$.

The above discussion shows that if node A is labeled then node B can be labeled:

1) via arc -1 if $x < \ell$;

2) via arc 0 if $x < u$ and $\gamma \neq 1$;

3) via arc +1 if $\gamma = -1$.

The state diagram of this composite arc is shown in Fig. 8. Note that the states that are impossible for $\gamma \neq 0$ are crossed out. If a non-breakthrough occurs, then for each composite arc that has its source node labeled and its sink node unlabeled, either $\{\gamma = 0$ and $x \geq u\}$ or $\{\gamma = +1$ and $x = \ell\}$. Increasing the node prices (which are not being computed) of the unlabeled nodes by +1 is equivalent to decreasing γ of this composite arc by 1 (or increasing γ by 1 if the arc is labeled conversely). This moves the state of the arc one box to the left, and hence to a labelable state (unless $x = \ell = u$). Moreover, this

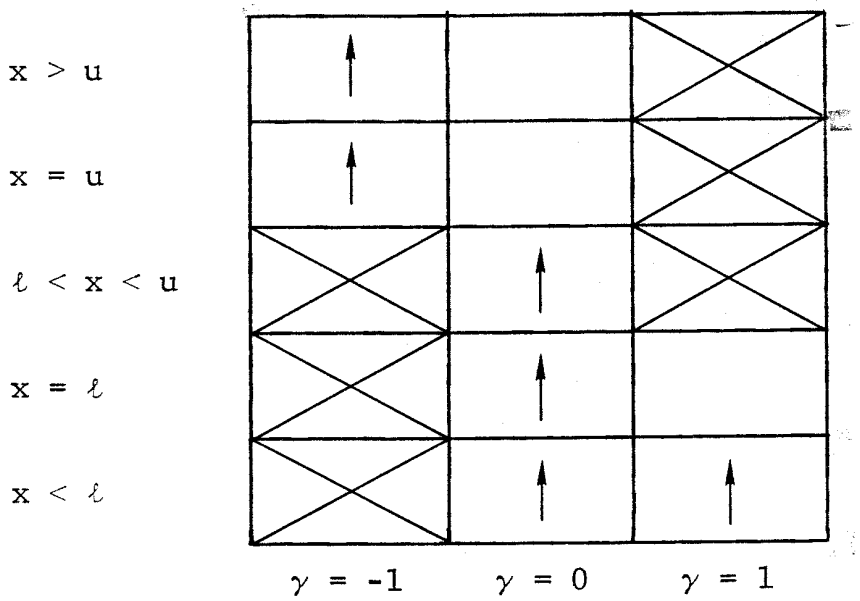

Fig. 8--Arc States in which Flow may be Increased in the Infeasibility Algorithm

change of the γ's puts the target arc into kilter, since a composite arc is in kilter if either $\gamma \neq 0$ or $\ell \leq x \leq u$.

Hence, the problem of minimizing Q may be solved by the out-of-kilter algorithm by replacing each arc by three arcs with appropriate bounds and costs. But these composite arcs were merely an artifice used for determining a new algorithm based on the out-of-kilter method. Now, discarding the composite arcs, this method may be summarized.

Begin with numbers $\gamma_1,\gamma_2,\ldots,\gamma_n$ all zero. Look for an arc, J, with $\gamma_J = 0$, and either $x_J > u_j$ or $x_J < \ell_J$. If no such arc exists, Q has been minimized and we are done. When an arc is found with these properties, label its source node if $x_J > u_J$ and its sink node if $x_J < \ell_J$. Then begin a labeling procedure that terminates if the other node of arc J is labeled or if no more nodes can be labeled.

If arc j has its source node labeled and its sink node unlabeled, the sink node can be labeled if

1) $\gamma_j = 0$ and $x_j < u_j$,

or

2) $\gamma_j = -1$,

or

3) $x_j < \ell_j$.

If the sink node is labeled and the source node unlabeled, the source node can be labeled if

1) $\gamma_j = 0$ and $x_j > \ell_j$,

or

2) $\gamma_j = +1$,

or

3) $x_j > u_j$.

When the labeling procedure results in a breakthrough (i.e., both nodes of arc J have been labeled), the cycle is determined, and ϵ is calculated and added to those arcs labeled (forward) from source to sink and subtracted from those arcs labeled (backward) from sink to source. ϵ is the minimum of

1) $u_j - x_j$ if $\gamma_j = 0$ and arc j labeled forward,
2) $x_j - \ell_j$ if $\gamma_j = 0$ and arc j labeled backward,
3) $\ell_j - x_j$ if $\gamma_j = +1$ and arc j labeled forward,
4) $x_j - u_j$ if $\gamma_j = -1$ and arc j labeled backward

for all arcs in the cycle. If an arc, j, is labeled forward and $\gamma_j = -1$ or labeled backward and $\gamma_j = +1$, then this arc imposes no limit on ϵ. All arcs with $\gamma_j = 0$ do impose a limit, and arc J is one of these. After the ϵ change in the cycle, x_J may or may not satisfy $x_J \geq \ell_J$ and $x_J \leq u_J$. If it does satisfy these conditions, then look for another arc to label. If it does not satisfy these conditions, begin the labeling procedure anew with arc J.

When the labeling procedure results in a non-breakthrough, subtract 1 from γ_j of all arcs with the source node labeled and the sink node unlabeled, and add 1 to γ_j for all arcs with the source node unlabeled and the sink

node labeled. The other arcs do not have a change made in γ_j. In particular, arc J has one node labeled and one unlabeled; hence, now $\gamma_J = \pm 1$. Thus, another arc must be found to begin the labeling procedure again.

Eventually, every arc will have either $\gamma_j \neq 0$ or $\ell_j \leq x_j \leq u_j$. The infeasibility of the flow has then been minimized. If $\gamma_j \neq 0$, arc j is not necessarily infeasible, but all arcs with non-zero γ can be thought of as part of a cut set for all of the infeasible arcs.

THE FEASIBILITY ALGORITHM

The feasibility algorithm is similar to the infeasibility algorithm, except that one is not interested in minimizing the infeasibility but only in finding a feasible flow if it exists. If the problem is feasible, no non-breakthroughs will occur in the infeasibility algorithm. Hence, the reduced costs, γ, will never be made non-zero.

The labeling procedure for the feasibility algorithm is as follows. If the source node of arc j is labeled and the sink node is unlabeled, then the sink node can be labeled if $x_j < u_j$. If the sink node is labeled and the source node is unlabeled, then the source node can be labeled if $x_j > \ell_j$. Only arcs which are infeasible are chosen as target arcs. If a non-breakthrough occurs, the

problem is infeasible. Otherwise, only breakthroughs will occur, and the problem will be feasible when all arcs are made feasible. For this algorithm, no γ_j are calculated. Hence, it is simpler computationally than the infeasibility algorithm.

TWO-PHASE ALGORITHMS

Either the infeasibility algorithm or the feasibility algorithm may be used as the first phase of a two-phase algorithm. Then the second phase is the out-of-kilter algorithm described in Sec. II. If phase 1 causes the network to be feasible, certain tests in the out-of-kilter algorithm become unnecessary. In the labeling procedure, if the source node of arc j is labeled and the sink node is unlabeled, the sink node can be labeled if $c_j \leq 0$ and $x_j < u_j$. The additional case, that $x_j < \ell_j$ and $c_j > 0$, cannot occur in this algorithm and need not be tested. Similarly, in the non-breakthrough procedure, the tests for $x > u$ can be omitted.

IV. COMPUTATIONAL METHODS

LABELING

The method of labeling nodes described in Sec. II was that of looking for an arc with one node labeled and one node unlabeled and then determining whether the unlabeled node could be labeled. A method has evolved that requires less searching for arcs with exactly one node labeled. The idea of this method is to set up what is called a scan list. Associated with the scan list are two indices: s, the length of the scan list, and p, the position of the scanner. Denote the scan list itself by R(1),R(2),...,R(s). The procedure begins with p = 1, s = 1, and R(1) = node which is the labeling origin. Then look at (i.e., scan) each arc for which the node R(p) is its source node or its sink node. If the other node of any of these arcs can be labeled, do so, increase s by 1 and set R(s) to the node just labeled. If R(s) is the terminal node, then the labeling procedure is done and a breakthrough has occurred. When all of the arcs joining node R(p) have been scanned and no breakthrough has occurred, increase p by 1, then repeat the process for the new node R(p). If $p > s$, then the scan list has been exhausted and a non-breakthrough has occurred.

When this procedure is used, each arc can be "looked at" at most twice: once from each of its nodes. But, in order for this method to be more efficient than one that merely searches for arcs with one node labeled, lists must be set up of the arcs that join each particular node.

LIST STRUCTURE

Let there be n arcs such that the j^{th} arc has source node S[j] and sink node T[j]. Suppose that there are m nodes numbered from 1 through m and, therefore, that S and T have values in this range. Assuming that the arcs are in no particular order, it is necessary to set up four lists, say U, V, g, and h, where U and V are arrays of length m+1 and g and h are of length n. Let σ_i be the number of arcs that have node i as their source node and let τ_i be the number of arcs that have node i as their sink node. Either σ_i or τ_i, but not both, may be zero. Then U and V are defined recursively:

$$U_1 = 1$$

$$U_{i+1} = U_i + \sigma_i \qquad i=1,2,\ldots,m$$

and

$$V_1 = 1$$

$$V_{i+1} = V_i + \tau_i \qquad i=1,2,\ldots,m \ .$$

Now, let $\rho_i = U_i$ for i=1,2,...,m. For each j from 1 through n, let i = S[j], $g[\rho_i] = j$, and then increase ρ_i by 1. When this is done, $g[U_i]$ through $g[U_{i+1} -1]$ is a list of the arcs with node i as the source node. The same procedure is repeated for V and h with the sink nodes, giving a list of the arcs with the same sink node.

The ALGOL procedure in the Appendix uses the above lists. This procedure executes the out-of-kilter algorithm as described in Sec. II. This may be compared with the simpler program in Ref. 4. The symbols used in this program are substantially the same as the ones given at the beginning (p. xi) and used throughout this Memorandum.

Space may be saved by arranging the arcs so that the source nodes are in order. Then the list g is unnecessary since g(j) = j. This procedure is used in the FORTRAN program of Ref. 5.

More complicated list structures may be needed if this procedure must store data on such peripheral devices as disks. In this event, it may be useful to double each

arc so that it has its forward and backward representations in the lists. Then only one "disk file" need be retrieved for scanning each node.

TENTATIVE CONCLUSIONS

Several experiments were done with the methods described above. Since these experiments were not at all extensive, and since the results depend greatly on the type of problem and machine software, these conclusions should not be regarded as final.

Several problems were solved using the algorithm described in Ref. 4 and the algorithm given in the Appendix. The largest problem solved had 1530 arcs. This problem was run on a relatively slow machine (IBM 360 Model 40) with the result that the program with the list structure described above ran five times as fast as the program without this list structure (30 min vs. 2.5 hr). This time ratio should increase on larger problems and decrease on smaller problems. The faster problem solution using the program with the list structure must be balanced against the greater storage capacity needed for the lists.

Several tests were made employing a two-phase algorithm, with the first phase being the feasibility algorithm described in Sec. III. This modification increased the

number of calculations, and hence cannot be recommended. In fact, Fulkerson's [3] original description of the algorithm seems to be the most efficient, even though certain Case 2 non-breakthrough calculations were not made. Therefore, some of these calculations are not included in the appended ALGOL procedure.

Appendix

ALGOL PROCEDURE

This ALGOL procedure should be self-explanatory since it uses the symbols appearing in the body of this Memorandum. Upper and lower case symbols are distinct. The symbol "pi" corresponds to the symbol π in the text, "outkilter" is the number of arcs that could not be brought into kilter by the procedure, and "refnode" is an arbitrary node whose π value is not changed by the procedure. This procedure was not checked in the ALGOL language; hence, all errors may not have been detected.

```
  Procedure network (m,n,S,T,c,u,ℓ,x,pi,refnode,outkilter)
  integer m,n,refnode,outkilter;
  integer array S,T,c,u,ℓ,x,pi;
  begin integer array U,V[1:m+2],g,h[1:n],L,R[1:m];
  integer J,aa,term,laborg,origin,i,j,p,k,s,a,kp,Kq,eps,eps1;
  Boolean breakthru;
  outkilter:= 0;
  go to setup;
endsetup:  for  j:= 1  step 1 until n do
           c(j):= c(j)+pi(S[j])-pi(T[j]);
```

```
  comment  look for an out-of-kilter arc;
search:  J:= 1; aa:= 0; breakthru:= true;
mainlp:  if x(J)<ℓ(J)∨c(J)<0∧x(J)<u(J) then go to fd;
         if x(J)>u(J)∨c(J)>0∧x(J)>ℓ(J) then go to bd;
return:  J:= J+1;
         if J≤n then go to mainlp;
         for j:= 1 step 1 until n do
         c(j):= c(j)-pi(S[j])+pi(T[j]); go to endn;
fd:  term:= S[J]; origin:= T[J]; laborg:= J;
  go to prelab;
bd:  term:= T[J]; origin:= S[J]; laborg:= -J;
prelab:  R(1):= origin; go to label;
  comment  count arcs beginning and ending at nodes;
setup:  for i:= 3 step 1 until m + 2 do
    begin U(i):= 0; V(i):= 0 end;
  for j:= 1 step 1 until n do
    begin  U(S[j] + 2):= U(S[j] + 2) + 1;
           V(T[j] + 2):= V(T[j] + 2) + 1 end;
  comment  cumulate counts;
    U(1):= 1; U(2):= 1;
    V(1):= 1; V(2):= 1;
  for i:= 3 step 1 until m + 1 do
```

```
    begin  U(i):= U(i) + U(i-1);
           V(i):= V(i) + V(i-1) end;
  comment  set up arc locator lists;
  for j := 1 step 1 until n do
    begin  g[U(S[j] + 1)]:= j;
           h[V(T[j] + 1)]:= j;
           U(S[j] + 1):= U(S[j] + 1) + 1;
           V(T[j] + 1):= V(T[j] + 1) + 1
    end;
  go to endsetup;
label1:  if ¬breakthru∧J=aa then go to label2;
  comment  zero out labels;
  for i:= 1 step 1 until n do L(i): = 0;
    s:= 1;
label2: p:= 1; aa:= J; breakthru:= false; L(origin):= laborg;
  comment  try to label the forward arcs;
label3: i:= R(p);
  for a:= U(i) step 1 until U(i+1)-1 do begin
    j:= g[a]; k:= T[j];
    if L(k)=0∧[x(j)<ℓ(j)∨c(j)≤0∧x(j)<u(j)] then
      begin L(k):= j; s:= s+1; R(s):= k end; end;
  comment  try to label the backward arcs;
```

```
for a:= V(i) step 1 until V(i+1)-1 do begin
  j:= h[a]; k:= S[j];
  if L(k)=0∧[x(j)>u(j)∨c(j)≥0∧x(j)>ℓ(j)] then
    begin L(k):= -j; s:= s+1; R(s):= k end; end;
comment test for terminal labeled;
if L(term)≠0 then go to break;
  p:= p+1;
comment if scan list exhausted, non-breakthru;
  if p>s then go to nobreak;
go to label3;
comment find flow increment in cycle;
break: eps:= 999999999; breakthru:= true;
    Kt:= term;
    j:= 1
breakloop: Kq:= L(Kt); kp:= abs(Kq);
    if Kq>0 then go to forwardbreak; Kt:= T(kp);
    if c(kp)≥0 then go to lowerbreak;
    go to upperbreak;
forwardbreak: Kt:= S[kp]
    if c(kp)>0 then go to lowerbreak;
upperbreak: eps:= min(eps,abs(u(kp)-x(kp)));
    go to endbreakloop;
```

```
lowerbreak:  eps:= min(eps,abs(ℓ(kp)-x(kp)));
endbreakloop:  R(j):= Kq;
    if Kt = term then go to increment;
    j:= j+1; go to breakloop;
  comment  increment flow;
increment:  for i:= 1 step 1 until j do
    if R(i)>0 then
    x(R(i)):= x(R(i)) + eps
      else x(-R(i)):= x(-R(i)) - eps;
    go to mainlp;
  comment  find delta for non-breakthru;
nobreak:  eps1:= 999999999;
    for j:= 1 step 1 until n do
    if L(S[j])≠0∧L(T[j])=0∧x(j)<u(j)
      ∨L(S[j])=0∧L(T[j])≠0∧x(j)>ℓ(j)
    then eps1:= min(eps,abs(c(j)));
  comment  test for case 2;
    eps:= eps1;
    if eps≠999999999 then go to change;
    if c(J)=0∨sign(L(origin))=sign(c(J))
      then go to infeas;
    eps:= abs(c(J));
```

```
  comment  change reduced costs;
change:  for j:= 1 step 1 until n do
  if L(S[j])=0∧L(T[j])≠0
    then c(j):= c(j) + eps else
  if L(S[j])≠0∧L(T[j])=0
    then c(j):= c(j) - eps;
  comment  change node prices;
    if L(refnode)≠0 then
      for i:= 1 step 1 until m do begin
         if L(i)=0 then pi(i):= pi(i) + eps end else
      for i:= 1 step 1 until m do
         if L(i)≠0 then pi(i):= pi(i) - eps;
         if eps=eps1∨x(J)=ℓ(J)∨x(J)=u(J) then
           go to mainlp;
infeas:  outkilter:= outkilter + 1; go to return;
endn:  end network
```

REFERENCES

1. Ford, L. R., and D. R. Fulkerson, Flows in Networks, Princeton University Press, 1962.

2. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, 1963.

3. Fulkerson, D. R., "An Out-of-Kilter Method for Minimum Cost Flow Problems," Journal SIAM, Vol. 9, No. 1 (March 1961), pp. 18-27.

4. Briggs, W. A., "Netflow," Comm. ACM, Vol. 8, No. 2, (February 1965), p. 103.

5. "Out of Kilter Network Routine," SHARE Distribution 3536, SHARE Distribution Agency, Hawthorne, New York, 1967.