AD-752 758

AN IMPROVED VERSION OF THE OUT-OF-KILTER METHOD AND A COMPARATIVE STUDY OF COMPUTER CODES

R. S. Barr, et al

Texas University

Prepared for:

Office of Naval Research

November 1972

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

56

Research Report
C.S. 102

AN IMPROVED VERSION OF THE OUT-OF-KILTER METHOD AND A COMPARATIVE STUDY OF COMPUTER CODES

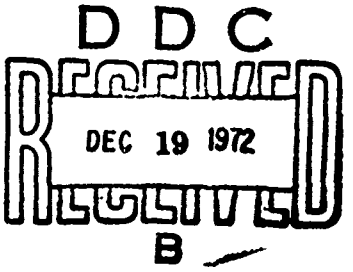
by

R.S. Barr

F. Glover*

D. Klingman

November 1972



*University of Colorado, Boulder

This research was partly supported by a grant from the Farah Foundation and by ONR Contracts N00014-67-A-0126-0008 and N00014-67-A-0126-0009 with the Center for Cybernetic Studies, The University of Texas. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 512
The University of Texas
Austin, Texas 78712

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)

Center for Cybernetic Studies
University of Texas

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b GROUP

3 REPORT TITLE

An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes

4 DESCRIPTIVE NOTES (Type of report and, inclusive dates)

5 AUTHOR(S) (First name, middle initial, last name)

R.S. Barr D. Klingman
F. Glover

6 REPORT DATE	7a. TOTAL NO OF PAGES	7b. NO OF REFS
November 1972	54	31

8a. CONTRACT OR GRANT NO	9a. ORIGINATOR'S REPORT NUMBER(S)
NR-047-021	Center for Cybernetic Studies Research Report Number 102
b. PROJECT NO	
c. N00014-67-A-0126-0008	9b OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d. N00014-67-A-0126-0009	

10 DISTRIBUTION STATEMENT

This document has been approved for public release and sale: its distribution is unlimited.

11 SUPPLEMENTARY NOTES	12 SPONSORING MILITARY ACTIVITY
	Office of Naval Research (Code 434) Washington, D.C.

13 ABSTRACT

The primary objectives of this paper are fourfold: (1) to present an improved formulation of the out-of-kilter algorithm; (2) to give the results of an extensive computational comparison of a code based on this formulation with three widely-used out-of-kilter production codes; (3) to study the possible sensitivity of these programs to the type of problem being solved; and (4) to investigate the effect of advance dual start procedures on overall solution time.

The study discloses that the new formulation does indeed provide the most efficient solution procedure of those tested. The resulting method has been found to be at least 100 times faster than the state of the art large scale LP code, OPHELIE/LP. Further, this streamlined version of out-of-kilter was found to be faster than the other out-of-kilter codes tested (SHARE, Boeing and Texas Water Development Board codes) by a factor of 2-5 on small and medium size problems and by a factor of 4-15 on large problems. The streamlined method's median solution time for 1500 node networks on a CDC 6600 computer is 33 seconds with a range of 33 to 35 seconds.

Some of the major characteristics of this study are (1) all of the solution techniques are tested on the same machine and the same problems; (2) a broad spectrum of problem sizes is examined, networks varying from 50 nodes to 1500 nodes and transportation problems varying from 10 x 10 to 150 x 150; (3) a broad profile of densities is examined ranging from 90% to 0.25%; (4) capacitated problems are examined in detail; and (5) the effect of topological changes in the network structure are studied.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

I

Unclassified

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Network Times						
Transportation Times						
Primal-Dual Method						
Out-of-Kilter Method						
Linear Programming Times						
Capacitated Network Problems						
Transportation Problems						
Network Problems						

ABSTRACT

The primary objectives of this paper are fourfold: (1) to present an improved formulation of the out-of-kilter algorithm; (2) to give the results of an extensive computational comparison of a code based on this formulation with three widely-used out-of-kilter production codes; (3) to study the possible sensitivity of these programs to the type of problem being solved; and (4) to investigate the effect of advance dual start procedures on overall solution time.

The study discloses that the new formulation does indeed provide the most efficient solution procedure of those tested. The resulting method has been found to be at least 100 times faster than the state of the art large scale LP code, OPHELIE/LP. Further, this streamlined version of out-of-kilter was found to be faster than the other out-of-kilter codes tested (SHARE, Boeing and Texas Water Development Board codes) by a factor of 2-5 on small and medium size problems and by a factor of 4-15 on large problems. The streamlined method's median solution time for 1500 node networks on a CDC 6600 computer is 33 seconds with a range of 33 to 35 seconds.

Some of the major characteristics of this study are (1) all of the solution techniques are tested on the same machine and the same problems; (2) a broad spectrum of problem sizes is examined, networks varying from 50 nodes to 1500 nodes and transportation problems varying from 10 x 10 to 150 x 150; (3) a broad profile of densities is examined ranging from 90% to 0.25%; (4) capacitated problems are examined in detail; and (5) the effect of topological changes in the network structure are studied.

Included in our results are the findings that: (1) capacitation of the arcs has an erratic effect on solution times; (2) advance dual start methods are inconsistent in affording computational advantages; and (3) changes in network topology have an erratic effect on the standard codes, but not on the code based on the new formulation.

IV

1.0 INTRODUCTION AND SCOPE OF THE COMPUTATIONAL ANALYSIS

The major thrusts of this paper are: to present an improved formulation of Fulkerson's out-of-kilter algorithm which leads to more efficient computer implementation; to perform an in-depth computational comparison of a code based on this formulation with three widely used out-of-kilter production codes. The first of these thrusts is present in section 2.0 and the second in section 3.0.

Developing improved methods for solving network problems and cataloging the relative efficiencies of alternative approaches is of considerable practical importance. Evidence of this is provided both by the fact that a sizeable proportion of the linear programming literature is devoted to network problems, and by the fact that an even larger share of the many concrete industrial and military applications of linear programming deal with such problems. Network problems often occur as subproblems in a larger problem (e.g., the traveling salesman or the plant location problem). Moreover, industrial applications of network problems often contain thousands of variables, and hence a streamlined algorithm is not only computationally worthwhile but a practical necessity.

Our interest in studying computer implementations of the out-of-kilter method was stimulated by the computational study of [15] which found that the SHARE out-of-kilter code was 6 times slower for solving transportation problems that a special purpose primal transportation code (embodying recently developed methods for accelerating the determination of basis trees and dual evaluators [17]). This finding which is contrary to network folklore, leads to the question of whether it might be possible to improve the out-of-kilter method. The study of this paper shows that such improve-

ment is indeed possible. It should be stressed that the techniques underlying this improvement, while oriented toward more efficient computer implementation, are refinements in methodology and not refinements in computer programming. They do not take advantage of any particular computer configuration, and have been coded in a "manilla" Fortran IV to permit implementation on a wide range of machines.

In testing the new method, we sought also to address several issues of general interest. Our study also examines:

1. The computational efficiency provided by special purpose algorithms for solving transportation and network problems (coded in FORTRAN) as opposed to general purpose linear programming codes (coded in a machine language) that use sophisticated procedures for exploiting sparse matrices.

2. The computational efficiency provided by different existing implementations of the out-of-kilter algorithm.

3. The effect of using advance dual procedures in conjunction with out-of-kilter codes on total solution time.

4. The effect of problem density on solution time, where density is equal to the number of admissible arcs in the problem divided by the number of total possible arcs. (This effect is of substantial practical relevance since most large "real world" network problems are quite sparse.)

5. The effect of altered topological structure on solution time. (To determine this effect, two network classes are examined and are called Type I and Type II networks. Type I networks have a path from each source to every sink and, in addition, a direct path from each transhipment node to each sink. Type II networks do not have paths between all source and sink combinations, but may have several complex paths between particular source and sink pairs. Besides these network structures, the classical bipartite

network of the transportation problem is examined. Structural variants of these networks are also examined which arise by varying the number of sources, sinks, transhipment sources and sinks, and the total number of nodes.)

6. The effect of problem size on solution time. (All codes were tested on problems varying from 50 nodes to 1500 nodes.)

7. The effect of arc capacitation, as reflected in the influence on computation time of the percentage of the arcs which are capacitated, and of different capacity ranges.

To accomplish these purposes it has been necessary to obtain computer codes that are representative of the "state of the art" (in addition to developing the code based on the new formulation). The codes we obtained represent what we understand to be the most efficient of those available, and have been widely used in current industrial applications (in fact world-wide) over the past 3-5 years.

The four out-of-kilter codes which we tested are those of SHARE, Boeing, the Texas Water Development Board (TWB), and our own. The SHARE code was written by R.J. Classen of the RAND Corporation and is available for general distribution [3,26]. The Boeing code, which was obtained through Chris Witzgall, was developed at the Boeing research laboratories. The Texas Water Development Board code (henceforth referred to as the TWB code) was developed by Paul Jensen at the University of Texas, drawing on the proposal of [2]. This code was later modified slightly by the Texas Water Development Board to enhance its computational efficiency on small problems. The fourth code, which was developed by the authors, is referred to as SUPERK.

All of these codes are in-core codes; i.e., the program and all of the problem data simultaneously reside in fast-access memory. They are all

coded in FORTRAN and none of them have been tuned (optimized) for a particular compiler. All of the problems were solved on the CDC 6600 (which has a maximum memory of 130,000 words) at the University of Texas Computation Center using the RUN compiler.[1] The computer jobs were executed during periods when the machine load was approximately the same, and all solution times are exclusive of input and output; i.e., the toal time spent solving the problem was recorded by calling a Real Time Clock upon starting to solve the problem and again when the solution was obtained.

The computational comparison involved 55 transportation and 160 network problems which were randomly generated using three different generators. All problems were restricted to less than 10,000 arcs and all the cost coefficients were restricted to lie between 1 and 100. The assignment of the supply and demand quantities depended on the generator being used. The transportation problems were all square (i.e., with the same number of origins and destinations) and the total supply of each m x m transportation problem was set equal to 1000 m. The supply and demand quantities for each node were picked using a uniform probability distribution over the interval from 0 to 2000. Both of the network generators set the total supply at 1000 times the number of nodes in the network. The generator used to create the type I networks [23] distributed the total supply randomly among the sources by dividing the total supply by the number of sources, then picking supply amounts for each source using a uniform probability distribution over the interval between 0 and twice this number. The demands were determined similarly. The generator used to create the type II networks [20] determined its supplies by assigning to each source node in sequence a quantity between 0 and half the net (unassigned) total supply, using a uniform probability distribution. The total demand of a sink node was

determined by accumulating randomly assigned amounts of the supply associated with sources connected to the sink. Of the 160 networks solved, 65 were type I networks and 92 were type II networks, 72 of which were capacitated. These last 72 networks were solved twice; once with and once without the capacities. Additional parameter characteristics of the networks are discussed in section 3.0.

2.0 REFORMULATION OF OUT-OF-KILTER ALGORITHM.

The out-of-kilter algorithm [8,12] defines certain "kilter" conditions which, taken together, constitute primal and dual feasibility criteria for arcs in a network. The method brings each non-conforming ("out-of-kilter") arc into kilter by adjusting its flow or changing its node potentials. In order to accomplish this,a labeling procedure is used which, after one or more applications, identifies a loop containing the non-conforming arc.

In our reformulation all of the above processes are redesigned and the network representation is altered to allow more efficient processing. Nevertheless, the basic logic of the original method is maintained. The reformulation modifies the labeling procedure in a manner which causes it to process less information on the forward pass and more information on the reverse pass. Net computational savings result because the reverse pass typically involves only a portion of the nodes encountered on the forward pass, and sometimes the reverse pass is not executed at all.

In addition to the new labeling scheme, the reformulated algorithm employs a special classification scheme for determining the "kilter status" of each arc. This scheme permits the current net capacity and marginal cost of the arc to be evaluated with increased efficiency, which in turn expedites the determination both of the flow augmenting path, when it

exists, and of the new marginal cost assignment (via implicitly modified "node potentials") when the flow augmenting path does not exist or is blocked.

Basic to both of these schemes is a simple change in arc representation which reclassifies each arc into an "original" arc and a "mirror," at an almost negligible increase in computer memory requirements. The reclassification, which is only symbolic and does not introduce any structural change in the network, causes each arc to become capacitated only from above, rather than from both below and above. Special relationships between the capacities, flows and marginal prices of an arc and its mirror are maintained by which the "mirror of the mirror" is identified as the original. The effect of this scheme is to reduce markedly the number of mathematical conditions which characterize the "kilter states" applicable to the arcs. These states, which identify the degree to which the current arc flow conforms to or deviates from optimality, require repetitive monitoring in the out-of-kilter method. Thus the ability of the "mirror arc" representation to contract the range of conditions by which these states are recognized leads to a convenient streamlining of the solution process.

Label-eligible nodes are also recognized with improved efficiency by the reformulation, utilizing a "partitioned successor" technique. After an initialization stage, the partitioned successor technique enables label-eligibility to be ascertained with only a portion of the customary effort.

The modifications of the original out-of-kilter method embodied in the reformulated method are organized to permit advanced primal and dual starts to be exploited to full advantage, thereby retaining flexibility to accommodate standard postoptimality considerations.

2.1 THE OUT-OF-KILTER ALGORITHM

In this section we present a brief review of Fulkerson's Out-of-Kilter Algorithm [12] (or, as it is sometimes called, the Ford and Fulkerson primal-dual algorithm [9]). This review is only meant to refresh the reader's memory and to introduce our notation. (For a more complete presentation the reader should refer to [8,10,28,29]. Our discussion largely follows the lines of [29].)

For our purposes, a network will be defined to consist of m nodes or junction points which are connected pairwise by a collection of n directed arcs (links). Although not all pairs of nodes need to be joined, the graph must be connected and each node in the network must have at least one arc leading into it and at least one leading out. Therefore, the problem which is solved by the out-of-kilter algorithm is that of finding the optimal flow in a circulatory network, and is defined as follows:

$$\text{Minimize:} \quad \sum_{(i,j)\in N} c_{ij}x_{ij} \tag{1}$$

$$\text{subject to:} \quad \sum_{(i,j)\in N} (x_{ij} - x_{ji}) = 0, \qquad i = 1,2,\ldots,m \tag{2}$$

$$x_{ij} \geq L_{ij}, \qquad (i,j)\in N \tag{3}$$

$$x_{ij} \leq K_{ij} \qquad (i,j)\in N \tag{4}$$

where x_{ij} is the flow from node i to node j, c_{ij} is the cost associated with sending one unit of flow from node i to node j, L_{ij} and K_{ij} are, respectively, the lower and upper bounds on the amount of flow on arc (i,j) and N is the set of all arcs in the network. ((i,j) denotes an arc from node i to node j.) This problem encompasses all (non-weighted) single commodity network problems

since any network (including those of transportation and assignment problems) can easily be made circulatory using standard techniques.

For such a network, a feasible circulation is defined as a set of flows satisfying relationships (2), (3), and (4). An optimal circulation satisfies (1), (2), (3), and (4).

Associated with this problem is a dual problem which may be stated as

$$\text{Maximize} \quad \sum_{(i,j)\in N} (L_{ij}u'_{ij} - K_{ij}u''_{ij})$$

$$\text{subject to:} \quad u_i - u_j + u'_{ij} - u''_{ij} \leq c_{ij}, \qquad (i,j)\in N$$

$$u_i \text{ - unrestricted}, \qquad i = 1,2,\ldots,m$$

$$u'_{ij},\ u''_{ij} \geq 0, \qquad (i,j)\in N$$

The dual variables u_i are called node "potentials" or "multipliers" and the expression $\bar{c}_{ij} = c_{ij} - u_i + u_j$ is called the "marginal cost" or "updated cost" associated with arc (i,j). By fundamental linear programming theory, a feasible circulation is optimal if and only if one of the following conditions is satisfied by each arc $(i,j)\in N$:

L: $\bar{c}_{ij} > 0$ and $x_{ij} = L_{ij}$

B: $\bar{c}_{ij} = 0$ and $L_{ij} \leq x_{ij} \leq K_{ij}$

K: $\bar{c}_{ij} < 0$ and $x_{ij} = K_{ij}$.

An arc which satisfies one of these conditions (called "in-kilter" conditions) is said to be "in-kilter"; otherwise, the arc is said to be "out-of-kilter." An "out-of-kilter" arc must then satisfy one of the following conditions (called "out-of-kilter" conditions):

L_1: $\bar{c}_{ij} > 0$ and $x_{ij} < L_{ij}$

L_2: $\bar{c}_{ij} > 0$ and $x_{ij} > L_{ij}$

B_1: $\bar{c}_{ij}=0$ and $x_{ij} < L_{ij}$

B_2: $\bar{c}_{ij}=0$ and $x_{ij} > K_{ij}$

K_1: $\bar{c}_{ij} < 0$ and $x_{ij} < K_{ij}$

K_2: $\bar{c}_{ij} < 0$ and $x_{ij} > K_{ij}$.

The general thrust of the algorithm is to bring each out-of-kilter arc in-kilter by adjusting its flow or changing its node potentials. In order to change the flow of an out-of-kilter arc (s,t), a suitable path (called a flow augmenting path) must be found from node t to node s and flow adjusted on each arc of the path by an equal amount, thus maintaining node conservation. This path is found (or its nonexistence determined) by alternating use of a labeling procedure and a potential change procedure.

The labeling procedure consists of generating from node t a tree of arcs through which flow may be changed without violating their bounds and without increasing the total cost unnecessarily. (Flow can be algebraically increased from node i to node j either by increasing the flow in the arc (i,j) or by decreasing the flow in the arc (j,i), if it exists. Total cost may need to be increased in order that boundary conditions on a given arc be met.) Arcs which qualify for inclusion in this tree at a given iteration are designated as "label eligible." If an arc is label eligible this is indicated by labeling its unlabeled node with an ordered pair. The first entry of this ordered pair indicates the node from which this node was labeled and the second the amount of flow change which could occur in the path to this point.

More specifically, let us assume that flow is to be increased on arc (s,t) (or else decreased on arc (t,s)). The steps of the procedure are then as follows:

1. Label node t with $(-,\infty)$.

2. For the arbitrary node i which is labeled with (h,α_i), label with

(i,α_j) every unlabeled node j which satisfies one of the following label eligible conditions, using the associated α_j.

label eligible	conditions	α_j
λ: $(i,j)\in N$	$c_{ij} > 0$ and $x_{ij} < L_{ij}$	$\min [\alpha_i, L_{ij}-x_{ij}]$
μ: $(i,j)\in N$	$c_{ij} \leq 0$ and $x_{ij} < K_{ij}$	$\min [\alpha_i, K_{ij}-x_{ij}]$
ν: $(j,i)\in N$	$c_{ji} \leq 0$ and $x_{ji} > L_{ji}$	$\min [\alpha_i, x_{ji}-L_{ji}]$
ρ: $(j,i)\in N$	$c_{ji} < 0$ and $x_{ji} > K_{ji}$	$\min [\alpha_i, x_{ji}-K_{ji}]$.

3. If node s is labeled, a "breakthrough" has been accomplished by identifying a flow augmenting path. The path is "recovered" by successively tracing backward through the predecessor nodes named in the labels along the chain of arcs ending at node s. Thereupon the flow in this path from node t to node s is increased by α_s(the maximum permissible quantity) by increasing or decreasing the flow in each member arc (i,j) depending on whether this arc is directed from t towards s or from s towards t.

4. If node s cannot be labeled by this procedure, then it is not possible with the present node potentials to find a path of label-eligible arcs from node t to node s. At this point, the potential change procedure is applied.

The potential change procedure, like the labeling procedure, allows only those changes (in this case, of marginal costs) that will not cause any arc to be forced to be out-of-kilter or further out-of-kilter. By applying these steps, either at least one new arc will become label eligible or the problem will be found to be infeasible (i.e., have no feasible solution).

Let L and $\bar{L}$ be the sets of labeled and unlabeled nodes, respectively, at step 4 of the labeling procedure and let

$$S_1 = \{(i,j):\ i \in L,\ j \in \bar{L},\ \bar{c}_{ij} > 0,\ x_{ij} < K_{ij}\}$$

$$S_2 = \{(i,j):\ i \in \bar{L},\ j \in L,\ \bar{c}_{ij} < 0,\ x_{ij} > L_{ij}\}\ .$$

Set $\theta = \min\{\min_{S_1}(\bar{c}_{ij}),\ \min_{S_2}(-\bar{c}_{ij})\}$, if $S_1 \cup S_2 \neq \emptyset$; otherwise, $\theta = \infty$.

If θ is infinite, the algorithm terminates because the problem is infeasible. If θ is finite, a new potential set is found by adding θ to the u_i's of each labeled node i and leaving the potential of the unlabeled nodes unchanged.

As a result, the new marginal costs are:

$$\bar{c}_{ij} = \bar{c}_{ij} - \theta,\ (i,j) \in S_1 \cup R_1$$

$$\bar{c}_{ij} = \bar{c}_{ij} + \theta,\ (i,j) \in S_2 \cup R_2$$

$$\bar{c}_{ij} = \bar{c}_{ij},\ \text{for all other } (i,j)$$

where $R_1 = \{(i,j):\ i \in L,\ j \in \bar{L}\} - S_1$

$R_2 = \{(i,j):\ i \in \bar{L},\ j \in L\} - S_2$

The algorithm now returns to step 2 in the labeling procedure, and the search for a breakthrough is continued, unless (s,t) is now in-kilter.

The complete algorithm may now be concisely stated as follows:

1. Begin with any set of potential values u_i and with any set of flow values x_{ij} that satisfy the node conservation equations (2). (A trivial start sets all $x_{ij} = 0$ and $u_i = 0$.)

2. Pick any out-of-kilter arc (i,j). If none exists, stop: the current flows and potential values are optimal.

3. Apply the labeling procedure letting s = i, and t = j if the arc is in either state L_i, B_i, or K_i; otherwise, set s = j and t = i. If s is labeled go to step 4; otherwise go to step 5.

4. Breakthrough: Increase (or decrease) the flow in the labeled path

from node s to node t and the arc (s,t) by the amount α:

$\alpha = \min[\alpha_s, L_{st} - x_{st}]$ if (s,t) is in state L_1 or B_1

$\alpha = \min[\alpha_s, K_{st} - x_{st}]$ if (s,t) is in state K_1

$\alpha = \min[\alpha_s, x_{st} - L_{st}]$ if (s,t) is in state L_2

$\alpha = \min[\alpha_s, x_{st} - K_{st}]$ if (s,t) is in state K_2 or B_2.

If (s,t) is in-kilter after this flow change, go to step 2. Otherwise, erase all labels and go to step 3.

5. Non-breakthrough: Apply the potential change procedure. If $\theta = \infty$, stop, the problem has no feasible solution. If θ is finite, update the marginal costs.

If (s,t) is now in-kilter go to step 2; otherwise, go to step 3 and continue the labeling procedure using the old labels and the current marginal costs.

2.2 NEW FORMULATION OF THE OUT-OF-KILTER ALGORITHM

The typical verbatim implementation of the out-of-kilter algorithm is subject to several types of computational improvement. To obtain an intuitive idea of where some of these potential improvements are likely to reside, it is useful to take note of portions of the method where computation is likely to be substantial. For example, set determination involves a considerable amount of search time due to the large number of rules through which arcs must be filtered. This occurs not only in checking the label-eligibility conditions, but also in the determination of sets S_1, S_2, R_1, and R_2 in the potential change procedure. Six out-of-kilter conditions must be checked for each arc, not only during the initial examination but following each potential change and breakthrough, raising the question of whether some abbreviated set of conditions may be found to exist that will accomplish

the same purpose and thus provide a less expensive filter from a computational standpoint.

Opportunity for improvement may also, or alternatively, be suspected to reside in repetitive use of certain data, not all of which changes at each step. For example, if more than one potential change must be performed during a given breakthrough attempt, the sets S_1, S_2, R_1, and R_2 must be regenerated each time, therefore requiring reinspection of arcs touching nodes which were previously labeled. Other repeated computations include the amount of "slack" in an arc, or the deviation of flow from one of the arc's stated bounds (e.g., see the definition of S_1, S_2, α, out-of-kilter conditions, and label-eligibility conditions), as well as the relative cost for arcs under inspection. Of course, the fact that these calculations involve substantial effort or even redundancy does not mean thay can be advantageously circumvented. Our goal in this section is to identify ways in which this can,in fact, be accomplished.

Reformulation of the Circulatory Network

Let us now refine our focus further. One of the most time consuming tasks in the labeling procedure and in the determination of the sets S_1, S_2, R_1, and R_2 is determining the direction of the arc each time label and membership eligibility is determined. For instance, in the labeling procedure if the arc is directed out of node i then λ and μ states are tested, whereas if the arc is directed into node i then ν and ρ states are examined. Reducing the effort involved in this testing would have direct implications for many of the calculations previously discussed. As a first step toward accomplishing this, we propose a reformulation of the circulatory network that allows us to look only at "pseudo-arcs" leading out of a node. On

the basis of this network reformulation, we will subsequently show how to reorganize the out-of-kilter method to permit a net improvement in these and other calculations.

The reformulation of the network proceeds quite simply as follows. Each arc (i,j) of the original network N is split into two interdependent pseudo-arcs. The first pseudo-arc is directed from node i to node j and has the same cost c_{ij} and upper bound K_{ij} as the original arc (i,j), but no lower bound. The other pseudo-arc is directed from node j to node i with its cost equal to the negative of the cost c_{ij}, and its upper bound equal to the negative of the lower bound L_{ij} of the original arc. This pseudo-arc, like the first, has no lower bound. The flows on the two pseudo-arcs are respectively equal to the flow of the original arc x_{ij} and the negative of this flow. If we denote the flow variables of the pseudo-arcs by x'_{ij} and x''_{ij}, respectively, we thus have

$$x'_{ij} = x_{ij} \tag{5}$$

and $$x''_{ij} = -x_{ij}. \tag{6}$$

which further implies

$$x_{ij} = 1/2(x'_{ij} - x''_{ij}). \tag{7}$$

Replacing x_{ij} in the objective function (1) and the node conservation constraint (2) by (7), x_{ij} in (3) by (6), and x_{ij} in (4) by (5), we obtain (discarding the one-half multiple in the objective function) the equivalent network problem $\bar{N}$:

Minimize $$\sum_{(i,j)\in N} c_{ij}(x'_{ij} - x''_{ij}) \tag{8}$$

subject to: $$\sum_{(i,j)\in N} [(x'_{ij} - x''_{ij}) - (x'_{ji} - x''_{ji})] = 0 \ , \ i=1,2,\ldots,m \tag{9}$$

$$x''_{ij} \leq -L_{ij}, \quad (i,j)\in N \tag{10}$$

$$x'_{ij} \leq K_{ij}, \quad (i,j)\in N \tag{11}$$

$$x'_{ij} + x''_{ij} = 0, \quad (i,j)\in N \tag{12}$$

The new network disposes of lower bounds in the original network at the expense of doubling the number of arc variables and introducing the extra constraints (12). Although this appears to be a tradeoff in the wrong direction, we will now show how to reverse this seeming disadvantage. To begin, observe the following structural properties of this network:

1. For each arc into a node there exists a "mirror" arc out of the node.

2. Each arc is only bounded from above. This property may be used to define a net capacity for each pseudo-arc as follows:

$$nc'_{ij} = K_{ij} - x'_{ij}$$
$$nc''_{ij} = -L_{ij} - x''_{ij}$$

That is, net capacity corresponds to the capacity of a pseudo-arc to accept a flow increase without exceeding its upper bound. From constraint (12) a change of flow on a pseudo-arc induces the negative of this change in the flow on its mirror, and thus correspondingly changes the net capacities of these arcs by equal but opposite amounts.

3. The marginal cost $\bar{c}'_{ij}$ of the pseudo-arc associated with x'_{ij} is equal to the marginal cost of the original arc (i,j); namely, $\bar{c}_{ij}$. On the other hand, the marginal cost $\bar{c}''_{ij}$ of the mirror pseudo-arc is equal to $-\bar{c}_{ij}$. Thus a change in the marginal cost of a pseudo-arc produces an equal but opposite change in the marginal cost of its mirror.

A foundation for exploiting these properties is given by the following scheme for data organization.

With each main arc associate a unique integer a_{ij} which is a number between 1 and n; and with its mirror arc, associate the number $(a_{ij} + n)$. These integers will be called arc numbers. Also associate with each node i the set B_i of all arc numbers associated with pseudo-arcs directed out-of

node i. Thus $B_i = \{b: b = a_{ij}$ if $(i,j) \in N$ or $b = (a_{ij} + n)$ if $(j,i) \in N\}$. In addition, let $\bar{c}(b)$, $nc(b)$ and $m(b)$ denote functions whose values correspond to the margin cost, net capacity, and mirror arc number, respectively, associated with arc number b; i.e.,

$$\bar{c}(b) = \begin{cases} \bar{c}'_{ij} & \text{if } b = a_{ij} \\ \bar{c}''_{ij} & \text{if } b = a_{ij} + n; \end{cases}$$

$$nc(b) = \begin{cases} nc'_{ij} & \text{if } b = a_{ij} \\ nc''_{ij} & \text{if } b = a_{ij} + n; \end{cases}$$

$$\text{and} \quad m(b) = \begin{cases} b + n & \text{if } b = a_{ij} \\ b - n & \text{if } b = a_{ij} + n \end{cases}$$

Finally, let $\gamma(b)$ denote a function such that for each arc number b, its value is the "to-node" index j corresponding to the direction of the pseudo-arc; i.e.,

$$\gamma(b) = \begin{cases} j, & \text{if } b = a_{ij} \\ j, & \text{if } b = (a_{ji} + n). \end{cases}$$

This data organization allows us (1) to associate a unique number with each pseudo-arc in a reformulated network, (2) to quickly determine the mirror's arc number, given the main's number and vice versa, (3) to identify the set of outward-directed pseudo-arcs for a given node, and (4) to determine the marginal cost and net capacity associated with an arc number. Ways of restructuring the out-of-kilter method to afford more significant advantages will now be considered.

New Labeling Procedure

To provide a point of reference, suppose the label process of Section 2.1 is to be applied at node i, which is currently labeled. Let $\bar{L}$ denote the set of unlabeled nodes. Then, using $\bar{c}(b)$, $nc(b)$, $m(b)$, and $\gamma(b)$ as defined above, if arc $(i,j) \in N$ (implying that the arc number $a_{ij} \in B_i$), the

λ and μ label eligibility conditions may be stated as:

λ: $\bar{c}(b) > 0$, $nc(m(b)) < 0$, $\gamma(b) \in \bar{L}$

μ: $\bar{c}(b) > 0$, $nc(b) < 0$, $\gamma(b) \in \bar{L}$

Similarly if arc $(j,i) \in N$ (implying that the arc number $b = (a_{ji} + n) \in B_i$), then the ν and ρ label eligible conditions may be stated as:

ν: $\bar{c}(b) \leq 0$, $nc(b) > 0$, $\gamma(b) \in \bar{L}$

ρ: $\bar{c}(b) > 0$, $nc(m(b)) < 0$, $\gamma(b) \in \bar{L}$

Thus, using the new data structure, the μ and ν conditions become identical (as a consequence of the relationship between marginal costs for a pseudo-arc and its mirror). Similarly the λ and ρ conditions become identical. Thus our restructuring has reduced the label eligibility conditions from four to two, and the standard labeling procedure can be simplified to examining each arc number in B_i one at a time, checking the new label eligibility states λ and $\mu (= \nu$ and $\rho)$. However, we can in fact improve the calculations somewhat more than this.

Partitioning

The labeling process can be facilitated by partitioning the set B_i into those arc numbers whose associated arcs are label-eligible and those which are not. We denote the set of label eligible arc numbers by P_i and the remaining arc numbers by $Q_i = B_i - P_i$.

Once the sets P_i and Q_i of the partition have been identified, the forward tree generation of the labeling process is accelerated since checking of net capacity, marginal cost, and direction of each arc is eliminated. Further, once the partition is determined it requires minimum maintenance. In particular, this maintenance involves checking the arcs in the flow augmentation cycle during breakthrough and checking those arcs affected by a change of node potentials. (Details of such a procedure will be discussed

in the breakthrough and potential change parts of this section.) Another benefit of the partition is that knowledge of the set Q_i shortcuts the determination of the sets used to determine node potential changes.

To implement the labeling process, the label for a given node j is simplified to consist only of the predecessor arc number b corresponding to pseudo-arc (i,j) across which node j is labeled. This change coupled with the partitioning of B_i has the following theoretical and computational advantages:

1. The use of predecessor arc numbers instead of a node index label avoids possible ambiguity in the reverse pass when a network contains multiple arcs between a given pair of nodes. (This situation can arise, for example, when a concave objective function is replaced by a piecewise linear approximation.)

2. By eliminating the α_j values in the label, less information is processed on the forward tree generation and more information is processed on the reverse pass; however, the flow augmentation pass typically involves only a portion of the nodes labeled on the forward pass, and sometimes the reverse pass is not executed at all. (This latter is the case when the out-of-kilter arc (s,t) becomes in-kilter by a potential change.)

3. Due to the network reformulation the direction of each arc in P_i is out-of-node i; thus, node labeling can be carried out quite rapidly using the function γ. That is, each arc number $b \in P_i$ is checked to determine whether node $\gamma(b)$ is labeled. If it isn't, $\gamma(b)$ is labeled with b; otherwise, the next arc number in P_i is examined.

A concise statement of the new labeling procedure (for the objective of increasing flow on pseudo-arc (s,t) or decreasing flow on pseudo-arc (t,s)) is as follows:

1. Label node t with the arc number of pseudo-arc (s,t).

2. For any arbitrary labeled node i (i.e., $i \in L$), label each unlabeled node j (where $j = \gamma(b)$, $b \in P_i$) with b.

3. If node s is labeled, go to breakthrough. If s cannot be labeled, go to the potential change procedure before returning to step 2.

Specialized Computer Results Pertaining to the New Labeling and Partitioning Procedures

We have conducted "localized" computer tests of the out-of-kilter algorithm to provide insight into the specific computational effects of the new labeling and partitioning procedures. A later section is devoted to "global" computational testing and analysis of a wide variety of problems with different out-of-kilter codes.

For the purpose of our present concern, special statistics have been collected on five, 500 node, type I networks and on five, 500 node, type II networks. Each time labeling was performed at a node i, a count was made of the number of arc numbers currently in P_i and in B_i; these counts were accumulated and are shown in Table 1 under columns "ΣP_i" and "ΣB_i". Further accumulated totals of the number of cycle elements and the number of labels generated are also shown in Table 1. (Other statistics in Table 1 will be discussed subsequently.)

The column "Total Labels Generated" shows that the median number of labels generated is 7961 for the type I networks and 15531 for the type II networks. Comparing this with the "Cycle Elements/Labels" column of Table 1 confirms the expectation that net computational savings must result by not generating α_j values on the forward labeling pass. In particular, on the reverse pass (which identifies the flow augmenting path) the percentage of "total" (i.e., "forward pass") α_j values required was only 9.7% (median)

TABLE 1
LABELING AND BREAKTHROUGH STATISTICS
ON THE 500 NODE NETWORKS

Density	Type	Total Cycle Elements	Total Labels Generated	Cycle Elements / Labels	ΣBi	ΣPi	$\frac{\Sigma Pi}{\Sigma Bi}$	BT*	Average Cycle Length	Labels / BT*	Label Outs
.006	I	1161	5,460	.213	34,918	7,206	.206	73	15.9	74.7	286
.009	I	769	7,940	.097	80,347	9,812	.122	76	10.1	104.4	201
.011	I	700	7,961	.088	94,104	9,956	.106	76	9.2	104.7	173
.014	I	739	8,432	.088	117,891	10,330	.088	77	9.5	109.5	164
.017	I	775	6,754	.155	110,156	8,142	.074	71	10.9	95.1	167
.006	II	712	15,531	.046	123,092	17,865	.145	65	10.9	238.9	268
.009	II	580	13,135	.044	140,465	14,985	.107	58	10.0	226.4	206
.011	II	893	15,202	.059	182,337	18,084	.099	74	12.0	205.4	228
.014	II	924	15,734	.059	228,282	18,526	.081	85	10.8	185.1	226
.017	II	811	16,288	.050	289,014	18,806	.065	79	10.2	206.1	203

*BT - Breakthroughs

for type I networks and 5% (median) for type II networks. (See the column "Cycle Elements/Labels" of Table 1.) The value of the partitioning scheme is also illustrated by the table. The column entitled $\Sigma P_i / \Sigma B_i$ shows that the median percentage of arcs which are label eligible from a given node is 10.6% for type I and 9.9% for type II networks. Consequently, the partition makes it possible to avoid checking a substantial percentage of arcs on the forward labeling pass.

Offsetting this, the column "Average Cycle Length" discloses that median average cycle length is 10.118 arcs for type I networks and 10.871 arcs for type II networks. Thus, to maintain the partition, the label eligibility of each of these arcs and their mirrors must be examined. However, this involves only a fraction of the effort avoided on the forward pass (as will be deomonstrated subsequently).

A good deal of checking is also saved by the unidirectional nature of the arcs in P_i as shown by the column entitled ΣT_i in Table 1. For instance, the median number of such checks on type I and type II networks would have been 9812 and 18084, respectively.

In combination, these statistics generally confirm the hypothesis that it is better to process less information on the forward pass and more information on the reverse pass.

Breakthrough

In determining the flow change α, the new network representation once again contributes to decreased computational effort in this case by using net capacity nc_{ij} to eliminate a subtraction operation and by using marginal costs to distinguish states.

The exact calculations performed in updating the flows are as follows:

1. Let C_μ denote the node labels in the flow augmenting path which are in label-eligible state $\mu(=\nu)$; similarly, let C_λ denote the node labels in this path which are in the label-ineligible state $\lambda(=\rho)$. (Note these states can be distinguished by inspection of the marginal cost.) Then

$$\alpha = \min[\min_{b\in C_\mu} nc(b), \min_{b\in C_\lambda} nc(m(b))]$$

2. Retrace the flow augmenting path to update the net capacity of each pseudo-arc and its mirror by setting:

$$nc(b) = nc(b) - \alpha$$

$$nc(m(b)) = nc(m(b)) + \alpha \text{ for all } b\in C_\mu \cup C_\lambda$$

3. Simultaneously update the sets P_i and Q_i, for each node i on the flow augmenting path as follows:

a. Move the arc number from Q_i to P_i for the pseudo-arc (if it exists) whose mirror lies in the flow augmenting path, provided the arc's net capacity becomes positive during the flow change and its marginal cost equals zero. (Only such arcs can change from label ineligibility to label eligibility.)

b. Move the arc number from P_i to Q_i for the pseudo-arc (if it exists) that lies in the flow augmenting path, provided either:

1) its marginal cost is non-positive and its new net capacity is zero;

or 2) its marginal cost is positive and its new mirror capacity is zero.

New Potential Change Procedure

By keeping a list of the labeled nodes L and using the sets B_i, the membership conditions of S_1, S_2, R_1, and R_2 can be halved. The arcs contained in S_1, S_2, R_1, and R_2 can then be found simply by examining the

arcs in the sets Q_i associated with labeled nodes. To see this, note that both nodes of any arc $b \in P_i$ will be labeled if node i itself is labeled. Thus, attention may be restricted to arcs $b \in Q_i$ to find all labeled - unlabeled (and unlabeled-labeled) arcs. An additional simplification facilitates the determination of arcs in S_1 and S_2, by means of the following observation.

Remark: If $i \in L$ and $b \in Q_i$, then arc b (or its mirror) is in S_1 or S_2 if and only if $\gamma(b) \in \bar{L}$, $\bar{c}(b) > 0$, and $nc(b) > 0$.

Proof: Suppose $i \in L$ and $b = a_{ij} \in Q_i$. Then by assumption $\gamma(b) = j \in \bar{L}$, $\bar{c}(b) = c_{ij} > 0$, and $nc(b) = nc'_{ij} > 0$. Thus, arc $(i,j) \in N$ belongs to set S_1. On the other hand, suppose $i \in L$ and $b = (a_{ji} + n) \in Q_i$. Then $\gamma(b) = j \in \bar{L}$, $\bar{c}(b) = -\bar{c}_{ij} > 0$, and $nc(b) = nc''_{ij} > 0$. Thus, arc $(j,i) \in N$ belongs to set S_2. The "only if" part of the remark follows similarly, completing the proof.

To take advantage of this observation in updating the current marginal costs let S_i denote the set of all arc numbers $b \in Q_i$ such that arc b (or its mirror) is in S_1 or S_2. Also let $\bar{R}_i$ denote the set of all arc numbers $b \in Q_i$ such that arc b is a labeled-unlabeled arc and $b \notin S_i$; i.e., $\bar{R}_i = \{b: \gamma(b) \in \bar{L}, b \notin S_i\}$. Finally let $S = \cup \bar{S}_i$ and $R = \cup \bar{R}_i$, where these unions are taken over the index set of labeled nodes. Then we identify the marginal cost increment θ by

$$\theta = \min_{b \in S} \bar{c}(b)$$

(If this calculation cannot be performed because S is empty, the problem has no feasible solution.) The new marginal costs may accordingly be found by updating the current marginal costs as follows:

$$\bar{c}(b) = \bar{c}(b) - \theta, \qquad b \in S \cup R$$

$$\bar{c}(m(b)) = \bar{c}(m(b)) + \theta, \qquad b \in S \cup R$$

$$\bar{c}(b) = \bar{c}(b), \qquad \text{for all other arc numbers.}$$

In order to maintain the partition efficiently, arc numbers are checked for movement from set P_i to Q_i or from Q_i to P_i at the same time the marginal costs are updated.

The checks required are the following:

1. If $b \in \bar{S}_i$, $nc(b) > 0$ and the updated marginal cost $\bar{c}(b) = 0$, then arc b has entered μ state and should be moved from Q_i to P_i.

2. If $b \in \bar{R}_i$, $nc(b) = 0$, the old marginal cost $\bar{c}(b) \geq 0$, and the new marginal cost $\bar{c}(b) < 0$, then arc $m(b)$ becomes label-ineligible and should be moved from P_i to Q_i.

To minimize effort in the determination of sets S and R where more than one change of potential must be effected to bring a given arc in-kilter, the previous sets S and R are purged of currently nonconforming arcs and then augmented by appropriate elements of the sets Q_i (where node i has been labeled since the previous potential change). Specifically, let L' denote the set of nodes which have been labeled since the last potential change, and let L and $\bar{L}$ denote all currently labeled and unlabeled nodes, respectively (i.e., $L' \subset L$). Then we can define the updated form of sets S and R as follows:

$$S' = \{b \in S: \gamma(b) \in \bar{L}, \bar{c}(b) > 0\} \cup \{b: \gamma(m(b)) \in L', \gamma(b) \in \bar{L}, \bar{c}(b) > 0, nc(b) > 0\} \tag{13}$$

$$R' = \{b \in R: \gamma(b) \in \bar{L}\} \cup \{b \in S: \gamma(b) \in \bar{L}, \bar{c}(b) = 0\} \cup [\{b: \gamma(m(b)) \in L', \gamma(b) \in \bar{L}\} - S'] \tag{14}$$

Thus S is screened to form subsets of S' and R', R is purged of non-conforming arcs and added to R', and the sets Q_i for $i \in L'$ are scanned to

form the remaining elements of S' and R'. This "recycling" of the sets S and R further restricts the population from which the new cut set is generated and, thus, reduces the effort required to effect multiple potential changes.

The basic steps of the new potential change procedure are:

1. If a previous potential change has been made for the current breakthrough attempt, perform step 3; otherwise go to step 2.

2. For each labeled node i, inspect Q_i and create the sets $\tilde{S}_i$ and $\tilde{R}_i$. Let $S = \cup\tilde{S}_i$ and $R = \cup\tilde{R}_i$ and go to step 4.

3. Form new sets S and R as in (13) and (14).

4. If $S = \emptyset$, stop; there is no feasible solution to the problem. Otherwise, let $\theta = \min_{b \in S} \bar{c}(b)$.

5. If $b \in S \cap R$ update the relative costs as follows:

$$c(b) = c(b) - \theta$$

$$c(m(b)) = c(m(b)) + \theta$$

while simultaneously updating the partition.

Specialized Computer Results Pertaining to the New Breakthrough and Change of Potential Procedures

Table 2 provides some statistics regarding the usefulness of the partition and the recycling of R and S in the breakthrough and change of potential procedures. To study the value of the partition, each time that a potential change was effected, the number of arcs in B_i and Q_i were counted and accumulated and when multiple changes were made, only the B_i and Q_i associated with the newly-labeled nodes (L') were counted. These cumulative totals, given in columns "ΣB_i" and "ΣQ_i" indicate the number of arcs which would have had to have been inspected without the partition (ΣB_i) as opposed to those which actually were inspected (ΣQ_i) to form the

TABLE 2
CHANGE OF POTENTIAL STATISTICS
ON THE 500 NODE NETWORKS

Density	Type	Σ(S+R)	Recycled ΣS+R	Recycled ΣQi	Recyl ΣS+R / Recyl ΣQi	ΣBi	ΣQi	ΣQi / ΣBi	# of Multiple Changes	# of Potential Changes	% of Multiple Changes
.006	I	28,983	15,948	25,691	.621	23,532	18,237	.775	149	213	70%
.009	I	66,037	32,632	51,179	.638	53,758	47,220	.878	64	125	51%
.011	I	63,363	26,166	38,280	.684	60,007	53,679	.895	45	97	46%
.014	I	65,365	21,160	34,299	.617	67,990	62,148	.914	35	87	40%
.017	I	80,848	37,454	50,092	.748	64,447	59,809	.928	49	96	51%
.006	II	109,238	68,444	215,595	.317	104,436	89,652	.858	145	203	72%
.009	II	106,786	67,192	212,566	.316	101,176	90,755	.897	102	148	70%
.011	II	142,447	79,064	175,912	.449	130,476	117,895	.904	95	154	62%
.014	II	134,316	63,557	145,981	.435	143,708	132,705	.923	81	141	57%
.017	II	159,168	60,556	128,058	.473	196,018	183,722	.937	62	124	50%

base population from which all sets S and R were derived. The column "$\Sigma Q_i/\Sigma B_i$" thus indicates that the partition avoided examining about 9% of the arcs unnecessarily - a relatively insignificant saving by comparison to the substantial gains from the partition that were identified relative to the labeling procedure.

A second set of statistics were gathered to study the effect of the recycling filters which, as the table shows, were used in generating the cut set in over one-half of the cases, on the average. To get an indication of the amount of work which these screening devices saved, each time S and R were recycled, the number of reviewed elements from the previous sets were counted as were the number of arcs which would have had to have been inspected if the recycling filters had not been used. These cumulative totals are given in columns "Recycled ΣS+R" and "Recycled ΣQ_i", respectively. The ratios of these totals, given in "Recycled ΣS+R/Recycled ΣQ_i" column, indicate the proportion of arcs passing through the filters to the arcs which would have otherwise required inspection. These ratios indicate that the filters eliminated from inspection, on the average, approximately 35% of the arcs in the type I networks and about 60% in the type II networks.

New Kilter Conditions

The six original out-of-kilter conditions may be compacted to the following four states for a given pseudo-arc number b:

	condition	corrective action
A:	$nc(b) < 0$	decrease flow on b
E:	$nc(m(b)) < 0$	increase flow on b
F:	$nc(b) > 0$, $\bar{c}(b) > 0$	increase flow on b
G:	$nc(m(b)) > 0$, $\bar{c}(m(b)) < 0$	decrease flow on b.

The equivalence of these conditions to the six original out-of-kilter conditions is easily established. From the symmetric properties of these conditions, it follows that whenever a pseudo-arc is in-kilter then its mirror is also in-kilter. Once a pseudo-arc is in-kilter it will, moreover, always remain in-kilter due to the equivalence of the new labeling, breakthrough, and potential change procedures to the original procedures. Thus by successively putting the "main" pseudo-arcs in-kilter (i.e., those pseudo-arcs corresponding to original arcs), an optimal solution can be obtained by examining each of these arcs once. Alternatively, if pseudo-arcs are inspected as main-mirror pairs in the state determination process, only states A, E, and F need be checked for the main pseudo-arc and only state G need be checked for the mirror (i.e., the main pseudo-arc can be considered in-kilter if it violates states A, E, and F, whereupon the mirror is immediately put in-kilter via state G).

The next section reports a computational comparison of a code (called SUPERK) using this reformulated structure and three other widely used out-of-kilter codes.

3.0 COMPUTATIONAL RESULTS AND THEIR INTERPRETATION

Section 3.1 presents our results on transportation problems. Section 3.2 provides the computational results and interpretations on uncapacitiated network problems and discusses the interrelationship of the data in sections 3.1 and 3.2. Further, this section also discusses the effect of type I and II networks and the effect of advance dual start methods on total solution time. Section 3.3 summarizes the computational data on the capacitated networks problems and section 3.4 indicates the computer memory requirements of the codes tested.

3.1 TRANSPORTATION RESULTS

Since most of the published computational results that relate to minimum cost flow problems deal with transportation problems, these problems were examined first. The results obtained from this comparison characterize the basic results that prevail for general minimum cost flow network problems as well. One of the unique findings obtained by a joint analysis of both types of problems is that an increase in the number of sources (origins) and sinks (destinations) for a fixed problem size causes the solution time to increase. This finding is unexpected since it is a part of the folklore that transportation problems are easier to solve than network problems. The implications of this will be discussed in detail later.

Tables 3 and 4 report our results on 55 transportation problems varying in size from 10 x 10 to 150 x 150 and varying in density from 90% to 5%. Each set of m x m transportation problems contained five problems. Table 3 contains the median solution time and solution time range for each of these problem sets. Table 4 gives the individual solution times for each problem in the 50 x 50, 80 x 80, and 100 x 100 problem sets. "NA" in Tables 3 and 4 indicate that we could not solve these problems using the TWB code due

TABLE 3

Median[1] Solution Times (sec) and Their Range
For Transportation Problems

Problem Size	Density[2]	SHARE Median	SHARE Range	Boeing Median	Boeing Range	TWB Median	TWB Range	SUPERK Median	SUPERK Range
10x10	.31	.10	.05-.14	.08	.04-.15	.07	.04-.10	.07	.05-.11
20x20	.66	.68	.64-.87	1.04	.58-2.36	.48	.43-.74	.23	.21-.27
30x30	.61	2.04	1.21-2.80	1.75	1.13-11.75	1.44	1.31-3.29	.46	.36-.62
40x40	.36	2.42	2.11-3.57	2.67	1.79-10.33	1.96	1.51-2.85	.48	.39-.64
50x50	.53	5.70	4.22-10.56	12.73	2.65-31.46	5.33	2.40-11.08	.84	.73-1.39
60x60	.20	5.28	4.41-12.30	12.91	2.95-29.12	5.05	3.16-8.29	1.01	.81-1.17
70x70	.28	9.46	6.73-18.48	17.39	2.97-63.96	10.03	4.10-17.71	1.19	1.14-1.86
80x80	.31	22.10	11.89-25.51	18.38	12.72-90.70	14.30	7.61-23.60	2.50	1.42-2.77
90x90	.28	26.35	13.27-32.88	21.68	11.2-73.0	NA		2.66	1.87-3.13
100x100	.20	21.17	16.81-28.78	45.03	9.91-79.80	NA		2.42	2.21-3.10
150x150	.23	56.20	46.98-88.01	535.16	49.3 -1164.0	NA		3.93	2.91-6.49

[1] All times are median times with five problems per group

[2] Mean Density

NA - Will not fit into the computer memory available.

TABLE 4

Solution Times (sec) For Selected Transportation Problems

Problem Size	Density	SHARE	Boeing	TWB	SUPERK
50x50	.26	4.44	2.65	2.40	.84
	.27	4.22	2.38	2.88	.73
	.53	6.69	28.82	5.33	.96
	.54	5.70	31.46	11.08	.73
	.87	10.56	12.73	6.65	1.39
80x80	.19	11.89	16.00	7.61	1.42
	.27	15.21	12.72	13.58	1.68
	.31	22.10	19.54	14.30	2.72
	.40	24.42	90.70	14.66	2.77
	.44	25.51	18.38	23.60	2.50
100x100	.13	16.81	45.03	NA	2.21
	.15	21.05	9.91	NA	2.79
	.20	21.17	54.67	NA	2.21
	.22	27.45	79.80	NA	3.10
	.29	28.78	39.59	NA	2.42
150x150	.14	48.29	535.16	NA	3.93
	.20	58.93	49.29	NA	4.55
	.23	46.98	760.90	NA	2.91
	.27	56.20	1164.00	NA	3.35
	.28	88.01	470.36	NA	6.49

NA - Will not fit into the computer memory available

to computer memory requirements of the code for problems of this size and density. This limitation in the size of problems that can be handled is due to a design incorporated into the code by the Texas Water Development Board which improves its efficiency at the expense of computer memory.

A noteworthy feature of the computational results that pervades the entire study is the fact that each apparent trend was found to require qualification. To nearly every positive conclusion there was an exception. The most important consistent finding is that the code based on the formulation of Section 2 is decidedly superior to the others. Aside from this, the behavior of the four codes varied widely on like problems. The range of solution times reported in Table 3 provides an illustration. Out of these erratically diverging times emerge the following uniformities. The Boeing Code exhibits the widest range of solution times for each problem size and SUPERK the narrowest range. Similarly, the Boeing Code is often the least efficient and, as already remarked, SUPERK is without exception the most efficient. (Roughly, SUPERK is at least 5 times faster and in many cases 8-10 times faster than the other three codes.) However the data in Table 4 indicate that among the codes SHARE, Boeing, and TWB, none is strictly dominant.

The times in Table 4 portray the sensitivity of out-of-kilter methods to density. Reduced density, as might be expected, leads to definite improvements in computation times. For instance all methods are uniformly slower on the .53density 50 x 50 problems reported in Table 4 than on the .27 density problems of the same size. It should be noted however that the times do not always strictly increase with density, particularly if the change in density is not large. An illustration is provided by the 100 x 100 problems in Table 4 of density .22 and .29. Clearly, density is not

the only factor that affects solution time.

An earlier in-depth study of transportation methods [15] makes it possible to compare our results with SUPERK to results obtained for more specialized codes that were designed specifically for bipartite (transportation) networks. This study discloses that the most efficient solution procedure for transportation problems arises by coupling a primal transportation algorithm (embodying recently developed methods for accelerating the determination of basis trees and dual evaluators) with a version of the Row Minimum start rule and a "modified row first negative evaluator" rule. The resulting method has been found to be at least 100 times faster than OPHELIE, and 6 times faster than the SHARE out-of-kilter code.[2] Since the present study and the study in [15] use overlapping transportation problem sets and the same computer, a direct comparison of results is warranted. This comparison yields the following observations:

1) SUPERK is about 100 times the OPHELIE/LP general simplex linear programming code [24,25] since the median of OPHELIE/LP on the 100 x 100 problems is 277 seconds. OPHELIE/LP is an "in-core, out-of-core" code; thus, this comparison is not completely fair to general simplex computer codes which exploit sparse coefficient matrices. However, this LP code does have the advantage of being coded in a machine language and of being designed to exploit all of the computational features of the CDC 6600. In addition, the authors are not aware of any in-core LP code capable of a 100 x 100 transportation problem. (This situation may soon be changed

by a code being developed by David Sommer, Robert Choquist, and others of Control Data, drawing on ideas for storing and manipulating data proposed by James Kalan of S.M.U.)

2) The special purpose simplex code of [15] is about 30% faster than SUPERK on transportation problems. Since the code of [15] is the fastest known special-purpose simplex code[3], we must conclude that SUPERK is substantially faster than most simplex-based codes; however, the existence of a faster code(for the restricted class of bipartite networks) substantiates the findings of [15] that out-of-kilter is not the most efficient algorithm for solving transportation problems.[4] This result is contrary to the results in [8, p. 109] where Ford and Fulkerson reported that the primal simplex method (MODI method) is twice as slow as the out-of-kilter method in this context.

3.2 UNCAPACITATED MINIMUM COST FLOW NETWORKS

In this section we examine how the four out-of-kilter codes compare on more general networks than those of transportation problems. These networks vary in size from 50 to 1500 nodes, have densities ranging from .001949 to .30, and contain different numbers of source and sink nodes. We also examine the effect of two advanced dual start methods on total solution time. Tables 5,6, and 7 contain our results.

Table 5 reports results for the Type I and Type II networks, whose topological characteristics were discussed in Section 1.0. Solvability differences of these problem types are also illustrated by the data in Tables 1 and 2. Tables 6 and 7 are devoted to reporting results on capacitated problems. The problems of these tables were solved twice using SUPERK - once with and once without the capacity values to afford comparisons of the performance of the method on a given network structure

TABLE 5

Type I and II Upcapacitated Network Problems
Various Sizes

Source	Sinks	Nodes	Density	Type	SHARE	Boeing	TWB	SUPERK	ADS-1	ADS-2
3	7	50	.03	I	.162	.080	.088	.048	.072	.062
5	3		.08	I	.294	.131	.159	.094	.120	.118
4	3		.15	I	.707	.332	.423	.224	.248	.250
3	2		.24	I	1.283	.806	1.001	.389	.232	.248
4	5		.30	I	1.743	.993	1.387	.588	.484	.456
8	7	100	.03	I	1.105	.546	.630	.333	.300	.292
9	9		.08	I	2.921	1.439	1.899	.874	.844	.846
5	8		.15	I	3.932	2.254	3.327	1.146	.925	.935
7	9		.24	I	4.640	2.978	5.476	1.497	1.312	1.328
5	5		.30	I	5.592	3.703	6.595	1.585	1.413	1.433
3	10		.03	II*	DNR	DNR	DNR	.510	DNR	DNR
3	10		.03	II*	DNR	DNR	DNR	.579	DNR	DNR
5	15		.24	II*	DNR	DNR	DNR	1.408	DNR	DNR
5	15		.24	II*	DNR	DNR	DNR	1.586	DNR	DNR
4	12	400	.006	II*	DNR	DNR	DNR	2.390	DNR	DNR
4	12		.014	II*	DNR	DNR	DNR	3.940	DNR	DNR
8	60		.006	II*	DNR	DNR	DNR	4.624	DNR	DNR
8	60		.014	II*	DNR	DNR	DNR	6.098	DNR	DNR
28	27	500	.006	I	15.850	3.170	4.868	1.813	3.091	3.131
25	26		.009	I	16.213	5.911	12.226	3.655	6.098	6.107
29	29		.011	I	15.537	5.906	NA	3.707	6.518	6.568
28	27		.014	I	17.516	5.710	NA	3.959	7.730	7.725
26	28		.017	I	21.748	6.913	NA	4.569	8.770	8.905
28	27		.006	II	26.655	131.103	26.730	6.263	DNR	DNR
25	26		.009	II	24.270	53.709	30.786	6.125	DNR	DNR
29	29		.011	II	29.278	48.130	NA	7.354	DNR	DNR
28	27		.014	II	30.625	55.691	NA	7.789	DNR	DNR
26	28		.017	II	34.152	64.990	NA	9.339	DNR	DNR
50	50	1000	.0029	II	85.626	46.237	NA	19.599	20.439	20.068
50	50		.0034	II	87.580	157.110	NA	18.193	21.672	21.942
50	50		.0038	II	83.579	175.331	NA	20.548	26.802	27.367
50	50		.0044	II	85.647	153.924	NA	21.574	24.570	24.150
50	50		.0048	II	104.549	190.878	NA	26.739	28.511	28.150
75	75	1500	.001949	II	137.809	122.519	NA	33.424	DNR	DNR
75	75		.001930	II	138.405	353.758	NA	35.158	DNR	DNR
75	75		.002270	II	128.248	252.608	NA	34.801	DNR	DNR
75	75		.002264	II	135.574	394.551	NA	33.710	DNR	DNR
75	75		.002547	II	138.535	283.173	NA	35.753	DNR	DNR

*Median of 6 problems
NA - Will not fit into the computer memory available
ADS-I - Advance Dual Start version I
ADS-II- Advance Dual Start version II

DNR-Did not run

TABLE 6
Solution Times (sec) for Type II, 100 Node, Capacitated and Uncapacitated Networks

Density	Capacitated Density	Upper Capacity Range	Number of Sources	Number of Transhipment Sources	Number of Sinks	Number of Transhipment Sinks	Total Cost Capacitated	Total Cost Uncapacitated	SUPERK Uncapacitated	Capacitated SUPERK	Capacitated TWB	Capacitated Boeing	Capacitated SHARE
.03	.20	4000-12000	3	0	10	0	10,841,534	9,897,120	.486	.473	.760	.642	1.433
.03	.40	4000-12000	3	0	10	0	14,928,690	11,948,054	.445	.644	1.147	1.642	2.076
.03	.80	4000-12000	3	0	10	0	17,540,352	8,784,154	.565	.771	1.560	2.261	2.619
.24	.20	4000-12000	3	0	10	0	2,852,000	2,810,265	1.461	1.692	NA	22.916	4.849
.24	.40	4000-12000	3	0	10	0	4,764,802	4,543,635	1.441	1.921	NA	6.228	6.033
.24	.80	4000-12000	3	0	10	0	3,473,794	2,611,451	1.306	2.385	NA	12.083	7.614
.03	.20	4000-12000	3	2	10	5	9,327,579	8,330,279	.593	.654	1.383	1.876	DNR
.03	.40	4000-12000	3	2	10	5	8,770,404	7,879,394	.611	.687	1.281	1.427	DNR
.03	.80	4000-12000	3	2	10	5	9,261,267	9,187,123	.666	.640	1.374	1.713	DNR
.24	.20	4000-12000	3	2	10	5	1,367,292	1,367,292	1.513	1.623	NA	18.542	DNR
.24	.40	4000-12000	3	2	10	5	1,533,735	1,377,162	1.498	1.602	NA	4.377	DNR
.24	.80	4000-12000	3	2	10	5	1,622,683	1,311,987	1.462	1.806	NA	6.924	DNR
.03	.20	4000-12000	5	0	15	0	9,703,652	9,647,599	.504	.527	.941	1.537	1.576
.03	.40	4000-12000	5	0	15	0	10,188,114	9,896,576	.553	.623	1.107	1.787	1.844
.03	.80	4000-12000	5	0	15	0	14,187,591	10,167,368	.598	.983	1.982	1.93	2.896
.24	.20	4000-12000	5	0	15	0	2,785,668	2,644,525	1.231	1.445	NA	19.234	4.294
.24	.40	4000-12000	5	0	15	0	2,229,115	1,950,309	1.714	2.051	NA	10.006	6.106
.24	.80	4000-12000	5	0	15	0	3,089,677	1,979,768	1.625	2.013	NA	7.970	6.821
.03	.20	10000-20000	3	0	10	0	13,715,604	13,010,453	.493	.510	880	1.945	1.854
.03	.40	10000-20000	3	0	10	0	12,402,040	12,140,154	.568	.609	1.345	2.586	2.108
.03	.80	10000-20000	3	0	10	0	10,575,749	10,507,594	.527	.543	1.139	.950	1.722
.24	.20	10000-20000	3	0	10	0	1,387,753	1,338,753	1.413	1.041	NA	13.378	3.422
.24	.40	10000-20000	3	0	10	0	1,624,931	1,557,338	1.335	1.587	NA	5.411	4.287
.24	.80	10000-20000	3	0	10	0	2,078,680	1,805,817	1.375	1.707	NA	5.222	5.301
.03	.20	10000-20000	3	2	10	5	12,663,627	12,523,493	.606	.565	1.289	1.858	DNR
.03	.40	10000-20000	3	2	10	5	10,611,713	10,298,817	.520	.600	1.070	1.598	DNR
.03	.80	10000-20000	3	2	10	5	12,079,884	11,915,394	.744	.803	1.591	2.690	DNR
.24	.20	10000-20000	3	2	10	5	2,683,285	2,645,343	1.437	1.519	NA	16.260	DNR
.24	.40	10000-20000	3	2	10	5	3,043,212	3,009,800	1.318	1.404	NA	7.773	DNR
.24	.80	10000-20000	3	2	10	5	2,292,124	2,140,199	1.902	1.914	NA	6.303	DNR
.03	.20	10000-20000	5	0	15	0	12,258,157	12,049,871	.644	.624	1.175	1.367	2.980
.03	.40	10000-20000	5	0	15	0	10,736,239	10,668,609	.559	.653	1.130	1.761	1.951
.03	.80	10000-20000	5	0	15	0	7,452,412	7,180,730	.593	.571	1.180	2.294	2.003
.24	.20	10000-20000	5	0	15	0	1,799,761	1,799,761	1.674	1.661	NA	15.462	4.835
24	.40	10000-20000	5	0	15	0	3,558,951	3,545,2??	1.493	1.521	NA	5.594	5.266
.24	.80	10000-20000	5	0	15	0	3,769,153	3,665,13?	1.541	1.795	NA	13.030	5.216

NA - would not fit into the computer memory available.

DNR- Did not run

TABLE 7
Solution Times (sec) for Type II, 400 Node, Capacitated and Uncapacitated Networks

Density	Capacity Density	Upper Capacitated Range	Number of Sources	Number of Transhipment Sources	Number of Sinks	Number of Transhipment Sinks	Total Cost Capacitated	Total Cost Uncapacitated	SUPERK Uncapacitated	Capacitated SUPERK	Capacitated TNE	Capacitated Boeing	Capacitated SHARE
.006	.20	16000-30000	4	0	12	0	56,655,355	48,247,790	3.068	3.851	12.787	31.924	16.703
.006	.40	16000-30000	4	0	12	0	56,958,135	48,133,540	2.715	3.608	10.419	16.605	15.059
.006	.80	16000-30000	4	0	12	0	88,018,190	52,716,108	3.276	5.910	18.776	25.641	24.557
.014	.20	16000-30000	4	0	12	0	34,708,109	34,181,777	4.108	4.570	24.079	50.851	20.361
.014	.40	16000-30000	4	0	12	0	41,347,558	39,778,617	3.460	3.794	20.095	20.823	16.436
.014	.80	16000-30000	4	0	12	0	49,968,189	37,371,180	3.812	7.124	30.911	28.180	28.222
.006	.20	16000-30000	3	5	10	50	64,114,945	62,509,006	4.414	4.766	NA	22.489	DNR
.006	.40	16000-30000	3	5	10	50	70,169,798	57,324,077	4.427	5.022	NA	21.077	DNR
.006	.80	16000-30000	3	5	10	50	71,274,141	53,212,018	5.005	6.377	NA	30.162	DNR
.014	.20	16000-30000	3	5	10	50	27,143,995	26,510,207	6.614	7.084	NA	66.956	DNR
.014	.40	16000-30000	3	5	10	50	22,283,234	21,517,178	6.355	6.921	NA	50.904	DNR
.014	.80	16000-30000	3	5	10	50	31,495,617	30,196,650	6.261	8.300	NA	38.825	DNR
.006	.20	16000-30000	8	0	60	0	48,592,205	45,065,825	4.717	4.389	NA	49.567	18.518
.006	.40	16000-30000	8	0	60	0	60,671,131	58,785,106	4.313	4.982	NA	28.216	20.160
.006	.80	16000-30000	8	0	60	0	59,579,090	55,043,731	4.548	5.363	NA	24.990	20.345
.014	.20	16000-30000	8	0	60	0	25,042,858	24,483,843	6.109	6.288	NA	46.352	24.049
.014	.40	16000-30000	8	0	60	0	32,514,425	31,617,938	6.500	6.676	NA	30.630	24.992
.014	.80	16000-30000	8	0	60	0	32,318,655	29,115,850	6.018	6.167	NA	43.040	25.663
.006	20	20000-120000	4	0	12	0	57,574,930	57,440,850	2.929	3.255	10.682	33.896	14.374
.006	.40	20000-120000	4	0	12	0	62,512,025	62,415,905	3.051	3.087	12.026	15.540	12.690
006	.80	20000-120000	4	0	12	0	69,186,437	66,909,631	2.798	3.665	13.826	6.732	16.791
.014	.20	20000-120000	4	0	12	0	36,588,701	35,513,439	4.068	4.528	23.318	38.426	19.042
.014	.40	20000-120000	4	0	12	0	18,318,326	18,318,326	3.026	3.084	15.191	24.629	10.774
.014	.80	20000-120000	4	0	12	0	43,857,362	38,244,163	4.707	5.162	26.286	25.128	21.540
.006	.20	20000-120000	3	5	10	50	41,507,356	38,921,644	4.361	4.482	NA	45.043	DNR
.006	.40	20000-120000	3	5	10	50	54,133,499	53,834,023	4.444	4.464	NA	20.261	DNR
.006	.80	20000-120000	3	5	10	50	56,657,091	55,016,208	4.979	5.279	NA	25.288	DNR
.014	.20	20000-120000	3	5	10	50	32,852,588	32,852,588	5.987	5.987	NA	52.877	DNR
.014	.40	20000-120000	3	5	10	50	29,235,379	29,235,379	6.233	6.240	NA	24.029	DNR
.014	.80	20000-120000	3	5	10	50	30,696,664	29,181,486	6.107	6.296	NA	28.215	DNR
.006	.20	20000-120000	8	0	60	0	46,871,755	46,214,579	4.509	4.451	NA	39.146	16.609
.006	.40	20000-120000	8	0	60	0	53,862,763	53,860,213	5.030	5.155	NA	32.052	19.131
.006	.80	20000-120000	8	0	60	0	56,219,472	55,304,779	4.700	4.850	NA	27.647	20.045
.014	.20	20000-120000	8	0	60	0	28,651,215	28,651,215	5.328	5.321	NA	42.451	21.405
.014	.40	20000-120000	8	0	60	0	31,705,319	31,663,210	6.484	6.590	NA	32.635	26 441
.014	.80	20000-120000	8	0	60	0	27,808,662	27,799,414	6.087	5.955	NA	32.443	DNR

NA - would not fit into the computer memory available
DNR- Did not run

in these two different situations. The uncapacitated problems in Tables 6 and 7 are further distinquished from those in Table 5 by virtue of containing transhipment sources and sinks as well as pure sources and sinks.

The fundamental inference to be drawn from the data in Table 5 is that SUPERK dominates the other codes regardless of problem size, problem type, density, and number of sources and sinks. This table also shows that the solution time increases for each problem type as problem size and/or density increases.

The solution times for type I networks possess a remarkable consistency that permits the codes to be uniformly ranked across these problems in terms of their relative efficiencies. In particular, the ranking of the codes in descending order of efficiency is SUPERK, Boeing, TWB, and SHARE. Several differences between these results and those for transportation problems are also apparent. The superiority of SUPERK over the Boeing code has dropped from about eight times faster on the transportation problems to twice as fast on the Type I networks; also the Boeing code has switched from being the least efficient to the most efficient of the codes other than SUPERK.

A quick glance at the Type II network times discloses that the rankings of the Boeing, TWB, and SHARE codes in terms of efficiency are reversed on these problems. Further the superiority of SUPERK over the best of the other codes is increased to a factor of 4.

The 500 node problems provide some useful insights both about the problem types and the codes. Type I and II 500 node problems have the same density, total supply, cost range, number of sources, and number of sinks. Comparing the percentage changes in solution time from the Type I problems to the Type II problems, the SHARE times changed the least, followed by SUPERK; in both cases the magnitude of this change is roughly

200%. The data of Table 1 give an idea of where the additional time required to solve the Type II networks was spent. This table shows that the number of breakthroughs and cycle lengths (number of arcs in the flow augmenting path) for the two types of networks are essentially equal. (Note that density does not seem to affect these statistics.) The big difference between the problem types is that the Type II networks require the generation of twice as many labels. Thus, approximately twice as many arcs must be examined in the labeling process. Further, Table 1 discloses that the average number of nodes labeled per breakthrough is approximately 100 for the Type I problems and 200 for the Type II problems. This increase in the number of labeled nodes essentially doubles the number of arcs to be examined in determining the sets S and R. In essence, these results indicate that solution times of SHARE and SUPERK are proportional to the number of arcs examined during the solution process. From this it may be concluded that the greater efficiency of SUPERK is due in large measure to the partition which reduces the number of arcs considered during labeling by nine-tenths and during the determination of S and R by one-tenth. Additional efficiency is evidently contributed by the label change and the recycling of S and R. The large fluctuation in the TWB and Boeing codes suggests that these codes involve some form of search among the arcs, either during the labeling process to identify arcs leading into and out of nodes in the labeled tree or during the potential change process to identify arcs that compose the sets S and R.

The statistics in Tables 1, 2 and 5 that the Type II problems are much harder to solve. (Correlated with this difficulty is the interesting fact that their optimal objective function values were twice as large as their corresponding Type I problems.) This solution difficulty is due to the complex paths between sources and sinks and due to the inability

to reach all sinks from any given source. The effect of this structure, as evidenced in Tables 1 and 2, is to greatly increase the number of labels generated and the number of potential changes made. These two things increase solution time respectively by requiring more arcs to be examined for inclusion in the cut set and by requiring more cut sets to be determined.

The data in Table 5 for the Type II problems discloses a consistent increase in the median solution time (holding density constant) as the number of sources and sinks increase. The same effect may be observed in Tables 6 and 7 for the uncapacitated solution times when an equal number of transhipment sources and sinks are introduced. Thus, in fact, the addition of transhipment sources and sinks appears to increase solution times in a fashion identical to increasing the number of pure sources and sinks. However, it seems likely that the effect of adding sources and sinks must at some point reverse itself and begin to decrease solution time. This conjecture comes from comparing the solution times of the 50 x 50 transportation problems to those of 100 node network problems, holding density roughly constant. Of these two classes of networks, both of equal size, the transportation problems - which consist only of source and sink nodes- are easier to solve. (Note that this result holds for all codes.)

The last part of our analysis on the uncapacitated network problems deals with testing advance dual start procedures. Two procedures were tested. The first used Dijkstra's [6] shortest path algorithm to find the shortest path from the master source to all other nodes using arc costs as "distances". Once this tree of shortest paths was determined, the associated node potentials were used to calculate the starting

marginal costs. This starting procedure is identified as version I of the "advance dual start" heading in Table 5. Version II is essentially the same except that the shortest path from master source to master sink is saturated with flow and these flow values along with the node potential values were used to initiate the algorithm. Both advanced start procedures were tested only using SUPERK. The results, which are indicated in Table 5 include the time to find the advance start in the total solution time. The outcome is somewhat surprising. The procedures yield on small problems slightly improved solution times or less but actually impede the solution times on large problems (500 nodes or more). The main reason for this negative effect seems to be that the node potentials initially determined cause a large number of nodes to be labeled eligible on subsequent label-outs. This in turn causes large trees to be generated, increasing the computational effort during labeling and cut set determination. (The start procedures may be subject to some improvement, however, through improved coding.)

The foregoing results with the findings of the studies by Srinavasan and Thompson [31] and by Glover, Karney, Klingman, and Napier [15] on special-purpose primal transportation codes; that is, if the central solution algorithm is coded to perform its basic functions with maximum efficiency then the computational time required to find a superior start is often not compensated by its savings. However, advance start procedures are often helpful when performing hand calculations because of the inefficiency of humans in carrying out the basic algorithmic steps.

3.3 CAPACITATED MINIMUM COST FLOW NETWORKS

The last type of network problems examined were capacitated networks.

While non-circulatory networks acquire capacitation on the arcs which are added to make them circulatory, we are using the term "capacitated network" to refer to networks that have capacity restrictions before the addition of such arcs. All arcs of the capacitated networks in this study have lower capacities of zero and upper capacities lying in the ranges indicated in Tables 6 and 7. The basic structural variation of the networks in these tables is specified as follows:

1. Two problem sizes were examined, consisting of 100 nodes and 400 nodes.
2. Each problem size has two distinct densities. The 100 node problems have densities of .03 and .24, and the 400 node problems have densities of .006 and .014. For each of these densities 18 problems were generated.
3. The percentage of the arcs which are capacitated from above (capacitated density) varied from .20 to .80.
4. Two distinct upper capacity ranges were set for the 100 and 400 node problems. In each case, one of the capacity ranges was purposely set quite large and the other somewhat smaller.
5. Three distinct source and sink node-sets were used.

The lock-step parameter variation of these problems is used to enable closer examination of the effects of specific variables while holding other variables fixed, as exhibited in the organization of the data in Tables 6 and 7. Between each set of horizontal lines in these tables, the source and sink node configuration is constant, with the only varying parameters being density and capacitated density. Also, on either side of the double horizontal lines are sets of problems whose only differing parameter is the upper capacitated range.

The following conclusions can be drawn from this data: solution time consistently increases with density; capacitated density has an erratic time effect, and restricting the capacity range and/or increasing the number of nodes generally increases solution time.

The role of density has been visible throughout all of the preceding computational interpretations. The capacitated problems, as well as the uncapacitated problems, are subject to substantial increases in solution times as density grows; in fact, for the TWB and Boeing codes, the effect is of major consequence. In one case the problems cannot be solved and in the other, solution times increase many-fold.

The erratic effect of capacitated density is believed to be directly connected to the "non-extreme point" solution approach of the out-of-kilter algorithm. From the point of view of a simplex-type procedure, as the number of capacitated arcs are increased, the number of feasible extreme points will (most likely) increase, and thus (probably) increase the number of pivots required to reach optimality. However the results of our computational experience with the out-of-kilter codes indicate that the right combination of capacities can actually reduce the solution time, as was the case in 11 of the 72 capacitated problems run with SUPERK. (Of course, most of the capacitated problems had different optimal solutions than their uncapacitated counterparts, as indicated by their objective function values.)

Basically, solution time appears to increase when the problem becomes more constrained (e.g., as objective function values increase, or the capacity range is restricted). For instance, in the third and sixth problems of Tables 6 and 7, there is a large difference in the objective function values when 80% of the arcs are capacitated. Also, in com-

parable problems for the two capacity ranges, the more restricted problems generally yield higher solution times. The SUPERK and SHARE data support this conclusion more strongly than the heavily mixed results of the TWB and Boeing codes.

Increasing the number of nodes, while holding the number of arcs constant, induces a substantial change in solution time. For instance, the .014 dense, 400 node problems of Table 7 have approximately the same number of arcs as the .24 dense, 100 node problems of Table 6. Comparing solution times, we find a three-fold difference. This change is more dramatic than the change produced by increasing the density of the 100 node problems from .03 to .24. (See Table 6.) Also, the data in Tables 5 and 7 tend to indicate that as the number of source and sink nodes increase, solution time increases, although Table 6 does not particularly substantiate (or refute) this conclusion.

All network problem sets were examined to determine whether any patterns existed in the number of breakthroughs and potential changes. Our overall observation is that as density increases (for a fixed problem size), the number of breakthroughs increases,the number of potential changes decreases, and solution time increases. (The large changes in the objective function values in Tables 6 and 7 induced by changes in density are indicative of the decrease in the number of potential changes.) The increase in the number of breakthroughs tends to place a burden on the labeling section of the codes and, at the same time, the determination of the cut sets becomes quite laborious with a large number of labeled nodes. This is true in spite of the fact that fewer potential changes are required to solve a problem "type" in which cost ranges, total supply, and total num-

ber of nodes are kept fixed. Except for the indicated connections to density, the out-of-kilter method is quite unpredictable with respect to the number of breakthroughs and potential changes performed.

As on the earlier problems, SUPERK strictly dominates the other codes for the data in Tables 6 and 7, and the superiority of SUPERK increases as the problem size and/or density increases. Among the other codes, SHARE is the most consistant and appears to be slightly more efficient.

3.4 MEMORY REQUIREMENTS OF THE CODES

An important factor in evaluating computer codes of this type is the size of problems which can be solved. Each of the codes investigated is an "in-core" code; that is, the program and all problem data must be stored within available memory for direct access. (Virtual memory schemes could be used to effectively give "in-core, out-of-core" features for large versions of any of the codes; however, other solution procedures would seem to be more amenable to designs of this type [15].)

The memory requirements for the main program and associated system routines for each of the codes is approximately the same, ranging from $14.2K_8$ 60-bit words for TWB to $17K_8$ 60-bit words for SUPERK, however, the problem storage requirements vary considerably. The SHARE code requires six node-length plus seven arc-length vectors to store a given problem, and Boeing uses six node-length and eight arc-length arrays. SUPERK necessitates only four node-length vectors, but uses eleven arc-length arrays. (The authors have determined that the number of arc arrays can be reduced to ten with minor programming changes, and to eight if auxiliary memory is used.) Although the original algorithm on which the TWB code was based [2] used four node-length and seven arc-length arrays,

program modifications, which were made for speed enhancement, involved the addition of two node-length and seven arc-length arrays plus a matrix whose dimensions are number-of-nodes by maximum number of arcs touching any given node. These modifications greatly restrict the size of problems which can be run, as reflected in the limited number of test problems which could be processed by the TWB code in a $220K_8$ field length.

These figures indicate, as might be expected, that SUPERK's computational efficiency is not without its price in terms of memory space. The increased number of arc-length arrays can become restrictive as the problem density increases, however, partially offsetting this is the fact that the relatively smaller number of node-length arrays is of value in the very sparse networks often found in industrial applications.

Footnotes

[1] In this connection, it should be remarked that the CDC 6600 has two FORTRAN compilers. One of these compilers, the FTN compiler, has three levels for optimization of the object code. This compiler produces object code which executes from 3 to 15 times faster than the alternative RUN compiler, which was the one available to us. Thus, the reported times should be substantially faster using the FTN compiler; however, we believe that the changes would be uniform throughout all the codes tested.

[2] These times for the SHARE code are much slower than the SHARE times reported in [15] for the same problems. The new times, given in Table 3, were the result of re-solving the original problems with the same code on the same machine during the course of preparing this paper. Since the new times were substantially slower, the code was carefully checked and the problems solved twice more with the same results. The only possible explanation we have for the difference is that the University of Texas operating system has been substantially altered during the three years between the two studies.

[3] The code developed by Srinivasan and Thompson is comparable in efficiency to the primal code of [15] for uncapacitated problems of 100% density; however the Srinivasan and Thompson code is not designed to accommodate capacitated problems or to take advantage of sparseness.

[4] Graves and Thrall [30] have specialized the out-of-kilter algorithm to transportation problems. Computational tests of this method conducted by Lee [21] indicate that 10 x 50 transportation problems require about 3 seconds to solve on the IBM 360/65. Our tests using a different computer and problem set indicate that 50 x 50 problems are solved by SUPERK in .84 seconds.

References

1. Balinsky, M.L. and R.E. Gomory, "A Primal Method for the Assignment and Transportation Problem," Management Science, 10(1964), 578-593.

2. Briggs, William A. Algorithm 248 - Netflow (H), Communications of the ACM, Vol. 8, No. 2, February, 1965.

3. Clasen, R. J. "The Numerical Solution of Network Problems Using the Out-of-Kilter Algorithm," RAND Corporation Memorandum RM-5456-PR, Santa Monica, California, March, 1968.

4. Dantzig, G.B. Linear Programming and Extensions Princeton, N.J.: Princeton University Press, 1963.

5. Dennis, J.B. "A High-Speed Computer Technique for the Transportation Problem," Journal of Association for Computing Machinery, Vol. 8, (1958), 132-153.

6. Dijkstra, D.W. "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik, 1 (1959), 269-271.

7. Flood, M.M. "A Transportation Algorithm and Code", Naval Research Logistics Quarterly, Vol. 8 (1961), 257-276.

8. Ford, L.R., Jr., and D. Fulkerson, Flows in Networks, Princeton, N.J.: Princeton University Press, 1962, 194.

9. Ford, L.R. and D. Fulkerson, "A Primal-Dual Algorithm for the Capacitated Hitchcock Problem", Naval Research Logistics Quarterly, Vol. 4, No. 1 (March 1957) 47-54.

10. Frank, Howard and Ivan T. Frisch, Communication, Transmission, and Transportation Networks Reading, Mass: Addison-Wesley Pub. Co., 1971.

11. Fulkerson, D.R. "Flow Networks and Combinatorial Operations Research", American Mathematical Monthly, Vol. 73, No.2 (February 1966), 115-138.

12. Fulkerson, D.R. "An Out of Kilter Method for Solving Minimal Cost Flow Problems", J. Soc. Indust. Appl. Math, Vol. 9 (1961)18-27.

13. Geoffrion, A.M. (ed.) Perspectives on Optimization: A Collection of Expository Articles, Reading, Mass: Addison Wesley, 1972.

14. Glickman, S., L. Johnson and L. Eselson, "Coding the Transportation Problem," Naval Research Logistics Quarterly, 7 (1960),169-183.

15. Glover, Fred, D. Karney, D. Klingman, and A. Napier, "A Computation Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," To appear in Management Science.

16. Glover, Fred and D. Klingman, "Double-Pricing Dual and Feasible Start Algorithms for the Capacitated Transportation (distribution) Problem," University of Texas at Austin, 1970.

17. Glover, Fred, D. Karney, and D. Klingman, "The Augmented Predecessor Index Method for Locating Stepping Stone Paths and Assigning Dual Prices in Distribution Problems," Transportation Science, Vol. 6, No. 1 (1972), 171-180.

18. Grigoriadis, M. and W. Walker, "A Treatment of Transportation Problems by Primal Partition Programming," Management Science, Vol. 14, No. 9 (May 1968), 565-599.

19. Hu, T.C. Integer Programming and Network Flows, Reading, Mass: Addison-Wesley Publishing Co., 1969.

20. Klingman, D. and J. Stutz, "Type II Network Generator", Center for Cybernetic Studies research report, forthcoming.

21. Lee, S. "An Experimental Study of the Transportation Algorithms", Master's Thesis, Graduate School of Business, University of California at Los Angeles, 1968.

22. Mueller, Merbach-Heiner, "An Improved Starting Algorithm for the Ford-Fulkerson Approach to the Transportation Problem," Management Science, Vol. 13, No. 1 (Sept. 1966), 97-104.

23. Napier, A. "A Computer Generator for Uncapacitated Networks," unpublished report.

24. "OPHELIE II: Mathematical Programming System." Control Data Corporation, Minneapolis, Minnesota, 1970.

25. "OPHELIE/LP: Linear Programming Subsystem of OPHELIE II." Control Data Corporation, Minneapolis, Minnesota, 1970.

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

27. Pla, Jean-Marie, "An Out-of-Kilter Algorithm for Solving Minimum Cost Potential Problems," Mathematical Programming, 1 (1971) 275-290.

28. Price, W. L. Graphs and Networks: An Introduction, Princeton, N.J.: Auerbach, Pubs., 1971.

29. Simonnard, Michel, Linear Programming, Translated by William S. Jewell Englewood Cliffs, N.J.: Prentice-Hall, 1966.

30. Spivey, W. A. and R. M. Thrall, Linear Optimization, New York: Holt, Rinehart and Winston, 1970.

31. Srinivasan, V. and G. L. Thompson, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm," Management Sciences Research Report No. 229, Graduate School of Industrial Administration, Carnegie-Melton University, Pittsburgh, Pennsylvania, December, 1970.