Implementation and Computational Comparisons of Primal, Dual and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems

F. Glover
University of Colorado
Boulder, Colorado

D. Karney
D. Klingman
University of Texas
Austin, Texas

ABSTRACT

This paper presents extensive computational experience with a special purpose primal simplex algorithm. The performance is compared to that of several "state of the art" out-of-kilter computer codes. The computational characteristics of several different primal feasible start procedures and pivot selection strategies are also examined.

The study discloses the advantages, in both computation time and memory requirements, of the primal approach over the out-of-kilter method. The test environment has the following distinguishing properties: (1) all of the codes are tested on the same machine and the same problems, (2) the test set includes capacitated and uncapacitated transhipment networks, transportation problems, and assignment problems, and (3) problem sizes ranging from 200 to 8,000 nodes with up to 35,000 arcs are examined.

1.0 INTRODUCTION

1.1 Scope of the Computational Analysis

Over the years a great deal of code development and computational testing [1,2,4,5,7,9,12,13,17,19,20,21,22,23,25,27,28] has been performed on transportation and minimum cost flow network problems. Primarily this development and testing has been confined to transportation codes, and most of the more recently developed codes (since 1960) are based on primal-dual algorithms of the out-of-kilter genre [1,2,4,7,9,17,23,25,28]. Surprisingly, the literature does not seem to report any code development or computational testing of a simplex based minimum cost flow network code. No doubt, this is due in part to the conclusion of

Networks, 4: 191-212
© 1974 by John Wiley & Sons, Inc.

Dantzig [5] and Ford and Fulkerson [8] that for hand calculations, the out-of-kilter method is better than the MODI, or Row-Column Sum Method [3,5] for transportation problems. Also the finding of [7] that a primal-dual ("non-simplex") code was superior to MODI codes, helped to shift code development away from specializations of the simplex algorithm.

The recent challenge to the supposed superiority of such primal-dual approaches in the favorable computational comparison of the primal simplex based transportation code [12] against the out-of-kilter codes of SHARE, Boeing, and the Texas Water Development Board on transportation problems (for which the primal transportation code proved to be faster by a factor of approximately 10), has led us to develop additional, more general, network codes in order to obtain more adequate comparisons of simplex and non-simplex procedures on problems other than transportation problems. The first step in this direction was the development of a new version of the out-of-kilter code, called SUPERK, reported in [1]. The second step, which constitutes our current focus, has been the development of both primal and dual simplex-based codes for general network flow problems, and the testing of these codes against the out-of-kilter methods previously developed.

The main purposes of this paper are to report this development and to discuss the computation comparison of these codes against the codes of Bennington [2], Boeing, General Motors, SHARE [4,25], SUPERK [1], Texas Water Development Board, and TRANS [12]. The test environment has the following distinguishing properties:

1. All of the codes are tested on the same machine (CDC 6600), the same problems, and computer jobs were executed during periods when the machine load was approximately the same.
2. The test set includes capacitated and uncapacitated minimum cost flow networks, transportation problems, and assignment problems.
3. Problem sizes ranging from 200 to 8,000 nodes with up to 35,000 arcs are examined.
4. Any researcher wishing to test his codes on the same problems may easily do so by simply obtaining a copy of the machine independent network generator [20] used to create the problems.

In addition to the computational comparison using the CDC 6600 we solved a subset of the test problems using the primal network codes on an IBM 360/65, a UNIVAC 1108, a CDC 6400, a CDC 3100, and a BURROUGHS 4704 in order to gain some insight on how such a code would perform on different machines.

1.2 Major Highlights of the Computational Analysis

The special purpose primal simplex network codes using streamlined procedures for updating the basis tree [11,16] turn out to be substantially superior to the dual simplex code and the non-simplex codes tested both in computational times and memory requirements. To illustrate, the primal network codes require only 1/3 to 2/3 the memory space for data of the non-simplex codes, enabling it to solve much larger problems. Further, the primal network codes were never equalled in computational efficiency by any of the other codes on the broad range of minimum cost flow network, transportation, and assignment problems tested. For example, the primal network codes were at least four times faster than the SHARE, Texas Water Development Board, General Motors, Boeing, and Bennington codes. The primal codes were also found to be at least 100 times faster than the state of the art large scale L.P. code, OPHELIE/LP. The primal method's median solution time on 1,500 node networks with 4,342 to 5,730 number of arcs is 17 seconds on a CDC 6600 computer using the RUN compiler (which is classed as 3 to 15 times slower than the FTN compiler). The largest problems solved were a 5,000 node, 35,000 arc problem and an 8,000 node, 15,000 arc problem with solution times of 384 seconds and 245 seconds, respectively, on the CDC 6600.

Another finding, which seems unusual in view of the marked superiority of primal simplex codes, (but which accords with the studies of [12,13] on transportation problems) is that the non-simplex network codes are superior to the dual simplex network code.

The study shows in addition that the solution times of the primal network code do not fluctuate on different computers in accordance with commonly believed variations in machine capabilities. For instance, the code runs only 10% slower on a UNIVAC 1108, only 12% slower on an IBM 360/65, only 20 times slower on a BURROUGHS 4704, and only 10 times slower on a CDC 3100. These results are presumed chiefly to be due to differences in the FORTRAN compilers available on the machines. However, they serve to reinforce the finding of [12] that any conclusion about the relative efficiencies of different codes based on running times involving different computers, compilers, or problems, must be more speculative than substantive.

The study also reveals that the primal network codes only run 10% slower on transportation problems than the primal transportation code [12] which is the fastest known transportation code. We have also conversely applied the primal transportation code [12] to network problems by first transforming these problems into equivalent transportation form using the methods of

[24,29]. By means of this device, the primal transportation code runs 25%-35% slower on network problems than the primal network code. Since the transformation of a network problem into a transportation problem greatly enlarges the problem (and since the network codes store only one more node-length array than the transportation code) we conclude that the best single code to have available is a primal network code.

In subsequent sections we expand upon these computational findings and also report the effect on solution times of varying problem structure (e.g., capacitation, the number of nodes, and the number of arcs).

2.0 PROBLEM STATEMENT AND TERMINOLOGY

A capacitated minimum cost flow network problem (transhipment problem) can be stated as follows:

Problem 1:

Minimize: $$\sum_{(i,j)\in A} c_{ij}x_{ij} \qquad (1)$$

Subject to: $$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = a_i,\ i \in N \qquad (2)$$

$$L_{ij} \leq x_{ij} \leq U_{ij},\ (i,j) \in A \qquad (3)$$

where N denotes the set of nodes, A denotes the set of arcs, a_i denotes the supply or demand at node i (with a nonzero supply being denoted by a negative a_i), $\sum_{i\in N} a_i = 0$, c_{ij} is the cost of shipping one unit of flow across the arc from node i to node j, and L_{ij} and U_{ij} are the lower and upper bounds on arc (i,j).

Some of the common terminology associated with transhipment problems is:

1. A *source node* is a node that only has arcs leading out of it. In addition, a source node usually has a nonzero supply associated with it.
2. A *sink node* is a node that only has arcs leading into it. A sink node usually has a nonzero demand associated with it.
3. A *transhipment node* is a node with arcs leading into it and leading out of it. If a transhipment node has a supply (demand) associated with it, it is called a *transhipment source node* (transhipment sink node). Otherwise, it is called a *pure transhipment node*.

4. A transhipment problem with no transhipment nodes is called a *transportation problem.*

5. A transhipment problem with no transhipment nodes, an equal number of source nodes and sink nodes, and in which the supply (demand) of each node is -1 (1), is called an *assignment problem.*

This terminology will be used in subsequent sections when presenting computational results. Note that none of the definitions requires the existence of every possible arc.

A set of $|N| - 1$ arcs (where $|N|$ denotes the cardinality of the set N) is a *basis* if each node of the network is an endpoint of at least one of these arcs and if no subset of these arcs constitutes a closed loop. This structure of the basis for Problem 1 is commonly known as the spanning tree structure [3,5,29]. An arc (and its associated variable x_{ij}) is called *basic* if it is contained among those arcs in the basis and is called *nonbasic* otherwise.

A basic solution is the unique assignment of values to the x_{ij} variables satisfying Equation (2) that results once each nonbasic x_{ij} has been set equal to L_{ij} or equal to U_{ij} (provided the value of the relevant bound is finite). If such a solution satisfies (3) for all of the variables, then it is called *primal feasible.*

Corresponding to a particular basis is a set of "node potentials" w_i (not unique) such that the "updated cost coefficients" π_{ij}, defined by $\pi_{ij} = - w_i + w_j - c_{ij}$ for each arc $(i,j) \in A$, is zero for all basic variables; a basic solution is dual feasible if in addition $\pi_{ij} \leq 0$ for all nonbasic variables set equal to L_{ij} and $\pi_{ij} \geq 0$ for all nonbasic variables x_{ij} set equal to U_{ij}. Using the triangularity (spanning tree) property of the basis, it is quite simple to determine the value of the basic variables once the value of the nonbasic variables are set and similarly to determine the node potentials and updated costs [11,16,3,5]. The details of such determinations are, however, unimportant to our present concern.

Since the node potentials on which the π_{ij} are based represent values assigned to the variables of the linear programming dual of the transhipment problem, their determination is called *pricing-out* the basis. By fundamental linear programming theory, a basic solution that is both primal and dual feasible is optimal for the transhipment problem.

The standard primal simplex approach to solving a transhipment problem is to obtain a primal basic "feasible" solution. (Such starting solutions normally involve artificial arcs since, in general, a one-pass procedure for obtaining a primal basic feasible solution to a transhipment problem has not been devised.) Once a starting basis has been obtained, the basis is priced-out and the updated costs are successively computed and scanned until a dual infeasiblity is found or optimality is determined. Once a dual infeasiblity is found, a pivot is made which brings some dual infeasible arc into the basis. This change of basis modifies the flow on the arcs in the loop formed in the old basis graph when the entering nonbasic arc is added to it. The basis arcs in this loop are normally referred as the *basis equivalent path* of the entering arc. The change of basis also modifies the node potentials in one of the subtrees created when the arc leaving the basis is deleted from the old basis graph.

The standard dual simplex approach [14] for solving a transhipment problem begins by obtaining a dual feasible solution. (In contradistinction to the primal approach, this can easily be done using the procedure of [15].) Once a starting basis has been obtained, the flow on each nonbasic arc is set equal to its lower bound and the flows on the basic arcs are determined using the triangularity property of the basis. These flows are then successively scanned until a primal infeasiblity is found or optimality is determined. Once a primal infeasiblity is determined, a pivot is made which brings into the basis, a particular nonbasic arc which has a nonzero coefficient in the unique updated linear equation which expresses this basic variable as a linear combination of the current nonbasic variables (for a complete description see [14]). The important aspect is that the coefficients of this linear equation must be calculated and further that these coefficients can be calculated by re-pricing-out the basis with pseudo-costs of zero on all basic arcs except for the basic arc leaving the basis. Its pseudo-cost is set equal to one. Using the new pseudo-node potentials, the coefficients of the linear equation for each nonbasic arc (i,j) is equal to $- w_i + w_j$ (again for a more complete description of this procedure, see [14]). The determination of the pseudo-node potentials and coefficients of the linear equation is called *double pricing*. Having determined the come-in arc, the pivot is executed by finding the basis equivalent path of the come-in arc and modifying the flows of the basic arcs appropriately.

3.0 DEVELOPMENT OF THE SPECIAL PURPOSE PRIMAL SIMPLEX CODE BY SUBROUTINE

3.1 Overview

The computer code was written in FORTRAN IV, is an in-core code, and was initially tested using the run compiler on a CDC 6600 with a maximum memory of 130,000 words. In this code, a transhipment problem with N nodes and A arcs (without exploiting the word size of the machine) requires 6N + 2A + 10,000 words for uncapacitated problems and 6N + 3A + 10,000 words for capacitated problems. It would be possible by exploiting the fact that the costs, flows, node numbers and node potentials are integer-valued, to store more than one per word and in this manner reduce these storage requirements. However, our purpose was to develop a code whose capabilities did not depend on the unique characteristics of a particular computer (e.g., word size, etc.). The obvious advantage of this approach is the ease with which it enables the code to be tested on different machines. Further, we used a "manilla" FORTRAN IV so that recoding to fit differing machine conventions would be minimized.

Within these constraints, we tried to minimize our storage requirements, at the same time making sure the code could solve the "thoroughly general" transhipment problem. For example, we designed the code to allow multiple arcs between the same nodes and to handle arbitrarily capacitated problems. (Thus, for example, the code will accommodate a piecewise linear convex cost minimization.) Since any lower and upper capacitated transhipment problem can be transformed into an equivalent transhipment problem with lower capacitates of zero, we perform this translation of variables upon inputing the problem in order to eliminate an arc length array of data.

The program consists of a main program and ten subroutines, and may be conceptually depicted as in boxes 1-4 in Figure 1. Subdividing the program into many different subroutines made it possible to test numerous variations without extensive recoding. However, this subdivision inevitably slowed the code somewhat by requiring the computer to process subroutine calls and returns rather than jump instructions.

The total time spent in each subroutine was recorded by calling a Real Time Clock (accurate to a millisecond) upon entering and leaving that subroutine. A count was also made of the total number of pivots performed. In Table I, we report median values (with five problems per group) for the total solution time (exclusive of input and output), the start time (time spent in box 1), the number of pivots, the total pivot time (total time spent in boxes 2, 3, and 4), and the average pivot time (total pivot time divided by the number of pivots).

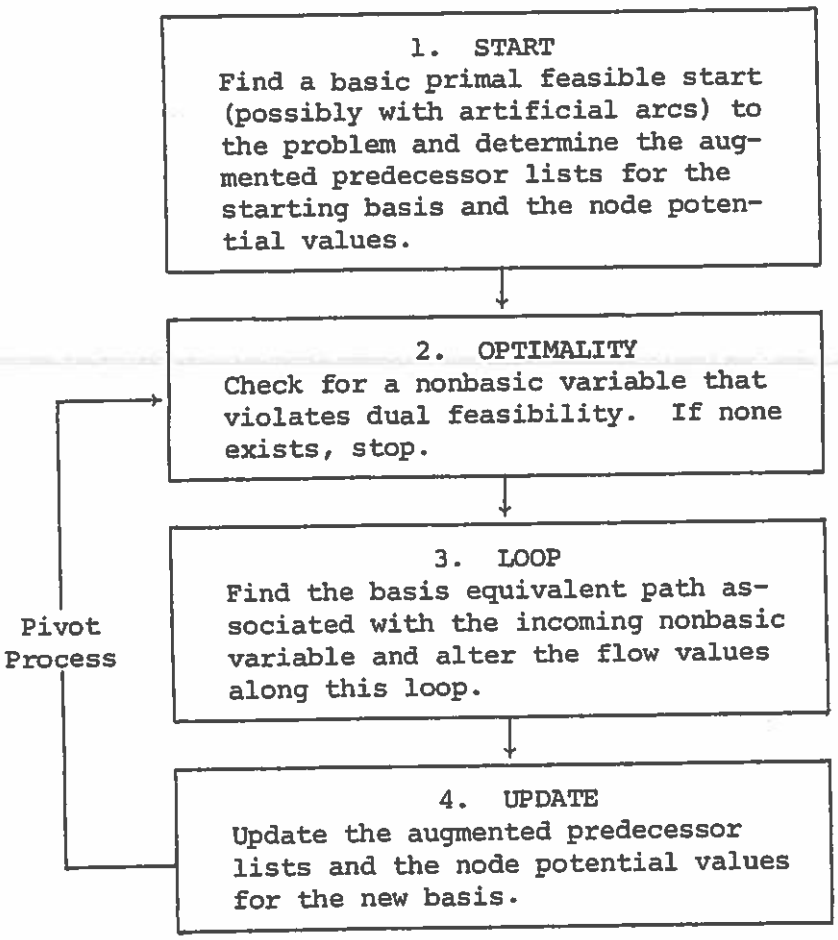

Fig. 1 Flow Diagram for the Primal Network Code.

3.2 Start Algorithm

A variety of start procedures exist for transportation problems and most of these procedures have been computationally tested [3,5,12,19,22,27,29]. The studies by [12,19,27] found that the Modified Row Minimum Rule was the best start in terms of total solution time for both totally dense and non-dense square transportation problems. In addition, the studies [17, 19,27] indicate that this start is also quite good for rectangular transportation problems.

Since no one-pass basic primal feasible starting procedure exists for transhipment problems (or for non-dense transportation problems) and since the studies [12,17,19] indicate that the Modified Row Minimum Rule yields good artificial basic primal

"feasible" starting solutions, we undertook to modify this starting procedure to handle transhipment problems. This modification was accomplished in the following manner.

The Modified Row Minimum Rule for transportation problems scans each row of the transportation tableau sequentially. Each time a row is scanned, the current minimum cost cell is selected (every cell is assumed to exist; thus inadmissible cells have a large cost) and a flow equal to the minimum of the current supply, demand, and upper bound associated with this cell is assigned to its variable. If the supply or demand associated with the cell is exhausted by the assignment then the cell is entered into the basis and the cells of the row or column (but not both) associated with the exhausted supply or demand are deleted from consideration. Then a new row is scanned. If the supply or demand is not exhausted, then the row is re-scanned to select a new cell for the basis. This rule differs from the Row Minimum Rule [12] in that it selects only one basic cell each time the row is examined, whereas the Row Minimum Rule continues to select row cells (on a single pass) until the total supply of the row has been exhausted.

One may easily verify that if a transhipment problem is written as a transportation problem using Orden's transformation procedure [24] (splitting each node into an origin node and a destination node, adding large "buffer" supplies and demands to the new nodes, and inserting a zero-cost arc from the "origin half" to the "destination half" of each split node), that the Modified Row Minimum Rule yields a starting basis for both the equivalent transportation problem and the transhipment problem if the costs are positive. While this result is not presented anywhere in the literature, it follows quite simply from the fact that the Modified Row Minimum Rule will first select as basic arcs the $|N|$ "inserted" arcs with zero cost since each such arc is uncapacitated, is the minimum cost arc in its row, and lie in different rows and columns.

For the above reasons, the first starting criterion that we coded *implicitly* transformed each transhipment problem into a transportation problem and applied the Modified Row Minimum Rule. Since this start yields a basis for both the transportation problem and the transhipment problem, we solved some transhipment problems twice, once using the transhipment code developed in this paper and once using the transportation code of [12] (applied to the problem after transforming it to the transportation structure). For both codes we used the Modified Row Minimum Rule and the same pivot criterion. To our surprise, the codes made exactly the same pivots. The transportation code required about 30% longer to solve these problems than the transhipment code. We examined other transportation start criteria

to see if they would also "naturally" yield a basis for transhipment problems in the manner provided by the Modified Row Minimum Rule. Unfortunately none of the standardly purposed procedures succeeds in giving such a start. Consequently, since no primal start procedures have been proposed in the literature the only other start procedures we tested were variations of the Modified Row Minimum Rule. None of the minor alternates that we tested reduced our total solution time. Consequently, to save space their description and solution times are not presented.

3.3 Pivot Criteria

An important factor influencing computational efficiency is the basis change criterion. The relevant tradeoffs for the basis change criterion involve time consumed in searching for a new arc to enter the basis and the number of pivots required to find an optimal solution (time per pivot versus total number of pivots).

Three different criteria for determining the variable to enter the basis were examined (box 2 of Figure 1) to ascertain their effect upon the total solution time.

The first criterion tested was the "Most Negative Evaluator Rule." This rule examines each arc and picks that arc to enter the basis whose dual constraint is violated by the largest amount.

The "Outward-Node First Negative Evaluator" was tested next. This method scans the arcs leading out of the nodes until it encounters the first arc whose updated cost signals that this arc is dual infeasible. This arc is then selected to enter the basis.

The last pivot procedure tested was the "Outward-Node Most Negative Evaluator." This criterion scans the arcs leading out of the nodes until it encounters the first node containing a dual infeasible arc, and then selects the outward arc of this node which violates dual feasibility by the largest amount to enter the basis.

The "Outward-Node Most Negative Evaluator" was found to be best from the standpoint of total solution time. (We have not itemized individual computation times in order to conserve space.) This result agrees with the findings of [6,12,27] on transportation problems. The rule is the one we initially conjectured would be best for transhipment problems since it was best on non-dense transportation problems [12].

In brief, the study discloses that the most efficient primal network code arises by coupling the primal algorithm with the Modified Row Minimum Rule and the Outward-Node Most Negative Evaluator. In subsequent sections, this code is referred to as PNET.

3.4 Alternative List Structures

The code used in the above computational testing employs the augmented predecessor index (API) method [11] for storing and updating a spanning tree. This structure was used because extensive testing [27] showed that the API method [11] is at least twice as fast as any previous purposed list structures for transportation problems. Subsequent to the above code development and computation testing, we developed a new list structure (for storing and updating the spanning tree information needed for executing pivots in a simplex algorithm) dubbed the augmented threaded index (ATI) method [16], which we felt would be more efficient than the API method [11]. Consequently, we modified subroutines 1, 3, and 4 of PNET in order to test the ATI method [16]. This code is called PNET-I. Computational results indicate that PNET-I is about 10% faster than PNET. This comparison is discussed in detail in Section 5.0. (Note that we simply used the conclusions on the best start and pivot procedures based on using the API method to develop PNET-I.)

4.0 DEVELOPMENT OF THE SPECIAL PURPOSE DUAL SIMPLEX CODE BY SUBROUTINE

4.1 Overview

A computer code embodying the ideas of the Section 2.0 and [14] was written in FORTRAN IV and tested on a CDC 6600 using the RUN compiler with a maximum memory of 130,000 words. To solve a capacitated transhipment problem with N nodes and A arcs (without exploiting the word size of the machine) this in-core code requires 3A + 9N + 11,000 words. For uncapacitated problems, it requires one less arc-length array. The code uses the augmented predecessor index method [14] to store and update the spanning tree.

The program consists of a main program and fifteen subroutines, and may be conceptually depicted as in boxes 1-5 in Figure 2. Subdividing the program into many different subroutines made it possible to test numerous variations without extensive recoding. However, this subdivision ineviatably slowed the code somewhat.

The total time spent in each of the boxes 1-5 was recorded by calling a Real Time Clock upon entering and leaving each of these functions. A count was also made of the total number of pivots performed. In Table I we report median values for the total solution time, the start time (time spent in box 1), the number of pivots, the total pivot time (total time spent in boxes 2-5), and the average pivot time (total pivot time divided

by the number of pivots) for the dual code and PNET. In addition, Table I contains the total time spent finding the nonbasic arc to enter the basis (total time spent in box 3) in the dual code.

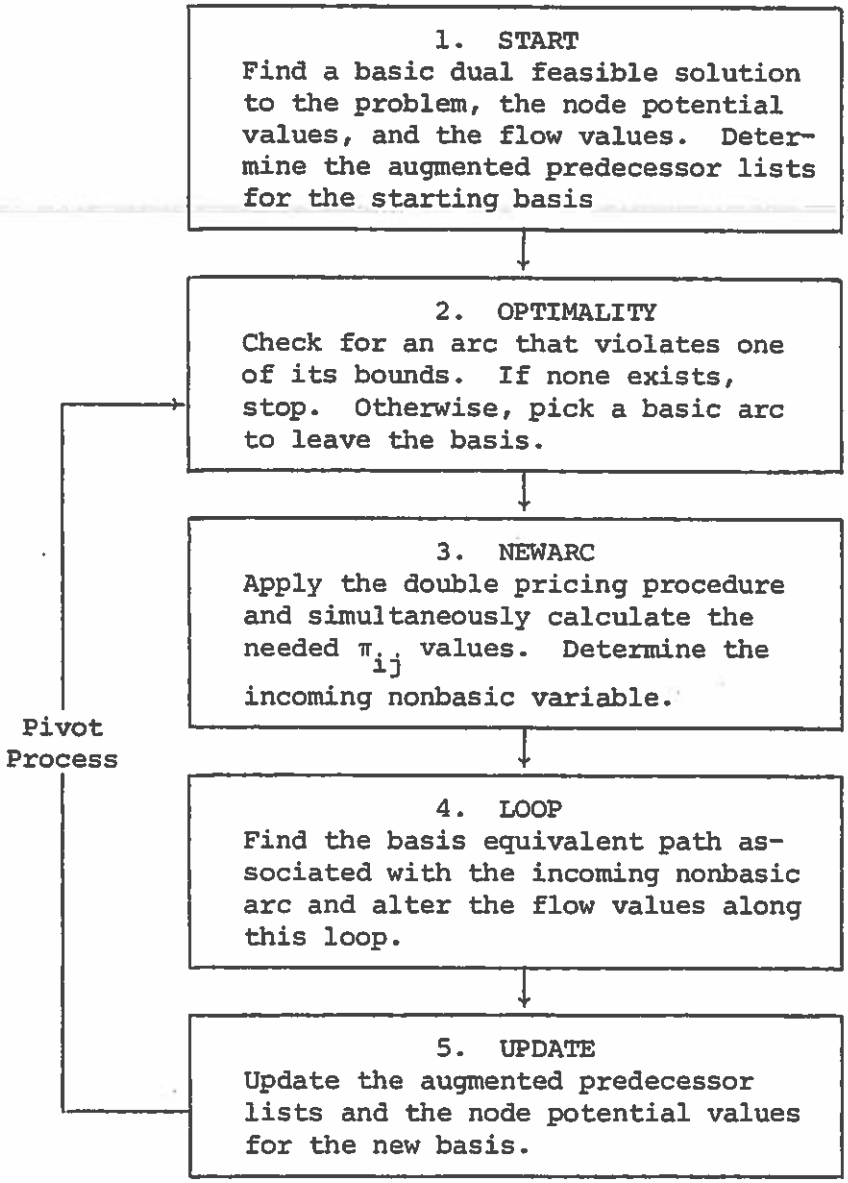

Fig. 2 Flow Diagram for the Dual Network Code.

TABLE I

COMPUTATIONAL RESULTS ON DUAL PIVOT CRITERIA AND PNET SOLUTION TIMES

Number of Nodes	Number of Arcs	MOST NEGATIVE CRITERION						MODIFIED FIRST NEGATIVE CRITERION					
		Solution Time	Start Time	Pivot Time	No. of Pivots	NEWARC*	Time/ Pivot	Solution Time	Start Time	Pivot Time	No. of Pivots	NEWARC*	Time/ Pivo
50	75	.104	.072	.032	6	.000	.005	.110	.077	.033	7	.000	.005
50	125	.231	.077	.154	27	.082	.006	.236	.78	.158	29	.048	.005
50	200	.260	.089	.171	29	.074	.006	.259	.087	.172	29	.073	.006
50	250	.405	.088	.317	44	.196	.007	.510	.097	.413	54	.179	.008
50	375	.604	.120	.484	57	.258	.008	.632	.127	.535	58	.249	.009
50	500	.588	.167	.421	41	.231	.010	.574	.162	.412	40	.232	.010
500	1500	21.604	4.451	17.153	410	8.580	.042	21.315	4.421	16.894	461	8.745	.037
500	2250	48.624	3.853	44.771	900	24.792	.050	40.499	3.812	36.637	861	20.787	.043
500	2750	35.781	3.870	31.912	975	30.369	.053	42.690	3.841	38.849	854	23.460	.045
500	3500	63.575	4.332	59.243	983	37.557	.060	46.184	4.429	41.735	818	26.852	.051
500	4250	69.401	4.628	64.773	992	43.631	.065	62.865	4.719	58.146	1053	40.163	.055
1000	2900	161.743	13.013	148.730	1741	70.314	.085	152.288	12.916	139.372	2126	78.500	.066
1000	3400	185.108	13.794	171.314	1866	86.636	.092	165.707	13.860	151.847	2147	85.058	.071
1500	4500	408.418	28.329	380.089	3008	186.491	.126	369.448	28.777	340.671	3603	193.655	.095

Number of Nodes	Number of Arcs	FIRST NEGATIVE CRITERION						PNET				
		Solution Time	Start Time	Pivot Time	No. of Pivots	NEWARC*	Time/ Pivot	Solution Time	Start Time	Pivot Time	No. of Pivots	Time/ Pivot
50	75	.118	.072	.046	9	.008	.005	.098	.022	.076	46	.002
50	125	.219	.074	.145	31	.048	.005	.078	.029	.049	37	.001
50	200	.289	.088	.201	37	.088	.005	.132	.040	.092	77	.001
50	250	.500	.096	.404	63	.219	.006	.182	.049	.133	121	.001
50	375	.558	.120	.438	62	.244	.007	.220	.048	.172	101	.002
50	500	.507	.167	.340	38	.244	.009	.294	.058	.236	110	.002
500	1500	23.671	4.425	19.246	595	12.358	.032	3.592	.337	3.255	594	.006
500	2250	25.889	3.843	22.046	852	15.473	.026	2.998	.392	2.616	910	.003
500	2750	33.311	3.845	29.466	1005	21.145	.029	4.489	.408	4.001	1246	.003
500	3500	40.269	4.396	35.873	1074	26.924	.033	5.450	.421	5.029	1779	.003
500	4250	38.872	4.675	34.197	907	26.775	.038	5.549	.446	5.103	1865	.003
1000	2900	117.599	12.944	104.655	2253	73.477	.046	9.961	.628	9.340	2758	.006
1000	3400	121.905	13.927	107.978	2188	77.257	.049	11.266	.672	10.594	3387	.003
1500	4500	245.956	28.396	217.560	3330	167.262	.065	21.542	.583	20.959	5051	.004

* Time to find the new arc entering the basis.

4.2 Start and Pivot Criteria

In developing the dual code only the dual start procedure developed in [15] was tested.

Three different pivot criteria were tested for picking the variable to leave the basis. These criteria were the first negative criterion, modified first negative criterion, and most negative criterion. The relevant tradeoffs for the basis change criteria involve the time consumed in searching for the variables to enter and leave the basis versus the number of pivots required to obtain an optimal solution (time per pivot versus total number of pivots)

The most negative criterion examines each basic variable and picks that variable to leave the basis which violates its (upper or lower) bound by the largest amount.

The modified most negative criterion scans the basic arcs out of a node until it encounters the first node containing a basic variable which violates a bound, and then selects the arc out of this node with the greatest violation. The search is then resumed on the next pivot with the node following the node of the last pivot.

Over 100 problems were studied using these criteria. Table I contains the typical results of testing these criteria on 50, 500, 1,000, and 1,500 node uncapacitated networks. (Cost ranges on these problems are 1-100 and the total supply is equal to 1,000 times the number of nodes.) Our results strongly indicated that the first negative criterion was best for most problem sizes. This was quite surprising since similar tests for the primal network code showed the modified first negative criterion to be preferred.

Table I shows that more pivots are required on the 1,000 node problems as the simplicity of the criterion increases. This is not always the case. For example, on the 500 node, 2,250 arc problem, the number of pivots is larger for the most negative and modified first negative criteria than for the first negative criterion. In general, the superiority of the first negative criterion strictly improves as the number of arcs increase. This seems somewhat peculiar since a change in the number of arcs does not affect the number of basic variables. A partial explanation of this result is provided by studying the "NEWARC" column. More precisely, observe that the "NEWARC" times are much larger for the most negative and modified first negative criteria than for the first negative criterion. This indicates that the basic variable picked by the former criteria have significantly more negative coefficient values in the equation of nonbasic variables associated with it than does the basic variable picked by the

latter criterion. This conclusion is occasioned by the fact that the updated costs associated with non-negative coefficient values do not have to be calculated. (Recall these problems are uncapacitated.)

Table I indicates that the most negative and modified first negative criteria have little to offer by comparison with the first negative criterion since they require more search time to find the basic variable to leave the basis and more computation to determine the nonbasic variable to enter the basis. Moreover, they do not substantially reduce the number of pivots. The version of the dual code using the first negative criterion will be called DNET.

4.3 Other Conclusions

It is interesting to compare the data in Table I for DNET and PNET. This comparison indicates that PNET consistently requires approximately twice as many pivots as DNET. However, PNET is consistently 6-12 times faster. The basis of the inferiority of DNET (and the dual method) to PNET is seen when the time per pivot of the two codes is compared. Table I indicates that as problem size and/or number of arcs increase the time per pivot in DNET grows disproportionately large. For instance, the median time/pivot for DNET on the 500 node networks is .045 compared to .003 for PNET. The primary cause is the large amount of time required to find the variable to enter the basis. Since the modifications of the dual algorithm in [14] were designed to minimize these calculations, we conclude that the dual method is not a competitive "dead start" procedure for solving large problems--i.e., its potential value for large problems lies in those situations in which an extremely good dual starting solution is available. It should be noted, however, that if the code were changed to use the augmented threaded index method [16] the dual times would be improved by a greater factor than its use in the primal code. We did not test this because the ATI method [16] was developed subsequent to our original testing; however, we strongly feel that its use would not change our conclusions.

5.0 COMPUTATIONAL COMPARISON AND CODE REQUIREMENTS

5.1 Computational Comparison of Several Codes

Once the best versions of the primal and dual codes had been ascertained, we sought to compare their computational efficiency to other codes.

The codes which we obtained for comparison include Bennington [2], Boeing, General Motors, SHARE [4,25], SUPERK [1], and the

Texas Water Development Board. All of these codes are variants of the out-of-kilter method [10] except for Bennington [2]. To our knowledge there are no primal codes in existence except for PNET and PNET-I. Thus none were available for testing. However, a primal code is currently being developed by Graves and McBride at UCLA.

The Boeing code was developed at Boeing, Inc., and the General Motors (GM) code at General Motors, Inc. The Texas Water Development Board (TWB) code was developed by Paul Jensen at the University of Texas. This code was later modified by the Texas Water Development Board to enhance its computational efficiency on small problems.

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 (including the primal and dual codes) have been tuned (optimized) for a particular compiler. All of the problems were solved on the CDC 6600 at the University of Texas Computation Center using the RUN compiler. 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 total 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.

To compare the codes fairly we sought a set of problems which would include different types of problems (e.g., assignment, transportation, and minimum cost flow network problems), both capacitated and uncapacitated, and with varying node and arc requirements. Also we desired the problems to be generally available to other researchers and the problem set to be reasonably small. The 40 benchmarked problems in [20] suited these requirements perfectly. The problem specifications of these 40 problems as required on the input cards to the network generator in [20] are given in Table II. Problems 1-5 are 100 × 100 transportation problems; problems 6-10 are 150 × 150 transportation problems. Problems 11-15 are 200 × 200 assignment problems. Problems 16-27 are 400 node capacitated transhipment problems; problems 28-35 are uncapacitated 1,000 and 1,500 node transhipment problems. Problems 36-40 are very large transhipment problems varying from 3,000 nodes and 35,000 arcs to 8,000 nodes and 15,000 arcs. Table III contains the solution times for each of these problems for each of the codes.

A noteworthy feature of the computational results is that PNET, PNET-I, and SUPERK are decidedly superior to the other codes. Roughly, PNET, PNET-I, and SUPERK are at least 4 times and in many cases 8-10 times faster than the other codes. Furthermore, PNET-I strictly dominates PNET and SUPERK. PNET-I is

approximately 10% faster than PNET, thus the ATI method [16] appears to be superior to the API method [11]. Additionally, PNET-I is roughly twice as fast as SUPERK which is the fastest known out-of-kilter code. This result is extremely important since it completely contradicts the folklore that the out-of-kilter method is the fastest (in terms of total solution time) means for solving transhipment problems. Another important result which can be gleaned from Table III is that the dual method is not competitive with either the primal codes or the out-of-kilter codes.

Table II

PROBLEM SPECIFICATIONS*

	Number of Nodes	Number of Sources	Number of Sinks	Number of Arcs	Total Supply	Transhipments Sources	Transhipments Sinks	Percent of High Cost	Percent of Capacitated	Upper Bound Range Min	Upper Bound Range Max	Random No. Seed
1.	200	100	100	1300	100,000	0	0	0	0	0	0	13502460
2.	200	100	100	1500	100,000	0	0	0	0	0	0	13502460
3.	200	100	100	2000	100,000	0	0	0	0	0	0	13502460
4.	200	100	100	2200	100,000	0	0	0	0	0	0	13502460
5.	200	100	100	2900	100,000	0	0	0	0	0	0	13502460
6.	300	150	150	3150	150,000	0	0	0	0	0	0	13502460
7.	300	150	150	4500	150,000	0	0	0	0	0	0	13502460
8.	300	150	150	5155	150,000	0	0	0	0	0	0	13502460
9.	300	150	150	6075	150,000	0	0	0	0	0	0	13502460
10.	300	150	150	6300	150,000	0	0	0	0	0	0	13502460
11.	400	200	200	1500	200	0	0	0	0	0	0	13502460
12.	400	200	200	2250	200	0	0	0	0	0	0	13502460
13.	400	200	200	3000	200	0	0	0	0	0	0	13502460
14.	400	200	200	3750	200	0	0	0	0	0	0	13502460
15.	400	200	200	4500	200	0	0	0	0	0	0	13502460
16.	400	8	60	1306	400,000	0	0	30.	20.	16,000	30,000	13502460
17.	400	8	60	2443	400,000	0	0	30.	20.	16,000	30,000	13502460
18.	400	8	60	1306	400,000	0	0	30.	20.	20,000	120,000	13502460
19.	400	8	60	2443	400,000	0	0	30.	20.	20,000	120,000	13502460
20.	400	8	60	1416	400,000	5	50	30.	40.	16,000	30,000	13502460
21.	400	8	60	2836	400,000	5	50	30.	40.	16,000	30,000	13502460
22.	400	8	60	1416	400,000	5	50	30.	40.	20,000	120,000	13502460
23.	400	8	60	2836	400,000	5	50	30.	40.	20,000	120,000	13502460
24.	400	4	12	1382	400,000	0	0	30.	80.	16,000	30,000	13502460
25.	400	4	12	2676	400,000	0	0	30.	80.	16,000	30,000	13502460
26.	400	4	12	1382	400,000	0	0	30.	80.	20,000	120,000	13502460
27.	400	4	12	2676	400,000	0	0	30.	80.	20,000	120,000	13502460
28.	1000	50	50	2900	1,000,000	0	0	0	0	0	0	13502460
29.	1000	50	50	3400	1,000,000	0	0	0	0	0	0	13502460
30.	1000	50	50	4400	1,000,000	0	0	0	0	0	0	13502460
31.	1000	50	50	4800	1,000,000	0	0	0	0	0	0	13502460
32.	1500	75	75	4342	1,500,000	0	0	0	0	0	0	13502460
33.	1500	75	75	4385	1,500,000	0	0	0	0	0	0	13502460
34.	1500	75	75	5107	1,500,000	0	0	0	0	0	0	13502460
35.	1500	75	75	5730	1,500,000	0	0	0	0	0	0	13502460
36.	8000	200	1000	15000	4,000,000	100	300	0	0	30	0	13502460
37.	5000	150	800	23000	4,000,000	50	100	0	0	0	0	13502460
38.	3000	125	500	35000	2,000,000	25	50	0	0	0	0	13502460
39.	5000	180	700	15000	4,000,000	100	300	.1	.7	3000	5000	13502460
40.	3000	100	300	23000	2,000,000	50	100	.1	.7	2000	4000	13502460

*Cost Range: Min = 1, Max. = 100

TABLE III

Solution Times (in seconds)

Problems	PNET	PNET-I	DNET	BENN	SUPERK	GM	SHARE	Boeing	TWB
1	1.30	1.17	12.85	20.25	5.68	46.25	17.76	30.25	21.23
2	1.49	1.35	13.56	24.36	6.47	63.30	21.34	21.59	21.39
3	1.94	1.74	21.44	34.56	6.87	105.72	26.16	31.47	28.63
4	1.64	1.47	17.96	31.45	6.57	70.74	25.13	36.47	27.60
5	1.88	1.73	23.34	52.10	6.77	90.10	30.97	47.73	NA
6	3.55	3.06	46.10	61.00	11.05	92.32	46.40	46.64	NA
7	4.06	3.57	74.88	DNR	12.86	157.31	65.92	113.12	NA
8	4.72	4.20	97.92	DNR	13.69	160.71	81.00	175.10	NA
9	4.80	4.35	101.65	DNR	13.40	158.01	81.21	186.99	NA
10	5.88	5.57	95.96	DNR	14.13	197.82	84.24	184.75	NA
11	3.52	3.12	19.87	17.44	6.44	35.67	19.93	30.39	NA
12	4.87	4.48	26.58	20.31	6.47	28.43	21.17	22.08	NS
13	5.52	4.91	27.98	24.92	7.25	31.39	25.81	20.02	NA
14	6.02	5.56	30.15	27.40	6.95	18.62	24.95	23.11	NA
15	6.50	5.91	31.57	DNR	7.56	23.48	27.05	21.08	NA
16	2.40	2.15	14.77	11.77	5.27	60.27	21.51	15.05	17.55
17	3.11	2.90	DNR	20.10	8.36	96.66	32.40	64.64	NA
18	1.92	1.70	DNR	11.31	5.13	61.54	20.06	18.31	19.15
19	2.60	2.40	DNR	20.62	8.49	DNR	31.75	61.07	NA
20	2.67	2.47	DNR	10.38	4.69	DNR	18.11	25.72	21.43
21	2.76	2.46	DNR	20.35	7.96	DNR	32.60	61.39	NA
22	2.22	2.01	DNR	9.97	4.60	DNR	17.91	24.84	NA
23	3.00	2.74	DNR	19.81	7.91	DNR	32.66	67.96	NA
24	3.12	2.91	DNR	11.71	5.59	DNR	25.27	21.57	NA
25	4.17	3.96	DNR	18.27	8.37	DNR	33.19	48.40	NA
26	4.45	4.15	DNR	11.38	5.51	DNR	25.05	19.34	NA
27	4.42	4.31	DNR	16.37	7.50	DNR	30.45	41.98	NA
28	6.35	5.67	DNR	DNR	13.91	DNR	53.87	83.98	NA
29	7.39	6.55	DNR	DNR	14.51	DNR	52.55	117.83	NA
30	9.08	8.10	DNR	DNR	16.00	DNR	61.33	152.21	NA
31	9.59	8.48	DNR	DNR	17.05	DNR	61.33	135.73	NA
32	15.70	13.59	DNR	DNR	22.88	DNR	78.63	553.93	NA
33	20.20	17.65	DNR	DNR	25.89	DNR	101.92	210.14	NA
34	17.10	14.86	DNR	DNR	25.42	DNR	92.25	248.16	NA
35	19.39	17.13	DNR	DNR	29.96	DNR	DNR	DNR	NA
36	384.08	DNR	DNR	NA	NA	NA	NA	NA	NA
37	245.40	DNR	DNR	NA	NA	NA	NA	NA	NA
38	140.98	DNR	DNR	NA	NA	NA	NA	NA	NA
39	193.42	DNR	DNR	NA	NA	NA	NA	NA	NA
40	105.09	DNR	DNR	NA	NA	NA	NA	NA	NA

NA - Code and data would not fit in 104,000 words of memory.

DNR- Did not run.

Looking at the out-of-kilter and dual codes solution times, it is interesting to note that these solution times are much more dependent on the number of arcs (holding all other parameters of the problem constant) than the primal codes.

5.2 Memory Requirements of the Codes

Table IV indicates the number of node and arc length arrays required in each of the codes tested for solving capacitated problems. It should be noted that memory requirements of all of

eing	TWB
0.25	21.23
1.59	21.39
1.47	28.63
6.47	27.60
7.73	NA
6.64	NA
3.12	NA
5.10	NA
6.99	NA
4.75	NA
0.39	NA
2.08	NS
0.02	NA
3.11	NA
1.08	NA
5.05	17.55
4.64	NA
8.31	19.15
1.07	NA
5.72	21.43
1.39	NA
4.84	NA
7.96	NA
1.57	NA
8.40	NA
9.34	NA
1.98	NA
3.98	NA
7.83	NA
2.21	NA
5.73	NA
3.93	NA
0.14	NA
8.16	NA
[illegible]	NA
[illegible]	NA
[illegible]	NA
[illegible]	NA
[illegible]	NA
[illegible]	NA

ıtion times,
are much more
arameters of

ength arrays
acitated
ts of all of

the codes tested were quite close (within 1,000 words) excluding the array requirements. Thus the important factor in comparing the codes is the number of node and arc length arrays. Also, it should be noted that the primal and dual codes require one less arc length array if the problem is uncapacitated, whereas, all of the problems solved by an out-of-kilter code are capacitated due to the circulation arcs.

TABLE IV

CODE SPECIFICATIONS

	Developer	Name	Type	Number of Arrays
1.	Barr, Glover, Klingman	SUPERK	Out-of-kilter	4N + 9A
2.	Bennington	BENN	Non-simplex	6N + 11A
3.	Boeing	Boeing	Out-of-kilter	6N + 8A
4.	Clasen	SHARE	Out-of-kilter	6N + 7A
5.	Glover, Karney Klingman	PNET	Primal network	6N + 3A
6.	Glover, Karney Klingman	DNET	Dual network	9N + 3A
7.	Glover, Karney, Klingman, Stutz	PNET-I	Primal network	5N + 3A
8.	General Motors	GM	Out-of-kilter	3N + 6A
9.	Texas Water Development Board	TWB	Out-of-kilter	$4N + N^2 + 7A$

N - Node Length

A- Arc Length

Looking at Table IV and keeping in mind that any meaningful network problem has to have more arcs than nodes, it is clear that the primal and dual codes have a distinct advantage (in terms of memory requirements) over all of the other codes. Further, this advantage greatly increases as the number of arcs increase and if the problem is uncapacitated. For example, consider a problem which has 10 times as many arcs as nodes.

PNET, PNET-I, or DNET require only about one-half the memory that the best (in terms of memory requirements) of the other codes require, enabling simplex based codes to solve much larger problems than other codes.

REFERENCES

1. Barr, R. S., F. Glover and D. Klingman, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes," to appear in *Mathematical Programming*.

2. Bennington, G. E., "An Efficient Minimal Cost Flow Algorithm," *O. R. Report 75*, North Carolina State University, Raleigh, North Carolina, June 1972.

3. Charnes, A. and W. W. Cooper, *Management Models and Industrial Applications of Linear Programming*, 2 Vols., John Wiley & Sons, Inc., New York, 1961.

4. 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.

5. Dantzig, F., *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.

6. Dennis, J. B., "A High-Speed Computer Technique for the Transportation Problem," *Journal of Association for Computing Machinery*, 8, 1958, pp. 132-153.

7. Flood, M. M., "A Transportation Algorithm and Code," *Naval Research Logistics Quaterly*, 8, 1971, pp. 257-276.

8. Ford, L. R. and D. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, New Jersey, 1962.

9. Fong, C. O. and M. R. Rao, "Accelerated Labeling Algorithms for the Maximal Flow Problem with Applications to Transportation and Assignment Problems," *W. P. No. 7222*, Graduate School of Management, University of Rochester, December 1972.

10. Fulkerson, D. R., "An Out-of-Kilter Method for Solving Minimal Cost Flow Problems," *J. Soc. Indust. Appl. Math.*, 9, 1961, pp. 18-27.

11. Glover, F., D. Karney and D. Klingman, "The Augmented Predecessor Index Method for Locating Stepping Stone Paths and Assigning Dual Prices in Distribution Problems," *Transportation Science*, 6, 1, 1972, pp. 171-180.

12. Glover, F., D. Karney, D. Klingman and A. Napier, "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," *Management Science*, 20, 5, 1974, pp. 793-813.

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

14. Glover, F., D. Klingman and A. Napier, "An Efficient Dual Approach to Network Problems," *OPSEARCH*, 9, 1, 1972, pp. 1-18.

15. Glover, F., D. Klingman and A. Napier, "Basic Dual Feasible Solution for a Class of Capacitated Generalized Networks," *Operations Research*, 20, 1, 1972, pp. 126-137.

16. Glover, F., D. Klingman and J. Stutz, "The Augmented Threaded Index Method," *Research Report CS 144*, Center for Cybernetic Studies, University of Texas, Austin, 1973.

17. Glover, F., D. Karney and D. Klingman, "A Note on Computational Studies for Solving Transportation Problems," *Proc. of the ACM 1973 Conference*, Atlanta.

18. Johnson, E., "Networks and Basic Solutions," *Operations Research*, 14, 4, 1966, pp. 619-623.

19. Klingman, D., A. Napier and G. Ross, "A Computational Study on the Effects of Problem Dimensions on Solution Time for Transportation Problems," *Research Report CS 135*, Center for Cybernetic Studies, University of Texas, Austin, 1973.

20. Klingman, D., A. Napier and J. Stutz, "NETGEN - A Program for Generating Large Scale (Un) Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," *Management Science*, 20, 5, 1974, pp. 814-821.

21. Langley, R. W., "Continuous and Integer Generalized Flow Problems," Ph.D. dissertation, School of Industrial and Systems Engineering, Georgia Institute of Technology, June 1973.

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

23. "Network Flow Routine," YIM/FOCUS Library Number H3 CODA NETFLOW; Control Data Corporation, Software Mfg. and Distribution, 215 Moffett Park Drive, Sunnyvale, California.

24. Orden, A., "The Transhipment Problem," *Management Science*, 2, 3, 1956, pp. 276-285.

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

26. Reinfield, N. V. and W. R. Vogel, *Mathematical Programming*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1958.

27. Srinivasan, V. and G. L. Thompson, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm," *JACM*, 20, 1973, pp. 194-213.

28. "UKILT-1100 Programmer Reference Manual," UNIVAC, Data Processing Division, Roseville, Minnesota.

29. Wagner, H., *Principles of Operations Research*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.

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.

Paper received October 8, 1973.