MANAGEMENT SCIENCE
Vol. 24, No. 12, August 1978
Printed in U.S.A.

GENERALIZED NETWORKS: A FUNDAMENTAL COMPUTER-BASED PLANNING TOOL*

F. GLOVER,† J. HULTZ,‡ D. KLINGMAN§ AND J. STUTZ§

This paper documents the recent emergence of generalized networks as a fundamental computer-based planning tool and demonstrates the power of the associated modeling and solution techniques when used together to solve real-world problems.

The first sections of the paper give a non-technical account of how generalized networks are used to model a diversity of significant practical problems. To begin, we discuss the model structure of a generalized network (GN) and provide a brief survey of applications which have been modeled as GN problems. Next we explain a somewhat newer modeling technique in which generalized networks form a major, but not the only, component of the model.

The later sections give a technical exposition of the design and analysis of computer solution techniques for large-scale GN problems. They contain a study of GN solution strategies within the framework of specializations of the primal simplex method. We identify an efficient solution procedure derived from an integrated system of start, pivot, and degeneracy rules. The resulting computer code is shown, on large problems, to be at least 50 times more efficient than the LP system, APEX III.

(NETWORKS; FLOWS; PROGRAMMING COMPUTERS)

1.0 Introduction

A generalized network (GN) problem is simply a type of LP problem and can thus be solved using any standard LP solution technique. However, none of the current LP systems is capable of fully exploiting the structure of generalized network problems. With the recent development of GN computer codes, Bradley's 1975 prediction that GN problems "in the near future . . . could come to be regarded as a fundamental model" [10] is coming true. Modelers have begun to devote attention to determining if an LP model is a GN problem and, more importantly, to devising formulations in which generalized networks play the role of critical components.

There are two powerful incentives for adopting a GN formulation whenever possible. The major advantage is the ability to solve GN problems—and by extension a variety of problems with GN components—with a remarkable degree of efficiency. The second motivation for using GN models is that they can be conceptualized graphically as well as algebraically. The pictorial presentation of a generalized network is a useful device for communicating mathematical models to nonscientific users and for teaching others how to formulate problems.

The purpose of this paper is to document the recent emergence of generalized networks as a fundamental computer-based planning tool and to demonstrate the power of the associated modeling and solution technologies when used in concert to solve real-world applications. The paper contains a nontechnical account of how generalized networks are used to model a diversity of significant practical problems. Using a graphical representation, we first define the model structure of a generalized network. Next we provide a brief survey of applications which have been modeled as GN problems. We then explain somewhat newer modeling techniques in which generalized networks form a major, but not the only, component of the model. This modeling approach yields a formulation that enables one to solve the problem as a sequence of GN problems resulting in dramatic gains in efficiency over alternative approaches. To provide an understanding of this approach and the role of generalized networks within it, we describe a real-world problem which has been solved by its use.

The paper also gives a technical exposition of the design and analysis of computer solution techniques for large-scale GN problems. It contains an indepth computational study of GN solution strategies within the framework of specializations of the primal simplex method. Here we identify an efficient solution procedure derived from an integrated system of start, pivot, and degeneracy rules. The resulting method is shown, on large problems, to be at least 50 times more efficient than the sophisticated state-of-the-art LP system, APEX-III. In others words, the method can solve a problem every week for a year and consume the same amount of computer time required to solve the problem only once with the LP system. The memory requirements of the method, as well as the solution times, are sufficiently small to warrant its use as a computer-based planning tool not only in a batch processing environment, but also in an interactive setting.

* Accepted by Michael Held; received April 14, 1977. This paper has been with the authors 4 months for 2 revisions.

† University of Colorado.
‡ Analysis, Research, and Computation, Inc., Austin, Texas.
§ University of Texas.

2.0 Problem Definition

The generalized network problem represents a large class of LP problems. This class includes any LP problem whose coefficient matrix, ignoring simple upper bound constraints, contains at most two nonzero entries in each column. A large portion of the literature on LP problems has been devoted to the special cases of the GN problem in which the nonzero elements of a column consist of a 1 and a -1 (either initially or by linear transformation). This condition identifies the problem as a pure network, whose instances include shortest path, maximum flow, assignment, transportation, and transshipment problems. The GN problem, by allowing other nonzero doubletons (and singletons) in a column, is actually the broadest classification of linear network related problems. Practical settings in which GN problems arise include problems of resource allocation, production, distribution, scheduling, capital budgeting, and so on.

A generalized network, like a pure network, is best represented as a directed graph. Under the assumed existence of a finite optimum, it is possible to transform the coefficient matrix (by scaling or by complementing a variable relative to its upper bound), so that if a column has two nonzero entries, at least one of these is -1. In this way, a directed arc is "formed" that leads from the node associated with the -1 to the node associated with the other nonzero entry. If both entries are -1, the arc may be directed either way. Columns with single nonzero entries give rise to arcs incident on only one node.

There is an important distinction between arcs in pure network problems and arcs in GN problems. In generalized networks, each arc's multiplier is the nonzero coefficient associated with the node at the head of the arc (i.e., the node to which the arc is directed). In pure networks, the multiplier is always $+1$.

Consider the following GN problem:

$$\text{Mimimize } 1x_{12} + 5x_{13} + 3x_{23} + 1x_{24} - 4x_{32} - 9x_{34}$$

Subject to:

$$\begin{array}{rrrrrrr}
-1x_{12} & -1x_{13} & & & & & = -5 \\
2x_{12} & & -1x_{23} & -1x_{24} & +1/3x_{32} & & = 0 \\
 & 1/2x_{13} & +1x_{23} & & -1x_{32} & -1x_{34} & = 0 \\
 & & & -1/5x_{24} & & +3x_{34} & = 10
\end{array}$$

$$0 \leqslant x_{12} \leqslant 3, \quad 0 \leqslant x_{13} \leqslant 4, \quad 0 \leqslant x_{23} \leqslant 6,$$

$$0 \leqslant x_{24} \leqslant 5, \quad 0 \leqslant x_{32} \leqslant 3, \quad 0 \leqslant x_{34} \leqslant 7.$$

The network associated with this problem is shown in Figure 1. As with pure network problems, each row of the coefficient matrix is associated with a node and each column with an arc. In other words, a node corresponds to a problem equation and an

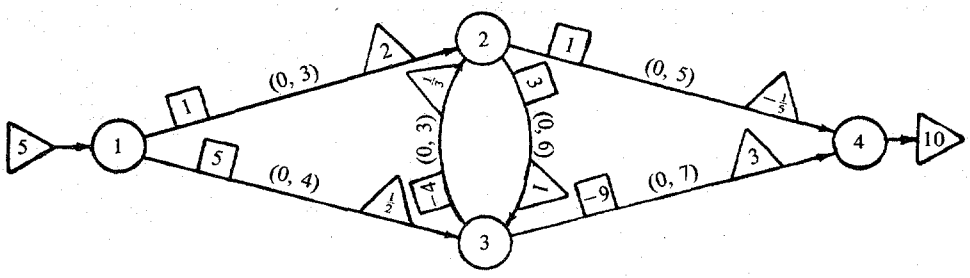

FIGURE 1. Generalized Network.

arc corresponds to a problem variable. The arc is directed from the node associated with the -1 entry toward the node associated with the other non-zero entry. Likewise, each arc has a cost, a lower bound, and an upper bound. In Figure 1 the cost is shown within the square and the lower bound and upper bound respectively are shown in parentheses. The nonzero multiplier associated with each arc is shown in Figure 1 within a triangle on the arc. The constant terms (right hand sides) of the problem equations identify supply and demand requirements attached to the corresponding nodes. A negative constant term identifies a supply (which by convention equals the absolute value of this term), a positive constant term identifies a demand, and a 0 constant term identifies a "conservation condition" in which the amount of flow entering the node must be exactly matched by the amount of flow leaving the node.

The flow passing across an arc in a generalized network problem is acted upon by the nonzero multiplier. It indicates that the flow entering the arc is multiplied by the value of the multiplier as the flow leaves the arc. Thus, the amount starting out on an arc will not necessarily be the amount arriving at the opposite end. For example, if 2 units start on the arc from node 1 to node 2 in Figure 1, 4 units will arrive at node 2 since the multiplier is 2. Likewise, 10 units starting on the arc from node 2 to node 4 will result in -2 units arriving at node 4 since the multiplier in this case is $-1/5$. It should be noted that the cost, lower bound, and upper bound of each arc apply only to the units of flow entering that arc.

Another important feature of GN problems is that total supply may not be the same as total demand. In pure network problems, total supply always equals total demand. However, the effect of multipliers is such that total supply and total demand may, in fact, be entirely different. This can result in odd structural consequences, such as absorbing and generating cycles. (See [3], [29], [30].)

3.0 Applications of Generalized Networks

Generalized networks can be used to model numerous problems for which there are no pure network equivalents. There are essentially two ways in which the multipliers on the arcs of generalized networks can function. They can act simply to modify the amount of flow of a particular good or they can transform the flow from one type of good to another. In the former case generalized networks can be used to represent situations involving evaporation, seepage, deterioration, breeding, interest rates, sewage treatment, purification processes of varying efficiencies, machine efficiencies and structural strength design. In the latter capacity, generalized networks can model processes of manufacturing, production, conversions of fuel to energy, blending, crew scheduling, allocating manpower to job requirements, and currency exchanges. The following applications lend insight into the possible uses of generalized networks.

A complete water distribution system with losses has been modeled by Bhaumik [7] as a generalized network problem. This model was primarily concerned with the movement of water through canals to various reservoirs. However, the model also had to consider the retention of water over several time periods. The multipliers in this case represented the loss due to both evaporation and seepage.

Turner and Gilliam [16] have proposed a file reduction model which has the form of a generalized transportation model (a special type of GN) with a single extra constraint. This model was designed to facilitate the reduction of extremely large microdata files to smaller, statistically representative files. The objective, in this case, was to minimize the amount of information lost in the reduction process. The arcs represented paths from the original records to the reduced records. A nonzero flow on an arc implied that the originating record was to be represented by the terminal record. The multipliers on the arcs were used to insure that the reduced file was truly representative of all of the original records.

Kim [35] has utilized generalized networks to represent copper refining processes. The electrolytic refining procedure, in this case, was modeled by a large d-c electrical network. The arcs were current paths with the multipliers representing the appropriate resistances. In this way, Kim analyzed the effect of short circuits in the refining process.

Charnes and Cooper [11] have identified applications of generalized networks for both plastic-limit analysis and warehouse funds-flow models. In plastic-limit analysis, the network was generated by forming the equations for horizontal and vertical equilibrium and by employing a coupling technique. The warehouse funds-flow model was actually a multi-time period model. The arcs were used to represent sales, production, and the inventory holding of both products and cash. The multipliers were introduced to facilitate the conversions between cash and products

A cash management problem has been modeled as a generalized network by Crum [12]. This model for a multi-national firm incorporated transfer pricing, receivables and payables, collections. dividend payments, interest payments, royalties, and management fees. The arcs represented possible cash flow patterns and the multipliers represented costs, savings, liquidity changes, and exchange rates.

Other applications of generalized networks include machine loading problems [11], [13], [43], blending problems [11], [43], the caterer problem [13], [43], and scheduling problems dealing with production and distribution, crew scheduling, aircraft scheduling, and manpower training [11], [13], [43].

4.0 Integer Generalized Networks

The uses of arc multipliers are not limited to the examples just discussed. In fact, upon adding the requirement of discreteness, which forces the flows on particular arcs to occur in integer quantities, the GN problem is capable of modeling an unexpected diversity of problems [11, Chapter 17]. For example, introducing discreteness into the GN model produces a framework for problems such as scheduling variable length television commercials into time slots, assigning jobs to computers in computer networks, scheduling payments on accounts where contractual agreements specify "lump sum" payments, and designing communication networks with capacity constraints. While these are "direct" applications, the use of special modeling principles enables even more complex applications to be modeled and solved as integer GN problems. In fact, this approach makes it possible to model any 0-1 LP problem as an integer GN problem [23], [27]. These procedures extend quite naturally to accomodate mixed integer 0-1 LP problems where the continuous part of the problem is a transportation, transshipment or generalized network problem itself. Reference [42] shows how contemporary financial capital allocation problems can be modeled as integer GN problems. Many other important real-world applications have a similar "mixed" structure, including a variety of energy models, plant location models, and physical distribution models. The remainder of this section briefly describes the basic principles of this approach and discusses a practical application which has profited by its use.

Figure 2 illustrates a useful modeling devise that finds application in a variety of settings. The costs, bounds, and multipliers are represented in the same fashion as earlier. In addition, the asterisk on the arc from node 0 to node A indicates that its flow must be an integer amount. Consequently, in view of the upper and lower bounds on this arc, the only acceptable flow values are exactly 0 and 1, and the multiplier thus ensures that either 0 or 3 units of flow are transmitted to node A. Further, the only possible way to distribute 3 units of flow into node A is to send exactly one unit to each of the nodes 1, 2, and 3 since each of the three arcs leaving A has an upper bound of 1. Thus, in sum, the following effect has been achieved: when

the flow on the arc from node 0 to node A is 0, the flow on each of the three arcs out of node A is 0; when the flow on the arc from node 0 to A is 1, the flow on each of the three arcs out of node A is 1.

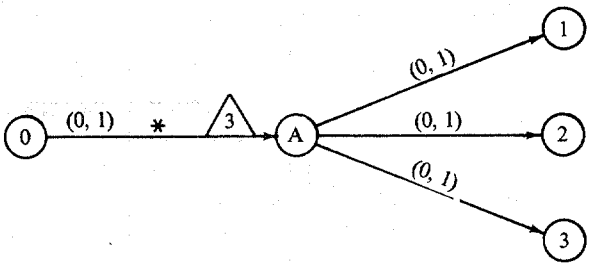

FIGURE 2. Generalized Network with Integer Flow Restrictions.

It should be noted that multipliers may also be attached to the arcs leaving node A, so that their flows may be further transformed. For example, the flow on the arc from node 0 to node A can represent an investment decision (invest if flow = 1, do not invest if flow = 0), and the flows on the arcs out of A can represent components of the investment (e.g., particular stocks in a portfolio, tracts of land in a real estate venture, items of equipment in a manufacturing operation, etc.). Multipliers on the latter arcs would then represent the number of items of each of these investment components that are obtained by selecting the main investment. (For example, a particular equipment investment may be composed of six machines of type 1, eight machines of type 2, and so forth.) The combination of arc multipliers and the 0-1 integer restriction gives rise to what is called an integer generalized network or a 0-1 generalized network. This modeling tool has a variety of important uses, as demonstrated more concretely by the following real-world application.

4.1 *Air Force Course Scheduling*

The Air Force requires Undergraduate Flight Training (UFT) graduates to take advanced flight training before their first operational assignment. In addition, UFT graduates must take from one to four survival training courses. Since the men come from different backgrounds, a different course schedule is required for each. Furthermore, both the flight and the survival training courses are offered only at certain times and at various locations around the country. They are subject to enrollment limits and have prerequisites. A set of feasible course schedules must be identified for each UFT graduate and given a "cost rating." Feasibility and cost considerations depend on factors such as attendance requirements at Combat Crew Training courses, various modes of transporting the students to the course locations, the number of dead days in the pipeline, the opportunity for the UFT graduates to take leave as desired, etc.

The objective is to select a particular course schedule for each UFT graduate so that the complete set of schedules selected will satisfy all class enrollment limits and result in the smallest total cost. To solve this problem, the personnel manager in the Training Pipeline Management Division previously assigned each graduate to a feasible schedule by hand, trying to assure that all enrollment limits were satisfied. Clearly, this was a difficult and time-consuming task to do by hand; further, the total cost of training these men was probably far from optimal when the assignments were made manually.

In search of a better approach, the Air Force developed an integer programming formulation for this problem. However, the IP formulation turned out to be almost totally resistant to solution. Consequently, we reformulated this integer programming problem as a 0-1 GN problem which is shown in Figure 3.

The elements of this diagram may be explained as follows. The node (Mi) represents the ith man and has a supply of exactly 1. Each man node is connected by arcs to its set of man/schedule nodes. These connecting man/schedule arcs have a multiplier a_{ij} equal to the number of classes in the schedule and a cost c_{ij} equal to the cost of assigning man i to his jth schedule. The asterisk again indicates that flow must be integer-valued.

The arcs emanating from a man/schedule node in Figure 3 lead to the individual classes making up the schedule. Each of these arcs has an upper bound of one. Thus, if a particular schedule is "selected," then every class in the schedule is also automatically selected. The objective is to pick a schedule for each man that will minimize the value of the assignments on the overall program, subject to the upper and lower attendance limits for each class, expressed as bounds on the arcs from class nodes to the sink node of Figure 3. All arc costs, except for those attached to the man/schedule arcs, are thus equal to 0.

The UFT problem typically involves 120 men, 200 classes, and 460 schedules, resulting in a 0-1 LP formulation with 520 constraints and 460 0-1 variables. The 0-1 GN formulation involves the same number of 0-1 variables, and introduces an additional 2,200 continuous variables (arcs) and 780 nodes. Viewed from an LP problem context, this might seem to represent a fair increase in size. However, it

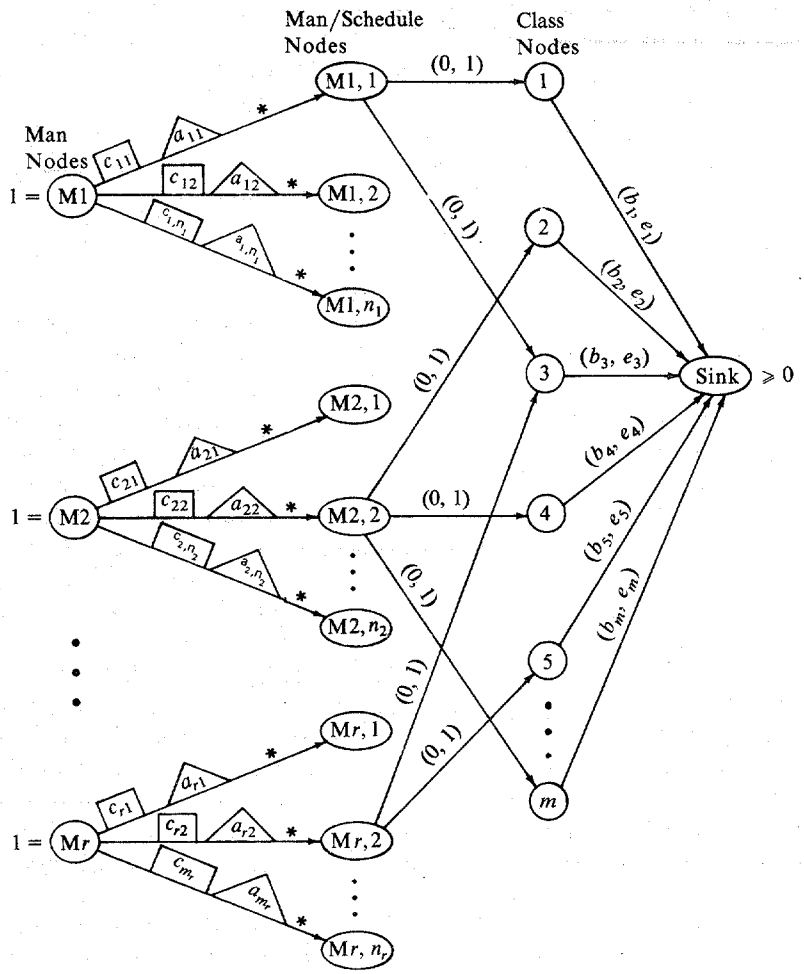

FIGURE 3. UFT Formulation.

actually represents a relatively small GN problem. This 0-1 GN problem was solved using a specialized branch and bound procedure with GN subproblems. The optimal solution was often found and verified after only 30 seconds and in some cases required a total solution time of only 10 seconds on a CDC 6600. The problem was thus transformed from one that had been extremely difficult to solve as an integer program to one that was solved easily as a integer GN problem.

5.0 Motivations for Using GN models

The two important advantages to adopting a GN formulation where appropriate have been outlined. Unlike LP problems, a GN can be represented in graph form. The ability to represent a generalized network graphically as well as algebraically facilitates the modeling procedure and is a useful device for communicating mathematical problems to nonscientific users. The major incentive for using GN models is the ability to solve these problems—and a variety of additional problems with GN components—with a remarkable degree of efficiency.

The following sections of the paper present an abridged computational analysis of algorithmic rules and computer implementation procedures for GN problems. The unabridged version [17] may be obtained by writing the authors. Computational studies of pure network solution procedures have done much to advance the state-of-the-art. Excellent testing has been performed on computer codes for transportation problems [18], [20], [28], [36], [39], [43] and for transshipment problems [1], [4], [5], [10], [19], [26], [33], [37], [41]. These studies have provided critical insights into the best methods for solving such problems as well as providing benchmark data for future solution procedures.

To date there have been no in-depth studies concerning the much broader class of GN problems, although computer codes do exist for solving such problems. Code development has been reported by Eisemann [14], Maurras [40], Glover, Klingman, and Stutz [25], Bhaumik and Jensen [8], Langley [38], and Balachandran [2], among others. Most of these papers report findings for only certain classes of GN problems and all of them are limited in the scope of the computational analysis. Thus, an important body of empirical research has heretofore been lacking in the network literature.

The code NETG reported by Glover, Klingman, and Stutz [25] was selected to form the basis for the computational testing of this study. This code is an implementation of the extended augmented predecessor index (EAPI) procedure [18, 24], and embodies many of the latest advances in solution methodology for generalized network problems.

In any computer implementation, there are numerous steps that can be performed in alternative ways. Experience from previous studies of pure network problems has shown that the determination of an effective set of decision rules to handle such alternatives can have an enormous impact on the efficiency of the implemented solution method. Consequently, one of our primary objectives in this study was to investigate decision rules for the GN problem and establish their relative merits. We determined the best rules and integrated them to produce a code which has been tested against the highly efficient linear programming system, APEX-III. This testing indicates that the streamlined version of NETG solves large-scale GN problems 50 times faster than APEX-III.

6.0 NETG Data Structures

Since a generalized network problem is simply a type of LP problem, it can be solved using any standard LP solution technique. Improvements in inversion and

reinversion processes, data compactification, and pivot selection strategies have provided dramatic increases in the efficiency of primal simplex computer codes in recent years. The structure of a generalized network problem could be detected by a primal simplex LP code; this information could then be used to reduce storage requirements and to simplify operations. Further, the inherent generalized upper bounding (GUB) constraints in GN problems could be exploited by those LP codes which have a GUB feature. However, none of the current LP systems is capable of fully exploiting the structure of generalized network problems.

One of the conspicuously exploitable features of generalized network problems is the sparsity of the coefficient matrix (at most two non-zero entries per column), and current LP codes are of course designed to take advantage of sparsity to store data economically. When the problem is transformed to graph form, storage may be reduced even further. By the use of simple ordered lists to capture the graph structure, NETG is designed to store only the head node identifier, the cost coefficient, the nonzero multiplier, and the upper bound for each column of the coefficient matrix. In this way, problem data can often be resident in "fast access" memory for extremely large problems.

Bases for generalized network problems have a special structure. With possible reordering of the rows and columns, the basis matrix forms a block diagonal matrix. Each of the blocks is either triangular or near-triangular and can be represented as a quasi-tree (a tree with an additional arc). Johnson [31], [32] originally proposed a linked list procedure for storing simple trees and suggested its use for the more complex quasi-trees. The EAPI method developed by Glover, Klingman, and Stutz [24] provides effective labeling procedures for restructuring (updating) quasi-trees by reference to such lists, and is used extensively in the updating routines of NETG.

7.0 Computational Evaluation of Solution Strategies

The computer code NETG is coded entirely in standard FORTRAN IV. We avoided the use of machine dependent operations in order to ease the transistion to various computers. The program was initially coded, debugged, and tested, using the RUN compiler on a CDC 6600 computer with a maximum main memory allocation of 130,000 words. The complete capacitated algorithm occupied $8N + 4A + 8500$ words of central memory, where N is the number of nodes and A is the number of arcs in the specific problem being solved.

Since most of the testing performed would be of a comparative nature, it was desirable to obtain a set of problems that met certain specifications and that could be made available on a repeated basis. For this reason, a generalized network problem generator (NETGENG) was developed. This code was a logical extension of the NETGEN [37] problem generator for pure network problems. All parameters in NETGEN were retained with the added feature that the user can specify a range of values from which the arc multiplier values are chosen. The problems were specifically chosen so that the effects of problem structure on solution time could be noted. The problems varied in size from 200 nodes and 1500 arcs up to 1000 nodes and 7000 arcs. Complete problem specifications and test results can be found in [17].

Earlier research with pure network problems [19], [20], [33] has established that certain factors play a critical role in determining solution speed. These are: start procedures, pivot selection techniques, degeneracy, tolerance levels, Big-M value, and pivot tie-breaking rules. The computational testing for GN problems involved varying these factors within NETG, solving generated test problems, and comparing solution times and pivots performed.

The testing was performed on a CDC 6600 computer located at the University of Texas at Austin. In each of the comparative tests, an attempt was made to execute the

codes involved during comparable time periods. The codes were timed by a clock routine supplied by CDC, which is generally accurate to two decimal places.

7.1 *Start Procedures*

The first phase of testing involved a comparison of three different start procedures. All of the starts tested were based on techniques that have proved effective for pure network problems. The first of these was the artificial start procedure. This procedure attached an artificial arc to every node in the problem. The artificial arcs were then assigned extremely large (Big-M) cost coefficients.

The second method tested was the sequential source minimum (SSM) procedure. This method made a specified number of passes, each time sequentially examining every node in the problem. If the node had an associated supply, flow was assigned to the least cost arc leading from this node to a node with positive demand (or to a node with zero demand if no positive demand node existed). The flow was set equal to the minimum of the supply, the upper bound on the arc, or the demand (if nonzero). If the flow on an arc was set equal to the supply or the demand, the associated node was eliminated from further consideration. If the process was terminated before supply and demand were exhausted, then artificial arcs were appended. For the purposes of testing, the number of passes was set to 1, 2, 3, 5, and exhaustive.

The exhaustive node supply procedure was the last start method tested. This method was similar to the sequential source minimum in the way it assigned flow to arcs. However, the procedure continued to assign flow out of a particular node until the supply at that node was exhausted or until no further arcs existed. At that point, the next node with supply was considered. Upon completion, remaining supply and demand were met by appending artificial arcs.

Each of the start methods described above was tested using two distinct pivot selection criteria. These were the node first negative and the node most negative criteria. Both methods were based on examining the nonbasic arcs leading out of a given node. The node first negative method selected the first encountered pivot eligible arc for the basis exchange. The node most negative method, on the other hand, selected the best pivot eligible arc (in terms of the magnitude of the updated cost coefficient) from the arcs out of the node. All other code parameters were held constant in all of the start procedure tests. Regardless of pivot criteria, the exhaustive pass SSM procedure proved to be the best start method in terms of resulting total solution time. It provided a reasonable trade-off between the time spent selecting an initial basis and the time recovered from using a reduced number of pivots. In some cases the exhaustive pass SSM method reduced total pivots by as much as 61% and total solution time by as much as 55% over the artificial start procedure.

7.2 *Pivot Selection Criteria*

It was noted during start procedure testing that the node most negative pivot strategy strictly dominated the node first negative strategy. Selecting the "best" arc out of a single node reduced total solution time by as much as 48%. For this reason we conducted additional testing to try to find the best pivot selection criteria.

Past experience has shown that pivot selection methods involving a candidate list can greatly decrease solution time. An S-R candidate list procedure employs an array of length R. The list contains the pointers to pivot eligible arcs selected by using the node most negative procedure R successive times. After each pivot, the best arc that is still pivot eligible in the list is selected to enter the basis. If there are no eligible arcs on the list or if the list has been used S times, the list is refilled by calling the node most negative procedure R more times. A number of variations of this method were tested. Each involved differing initial values of S and R or differing methods for dynamically adjusting these values.

Testing showed that pivot selection involving a candidate list was far superior. An initial list size of approximately 5-10 was the best. In addition, if the candidate list could not be totally filled (i.e., k candidates were found, where $k < R$) then setting $R = k$ and $S = \frac{1}{2}k$ proved to be the most effective dynamic reduction method.

7.3 *Other Procedures Tested*

The initial version of NETG had no check routines for identifying a degenerate pivot during the calculation of a basis representation. Consequently, in the presence of a degenerate pivot, the method computed unnecessary representation components and modified flows on the basis exchange cycle by a zero amount. NETG was then modified to exploit degenerate pivots, skipping the flow update procedures whenever possible. This modification reduced the total solution time by up to 25%.

Tolerance levels define ranges within which values are assumed to be zero. In order to examine the effect of tolerance values, values of 0.000001, 0.01, 0.5, and 1.0 were tested. Varying the tolerance levels effects pivot eligible arcs and this had extremely interesting effects upon solution times. The best strategy was to select a moderate tolerance value of 0.01.

The final parameter value tested was the Big-M value. (NETG did not employ a Phase I-Phase II procedure.) Testing indicated the Big-M should be set as small as possible while still insuring feasibility; e.g., in one case, the total solution time was reduced by over 42% simply by changing the Big-M value from 10000 to 150.

The last decision rule tested was one for resolving ties in the test for a minimum ratio. NETG normally selects the first encountered minimum ratio. An alternative rule for breaking pivot ties was tested that selected a minimum ratio with the largest denominator. In the majority of cases, this rule reduced the total number of pivots but not solution time.

8.0 Code Comparisons

In order to assess the efficiency of the solution procedure we compared NETG, enhanced with the newly determined decision rules, with the linear programming computer code APEX-III.

APEX-III is maintained by CDC and is operational on all CDC 6600 series and CYBER-70 series computers. The purpose of this test was to determine the advantages that specialized procedures have over standard LP approaches.

TABLE I

NETG vs. APEX-III

			NETG		APEX-III	
PROBLEM	NUMBER OF NODES	NUMBER OF ARCS	SBU's[a]	Cost[b]	SBU's	Cost
1	100	1000	7.51	$1.35	62.65	$ 11.28
2	100	1000	7.29	$1.31	80.93	$ 14.57
3	100	1000	9.70	$1.75	94.72	$ 17.05
4	250	4000	16.65	$3.00	453.02	$ 81.54
5	250	4000	14.74	$2.65	742.61	$133.67
6	500	5000	22.55	$4.06	1044.34	$187.98[c]
7	1000	6000	50.22	$9.04	1633.64	$294.06[d]

[a] CYBER-74 System Billing Unit.

[b] Computer at $0.18 per SBU.

[c] Stopped after 10,000 iterations.
Objective Function Value = 25,337,282.
Optimal Objective Function Value = 3,354,927.

[d] Stopped after 10,000 iterations.
Objective Function Value = 1,340,958,349.
Optimal Objective Function Value = 3,964,490.

The two codes were tested on seven problems generated by NETGENG. These problems ranged in size from a 50 origin by 50 destination generalized transportation problem to a 1000 node generalized transshipment problem.

The comparison between NETG and APEX-III was performed on a CDC CYBER-74 computer, compiling NETG with the CDC FTN compiler. The results are documented in Table I. The basis of comparison for these tests was a quantity called a System Billing Unit (SBU). Each procedure incurs SBU's based on the amount of CPU second used, I/O operations performed, and central memory used. In this way, SBU's may be used to compute the total cost for a job. Cost figures have been included, based on the lowest CDC price per SBU, $0.18.

The results were quite remarkable, especially when the dollar charges were compared. NETG was in some cases more than 50 times more efficient than APEX-III. In fact, problems 6 and 7 had to be prematurely terminated on APEX-III after 10,000 iterations due to the exhorbitant processing costs involved. Yet NETG solved both of these problems in fewer SBU's than APEX-III required to solve the smallest of the problems tested.[1]

[1] The research was partly supported by ONR Contract N00014-76-C-0383 with Decision Analysis and Research Institute and by Project NR047-021, ONR Contracts N00014-75-C-0616 and N00014-75-C-0569 with the Center for Cybernetic Studies, The University of Texas.

We are especially grateful to Michael Held, the Associate Editor, and the referees for their many helpful and informative suggestions for improving the clarity and readability of this paper.

References

1. AASHTIANI, H. AND MAGNANTI, T., "Implementing Primal-Dual Network Flow Algorithms," Working Paper OR 055-76, Massachusetts Institute of Technology, 1976.
2. BALACHANDRAN, V., "An Integer Generalized Transportation Model for Optimal Job Assignment in Computer Networks," *Operations Res.*, Vol. 24, No. 4 (1976), pp. 742–759.
3. BALAS, E. AND IVANESCU (HAMMER), P., "On the Generalized Transportation Problem," *Management Sci.* Vol. 1 (1964), pp. 188–202.
4. BARR, R., GLOVER, F. AND KLINGMAN, D., "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes," *Math. Programming*, Vol. 7 No. 1 (1974), pp. 60–87.
5. ———, ——— AND ———, "Enhancements of Spanning Tree Labeling Procedures for Network Optimization," Research Report CCS 262, Center for Cybernetic Studies, University of Texas at Austin, 1976.
6. ———, ——— AND ———, "The Alternating Basis Algorithm for Assignment Problems," *Math. Programming*, Vol. 13 (1977), pp. 1–13.
7. BHAUMIK, G., *Optimum Operating Policies of a Water Distribution System with Losses*, Unpublished Dissertation, University of Texas at Austin, August, 1973.
8. ———, AND JENSEN, P., "A Computationally Efficient Algorithm for the Network with Gains Problem," Working Paper, Department of Mechanical Engineering, University of Texas at Austin, 1974.
9. BRADLEY, G. "Survey of Deterministic Networks," *AIIE Transactions*, Vol. 7, No. 3 (1975), pp. 222–234.
10. ———, BROWN, G. AND GRAVES, G., "Design and Implementation of Large Scale Primal Transshipment Algorithms," *Management Sci.* Vol. 24 (1977), pp. 1–35.
11. CHARNES, A. AND COOPER, W., *Management Models and Industrial Applications of Linear Programming*, Vols. I and II, Wiley, New York, 1961.
12. CRUM, R., "Cash Management in the Multinational Firm: A Constrained Generalized Network Approach," Working Paper, University of Florida, Gainesville, Florida, 1976.
13. DANTZIG, G., *Linear Programming and Extensions*, Princeton Univ. Press, Princeton, N.J., 1963.
14. EISEMANN, D., "The Generalized Stepping Stone Method for the Machine Loading Model," *Management Sci.*, Vol. 11, No. 1 (1964), pp. 154–177.
15. ELAM, J., GLOVER, F. AND KLINGMAN, D., "A Strongly Convergent Primal Algorithm for Generalized Networks," Research Report CCS 288, Center for Cybernetic Studies, University of Texas at Austin, 1977.
16. GILLIAM, G. AND TURNER, J., "A Profile Analysis Network Model to Reduce the Size of Microdata Files," Working Paper, Office of Tax Analysis, Office of the Secretary of the Treasury, Washington, D.C., 1974.
17. GLOVER, F., HULTZ, J., KLINGMAN, D. AND STUTZ, J., "A New Computer-Based Planning Tool," Research Report CCS 289, Center for Cybernetic Studies, University of Texas at Austin, 1977.

18. GLOVER, F., KARNEY, D. AND KLINGMAN, D., "The Augmented Predecessor Index Method for Locating Stepping Stone Paths and Assigning Dual Prices in Distribution Problems," *Transportation Sci.* Vol. 6, No. 2 (1972), pp. 171–179.
19. ———, ——— AND ———, "Implementation and Computational Study on Start Procedures and Basis Change Criteria for a Primal Network Code," *Networks*, Vol. 4, No. 3 (1974), pp. 191–212.
20. ———, ———, ——— AND NAPIER, A., "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," *Management Sci.* Vol. 20, No. 5 (1974), pp. 793–819.
21. ——— AND KLINGMAN, D., "On the Equivalence of Some Generalized Network Problems to Pure Network Problems," Math. *programming*, Vol. 4, No. 3 (1973), pp. 351–361.
22. ——— AND ———, "A Note on Computational Simplifications in Solving Generalized Transportation Problems," *Transportation Sci.*, Vol. 7, No. 4 (1973), pp. 351–361.
23. ———, ——— AND MCMILLAN, C., "The NETFORM Concept," Proceedings of ACM'77, Seattle, October 1977.
24. ———, ——— AND STUTZ, J., "Extensions of the Augmented Predecessor Index Method to Generalized Network Problems," *Transportation Sci.*, Vol. 7, No. 4 (1973), pp. 377–384.
25. ———, ——— AND ———, "Implementation and Computational Study of a Generalized Network Code," Presented at the 44th National ORSA Conference, San Diego, California, 1973.
26. ———, ——— AND ———, "The Augmented Threaded Index Method for Network Optimization," *INFOR*, Vol. 12, No. 3 (1974), pp. 293–298.
27. ——— AND MULVEY, J., "Equivalence of the 0-1 Integer Programming Problem to Discrete Generalized and Pure Networks," MSRS 75-19, University of Colorado, Boulder, Colorado, 1975.
28. HARRIS, G., "A Code for the Transportation Problem of Linear Programming," J. Assoc. Comput. Mach. Vol. 23, No. 1 (1976), pp. 155–157.
29. HULTZ, J., *Algorithms and Applications for Generalized Networks*, Unpublished Dissertation, University of Texas at Austin, 1976.
30. JEWELL, W., "Optimal Flow Through Networks with Gains," *Operations Res.* Vol. 10, No. 4 (1962), pp. 476–499.
31. JOHNSON, E., "Programming in Networks and Graphs," ORC Report 65-1, University of California at Berkeley, 1965.
32. ———, "Networks and Basic Solutions," *Operations Res.*, Vol. 14, No. 4 (1966), pp. 619–623.
33. KARNEY, D. AND KLINGMAN, D., "Implementation and Computational Study on an In-Core Out-of-Core Primal Network Code," *Operations Res.* Vol. 24 (1976).
34. KAZEMERSKY, P., *A Computer Code for Refueling and Energy Scheduling Containing an Evaluator of Nuclear Decisions for Operation*, Unpublished Dissertation, Ohio State University, 1974.
35. KIM, Y., "An Optimal Computational Approach to the Analysis of a Generalized Network of Copper Refining Process," Presented at the Joint ORSA/TIMS/AIIE Conference, Atlantic City, New Jersey, 1972.
36. KLINGMAN, D., NAPIER, A., AND ROSS, G., "A Computational Study of the Effects of Problem Dimensions on Solution Times for Transportation Problems," J. Assoc. Comput. Mach., Vol. 22, No. 3 (1975), pp. 413–424.
37. ———, ——— AND STUTZ, J., "NETGEN—A Program for Generating Large Scale (Un) Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," *Management Sci.* Vol. 20, No. 5 (1974), pp. 814–821.
38. LANGLEY, R., *Continuous and Integer Generalized Flow Problems*, Unpublished Dissertation, Georgia Institute of Technology, 1973.
39. ———, KENNINGTON, J. AND SHETTY, C., "Computational Devices for the Capacitated Transportation Problem," *Naval Res. Logist. Quart.*, Vol. 21, No. 4 (1974), pp. 637–647.
40. MAURRAS, J., "Optimization of the Flow Through Networks with Gains," *Math. Programming*, Vol. 3 (1972), pp. 135–144.
41. SRINIVASAN, V. AND THOMPSON, G., "Accelerated Algorithms for Labeling and Relabeling of Trees with Applications for Distribution Problems," J. Assoc. Comput. Mach., Vol. 19, No. 4 (1972), pp. 712–726.
42. TAVIS, L., CRUM, R. AND KLINGMAN, D., "Implementation of Large-Scale Financial Planning Models: Solution Efficient Transformations," Research Report CCS 267, Center for Cybernetic Studies, University of Texas at Austin, 1976.
43. WAGNER, H., *Principles of Operations Research*, Prentice-Hall, Englewood Cliffs, N.J., 1969.