AD 048 297 TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES F/G 9/2
A FUNDAMENTAL COMPUTER-BASED PLANNING TOO--ETC(U)
JUN 77 F. GLOVER, J HULTZ, D. KLINGMAN N0014-75-C-0616
UNCLASSIFIED CCS-307 N/L

| OF |
ADA
048297

END
DATE
FILMED
1 -79
DDC

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

Research Report CCS 307

GENERALIZED NETWORKS: A FUNDAMENTAL COMPUTER-BASED PLANNING TOOL

by

F. Glover *

J. Hultz **

D. Klingman ***

J. Stutz ****

June 1977

* Professor of Management Science, University of Colorado, Boulder, CO 80302.

** Senior Analyst, Analysis, Research, and Computation, Inc., P.O. Box 4067, Austin, Texas 78765.

*** Professor of Operations Research and Computer Sciences, BEB 608, University of Texas, Austin, Texas 78712

**** Associate Professor of Operations Research and Computer Sciences, BEB 613, University of Texas, Austin, Texas 78712.

This 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. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 203E
The University of Texas
Austin, TX 78712
(512) 471-3322

ABSTRACT

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.

Part I of the paper is a non-technical account of how generalized networks are used to model a diversity of significant practical problems. First 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 based on NETFORM (network formulation) principles in which generalized networks form a major, but not the only, component of the model.

Part II is a technical exposition of the design and analysis of computer solution techniques for large-scale GN problems. It contains a study of GN solution strategies within the framework of specializations of the primal simplex method. Here we identify an efficient solution procedure that derives 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. The computer memory requirements of our 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.

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 non-scientific 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. Part I of the paper is a non-technical account of how generalized networks are used to model a diversity of signifi-

cant practical problems. Using a graphic 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, based on NETFORM (network formulation) principles, in which generalized networks form a major, but not the only, component of the model. The NETFORM concept 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 the NETFORM approach and the role of generalized networks within it, we describe two real-world problems which have been solved by its use.

Part II of the paper is a technical exposition of the design and analysis of computer solution techniques for large-scale GN problems. It contains an in-depth computational study of GN solution strategies within the framework of specializations of the primal simplex method. Here we identify an efficient solution procedure that derives 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 other words, our 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 our 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.

PART I - GENERALIZED NETWORK MODEL

1.0 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 non-zero 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 non-zero 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 non-zero 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 non-zero entries, at least one of these is -1. In this way, a directed arc is "formed" from the node associated with the -1 to the node associated with the other non-zero entry. If both entries are -1, the arc may be directed either way. Columns with single non-zero 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 non-

zero 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:

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

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

$$0 \leq x_{24} \leq 5,\ 0 \leq x_{32} \leq 3,\ 0 \leq x_{34} \leq 7$$

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 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 non-zero 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

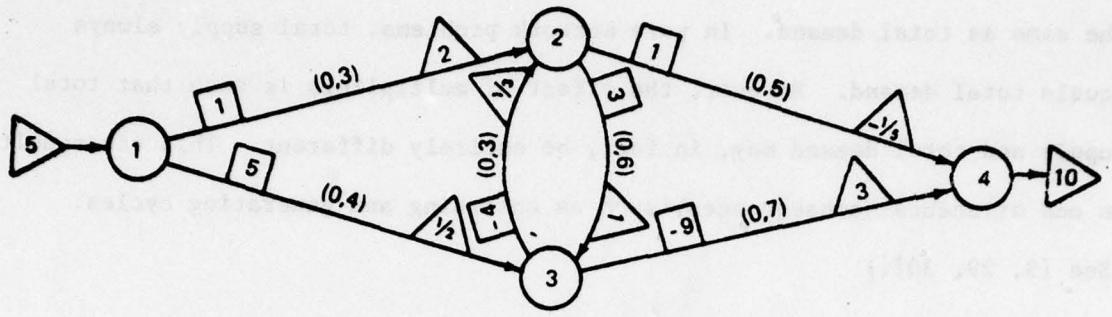


Figure 1

Generalized Network

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 non-zero 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].)

2.0 APPLICATIONS OF GENERALIZED NETWORKS

Generalized networks can be used to model numerous problems for which there no pure network equivalent. There are essentially two ways in which the ltipliers on the arcs of generalized networks can function. They can act simply to modify the amount of flow of particular goods 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 non-zero 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].

3.0 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, sometimes called NETFORM principles, enable even more complex applications to be modeled and solved as integer GN problems. In fact, the NETFORM principles and techniques make it possible to model any 0-1 LP problem as an integer GN problem [23, 27]. These procedures extend quite naturally to accommodate mixed integer 0-1 LP problems where the continuous part of the problem is a transportation, transshipment or generalized network problem itself. Reference [42] illustrates a NETFORM extension and 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 NETFORM representation is mathematically equivalent to any of the algebraic representations that can be arrived at by customary mathematical programming formulation techniques. However, by the creation of the network-related structures, which may or may not be implicit in any customary formulation, the NETFORM representation permits the application of specialized solution methods, tailored to employ algorithmic advances discussed in Part II, for exploiting the graphical relationships of network components. The remainder of this section briefly describes the basic principles of the NETFORM approach and discusses two applications which have profited by its use.

Figure 2 illustrates one of the useful modeling devices of the NETFORM approach 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.

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

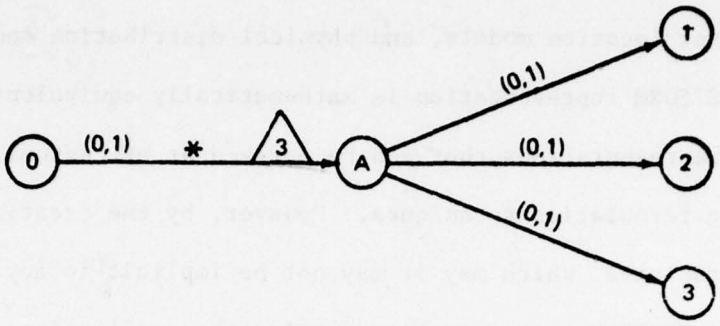


Figure 2

Generalized Network with Integer Flow Restrictions

(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 NETFORM modeling tool has a variety of important uses, as demonstrated more concretely by the following two real-world applications.

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.

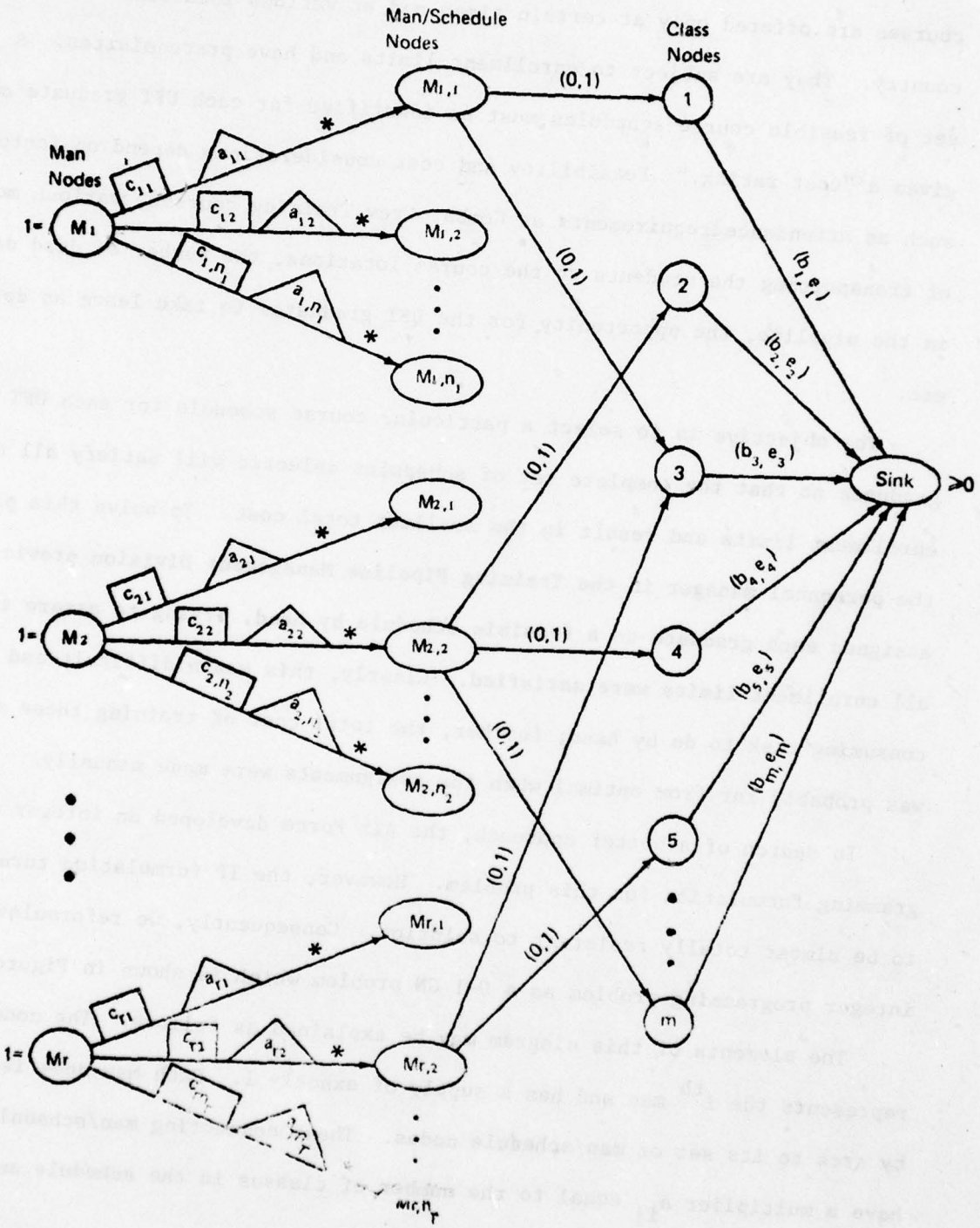
The objective is to select a particular course schedule for each UFT graduate so that the complete set of schedules selected will satisfy all class enrollment limits and result in the smallest total cost. To solve this problem, the personnel manager in the Training Pipeline Management Division previously 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 i^{th} 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

Figure 3

UFT NETFORM Formulation



c_{ij} equal to the cost of assigning man i to his j^{th} 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 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 NETFORM.

Refueling Nuclear Reactors

A mixed integer programming problem for determining the minimum cost refueling scheduling for nuclear reactors has been modeled by Kazmersky [34]. The solution to the problem was facilitated by the use of the NETFORM concept. Although the mixed integer programming formulation bore no apparent connection to networks, we discovered a way to express the problem by a convenient NETFORM representation after interacting closely with Dr. Kazmersky. The NETFORM representation was not only equivalent to the original formulation but also succeeded in reducing its size. We will forego the details of the transformation of the original problem to a 0-1 GN problem because the steps were somewhat intricate and the original formulation by itself consumed more than twenty pages of [34]. However, we were able to exploit the 0-1 GN formulation by developing a branch and bound solution procedure which employed the GN optimization procedures discussed in Part II of this paper. Four versions of the refueling problem were solved using data supplied by the TVA. Solution times for the first three versions, using MPSX on an IBM 370/168, were half an hour to two hours. By contrast, the same problems were easily solved in 10 to 20 minutes using the 0-1 GN formulation and the specialized branch and bound solution approach. The fourth version, which involved 173 constraints, 126 zero-one variables, and 511 continuous variables, was by far the most difficult. The original mixed integer formulation was run for seven hours on an IBM 370/168, again using MPSX, and was then taken off the machine to avoid further computer run costs. The best (minimum cost) solution obtained had an objective function value of $136,173,440. With a 30 minute time limit imposed on the 0-1 GN solution effort, a solution was obtained that was more than $10,000,000 cheaper,

with an objective function value of $125,174,727. This application demonstrates that by using the NETFORM approach one may obtain solutions to problems that are too complex to be solved optimally (within practical time limits) by standard approaches.

4.0 MOTIVATIONS FOR USING GN MODELS

The two important advantages to adopting a GN formulation where appropriate have been outlined in Part I. 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 non-scientific users. The major incentive for using GN models is that they, and a variety of problems with GN components, can be solved with a remarkable degree of efficiency. The computer software described in Part II of this paper is capable of solving large scale GN problems up to fifty times faster than state of the art LP codes.

PART II - DESIGN AND ANALYSIS OF LARGE-SCALE GENERALIZED NETWORK COMPUTER PROCEDURES

1.0 OVERVIEW

Part II presents 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 transportation [18, 20, 28, 36, 39, 43] and transshipment [1, 4, 5, 10, 19, 26, 33, 37, 41] computer codes. 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.

2.0 PROBLEM STATEMENT

$$\text{Minimize } c^T x \qquad (1)$$

Subject to:

$$Gx = b \qquad (2)$$

$$0 \leq x \leq u, \qquad (3)$$

where each column of the coefficient matrix G contains at most two non-zero entries. It will be assumed that u is finite (e.g., using "regularization" [11] if necessary). The problem as stated may be conceptualized as a graph containing vertices and non-directed edges. However, we will further assume that if a column of G has two non-zero entries, then at least one of these is negative and by appropriate scaling has a value of -1. This allows the problem to be presented in a directed graph (digraph) setting, as noted in Part I.

Each column of G is associated with a directed edge (arc) of the digraph and each row is associated with a vertex (node). An arc runs from the tail node, associated with the -1 coefficient, to the head node, associated with the other non-zero entry. In the case of a single non-zero entry in the column of G, the associated arc is called a self-loop and is incident on a single node. The flow on the arc is defined to be the value assigned to the corresponding component of x (i.e., to the variable whose column of G gives rise to the arc). Thus, by this association between arcs and variables, each arc has an associated cost per unit flow (component of c) and an upper bound on the flow (component of u). The coefficient in the column of G associated with the head of the arc is the arc multiplier.

The right-hand side value (component of b) associated with a particular node determines whether flow will be inserted or removed from the network at that node. If the right-hand side value is negative, flow is inserted and the node is called a source node. Likewise, if the right-hand side value is positive, the node is called a sink node and flow is removed at that point. If a parti-

cular node has both arcs heading into it as well as out of it, then it is called a transshipment node. All other nodes will be either pure sources or pure sinks.

Since the computer code NETG, on which this study is based, employs a specialization of the primal simplex method, a brief discussion of this specialization is in order. We begin by examining the basis structure for this specialization.

It may be assumed without loss of generality that G has rank m, where m is the number of rows in G. Otherwise, the problem is equivalent to a set of disjoint minimum cost flow networks (see [21]). Any basis for the problem, then, will be composed of m linearly independent column vectors selected from G. Graphically this corresponds to a set of m arcs incident on the m nodes of the problem. It has been shown [15, 24, 38, 40] that this graph will be composed of a set of disjoint quasi-trees. Each quasi-tree is a simple tree to which a single arc is added. The additional arc forms exactly one cycle, which is a unique series of arcs leading from a node back to that node. By convention, to allow this characterization to apply to the case in which the additional arc is a self-loop, a self-loop is regarded as a cycle.

The EAPI method, on which NETG is based, is specifically designed to store the bases of generalized network problems in the graphical quasi-tree form. All of the primal simplex operations are carried out by working with the basis graphs which are stored using linked list procedures. The inherent advantages of this method will be fully described subsequently.

3.0 DATA STRUCTURES IN THE EAPI METHOD

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. In many cases, the special structure of a generalized network problem can be detected by a primal simplex LP code; this information is then used to reduce storage requirements and to simplify operations. 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, and current LP codes are of course designed to take advantage of sparsity to store data economically. When the problem is transformed to digraph form, storage may be reduced even further. By the use of simple ordered lists to capture the digraph structure, NETG is designed to store only the head node identifier, the cost coefficient, the non-zero multiplier, and the upper bound for each column of the coefficient matrix. In this way, problem data can often be resident in fast main memory for extremely large problems.

As noted in section 2, 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. 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.

The original problem data, compactly stored, and the basis matrix, stored via linked lists, are the only data elements that need to be kept by a specialized code such as NETG. The advantage here is that a basis inverse need not be maintained. Inverses generally require considerable amounts of storage and involve numerous arithmetic operations to utilize and maintain them. These arithmetic operations additionally require considerable computer time and also introduce numerical inaccuracies. Instead, the specialized labeling rules of the EAPI method operate on the basis graph in a manner that obviates the use of a basis inverse.

The EAPI method [24] orients each quasi-tree so that the cycle is conceptually located at the top, with attached trees hanging downward. This orientation is called a rooted cycle. The linked lists and labeling procedures provide the means of traversing the tree in both upward and downward directions.

Without detailing minutely the rules and processes of these procedures, it is useful to sketch their main functions. In particular, the two types of quasi-tree traversals accommodated by the procedures are used to carry out basis exchange steps. Upward traversal is associated with operations normally requiring pre-multiplication by the inverse, such as determining the representation of an entering vector (arc). This operation is performed by traversing the unique paths from selected vertices up to their associated cycle(s). Simultaneously, the associated triangular system of equations is solved, in effect, by back substitution. A directed trace of the cycle(s) completes the operation. The process of determining the representation of a vector requires traversing at most two quasi-trees and generates only the non-zero entries of this representation.

Downward quasi-tree traversal is analogous to post-multiplication by the inverse. This operation is used to calculate updated dual variable values. In a network interpretation, there is a dual variable associated with each of the nodes in the problem. These are often referred to in the literature as node potentials. At each iteration, new dual values associated with a subset of the nodes in a single quasi-tree must be determined. A single dual value associated with a node on the cycle may be determined [22, 24]. The resulting system of equations is triangular, and may be solved by traversing downward through the quasi-tree, again employing back substitution. This process automatically restricts attention to only those dual values that change during a basis exchange.

4.0 COMPUTATIONAL TESTING

The computer code NETG is coded entirely in standard FORTRAN IV. We avoided the use of machine dependent operations in order to ease the transition 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].

Based upon earlier research with pure network problems [19, 20, 33], it has been 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 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 code uses a real-time clock routine supplied by CDC, and is generally accurate to two decimal places.

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. It attached an artificial arc (self-loop) to every node in the problem. The artificial arcs were then assigned extremely large (Big-M) cost coefficients. The artificial method was the fastest start procedure to execute, but it did not yield an advanced initial basis.

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 associated supply, flow was assigned to the arc having the least cost that led to a node with positive demand, or, if there was no node with positive demand, to a node with zero demand. The flow was set equal to the minimum of the supply, the upper bound on the arc, or the demand (if non-zero). 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 non-basic 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.

Pivot Selection Criteria

It was noted during start procedure testing that the node most negative pivot strategy strictly dominated the node first negative strategy. The indication was that it would be worthwhile to spend time searching for the "best" pivot eligible arc. Selecting the "best" arc out of a single node, as the most negative method did, 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 to methods that did not. It was found that an initial list size of approximately 5-10 was the best. In addition, if the candidate list cannot be

totally filled (i.e., k candidates are found, where $k < R$) then setting $R = k$ and $S = \frac{1}{2}k$ proved to be the most effective dynamic reduction method.

Flow Update Procedures

The initial version of NETG often performed wasted operations in the process of updating the flows on basic arcs. Since it had no check routines for identifying a degenerate pivot during the calculation of a representation, unnecessary representation components were found and flows were modified by a zero amount. NETG was modified to exploit degenerate pivots. It identified a degenerate pivot and totally skipped the flow update procedures whenever possible. Both of these recalculated the representation during flow updates. Another version was created to allow for storing the representation vector.

It proved advantageous to check for degenerate pivots. Although the total number of pivots was not necessarily reduced, the total solution time was reduced by up to 25%. However, the tests also indicated that it was not advantageous to go one step further and keep a representation array. The overhead of maintaining this extra information was apparently not offset by a reduction in other calculations.

Pivot Tolerance

Tolerance levels are used in computer programming to avoid the problems caused by round-off error. These tolerances define ranges within which values are assumed to be zero. Such tolerances must be used in the testing of updated cost coefficients. Negative coefficients indicate pivot eligible arcs; however, very small negative numbers may actually be equivalent to 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 had extremely interesting effects upon solution times. All levels produced solutions with the same optimum objective function value. The value of the tolerance dramatically changed the choices of pivot eligible arcs. Extremely small values forced the code to perform a series of disadvantageous pivots. On the other hand, large tolerances allowed the code initially to overlook advantageous pivot selections. The best strategy was to select a moderate tolerance value of 0.01.

Big-M Value

The final parameter value tested was the Big-M value. NETG did not employ a Phase I-Phase II procedure, therefore an exact value of Big-M had to be selected. All of the test problems had cost ranges of 1-100. Experience with pure network computer codes indicated that Big-M should be set as small as possible while still insuring feasibility. Big-M values of 10000, 2000, 1000, 500, 250, 150, and 100 were all tested. When 100 was eliminated because of resultant infeasibilities, a value of 150 was clearly the best in terms of total solution time. In one case, the total solution time was reduced by over 42% simply by changing the Big-M value from 10000 to 150.

Breaking Pivot Ties

The last decision rule tested was one for resolving ties in the test for a minimum ratio. NETG simply selected the most convenient (often the first encountered) minimum ratio. A new version was developed to test a heuristic rule for breaking pivot ties. The rule specified that the ratio with the largest denominator be selected from among the set of tied ratios.

In the majority of cases, this rule reduced the total number of pivots performed. However, the reduction was not enough to offset the increased number of operations involved with breaking the tie.

Code Comparisons

In order to assess the efficiency of the solution procedure described above, we compared NETG, enhanced with the newly determined decision rules, with the well-known 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.

The two codes were tested using seven problems generated by NETGENG. The specifications of these problems are given in Table I. They ranged in size from a 50 origin by 50 destination generalized transportation problem to a 1000 node generalized transshipment problem.

To perform the comparison between NETG and APEX-III, a CDC CYBER-74 computer was used and NETG was compiled using the CDC FTN compiler. The results are documented in Table II. 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 seconds 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.

Table I

Problem Specifications

Problem	Nodes	Sources	Sinks	Arcs	Multiplier Range		Cost Range		Upper Bound Range		Transshipment		Total Supply	Percent High Cost	Random Number Seed
					Min	Max	Min	Max	Min	Max	Sources	Sinks			
1	100	50	50	1000	.5	1.5	1	100	---		0	0	100000	0	13502460
2	100	5	50	1000	.5	1.5	1	100	---		2	10	100000	0	13502460
3	100	5	50	1000	.5	1.5	1	100	100	100	2	10	100000	0	13502460
4	250	125	125	4000	.5	1.5	1	100	---		0	0	100000	0	13502460
5	250	15	75	4000	.5	1.5	1	100	---		5	15	100000	0	13502460
6	500	50	200	5000	.3	1.7	1	100	---		10	50	100000	0	13502460
7	1000	200	500	6000	.2	1.4	1	100	---		25	75	100000	0	13502460

Table II

NETG vs. APEX-III

Problem	NETG		APEX-III	
	SBU's[a]	Cost[b]	SBU's	Cost
1	7.51	$1.35	62.65	$ 11.28
2	7.29	$1.31	80.93	$ 14.57
3	9.70	$1.75	94.72	$ 17.05
4	16.65	$3.00	453.02	$ 81.54
5	14.74	$2.65	742.61	$133.67
6	22.55	$4.06	1044.34	$187.98[c]
7	50.22	$9.04	1633.64	$294.06[d]

[a]CYBER-74 System Billing Unit.

[b]Computed 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 comparison of NETG with APEX-III succeeded in proving that generalized network problems can be efficiently solved by using codes specially adapted for that purpose. The ability to solve large scale generalized network problems with such a high degree of efficiency allows management scientists to begin using this important class of problems as a fundamental computer based planning tool.

REFERENCES

1. H. Aashtiani and T. Magnanti, "Implementing Primal-Dual Network Flow Algorithms," Working Paper OR 055-76, Massachusetts Institute of Technology, 1976.

2. V. Balachandran, "An Integer Generalized Transportation Model for Optimal Job Assignment in Computer Networks," *Operations Research*, 24, 4 (1976), 742-759.

3. E. Balas and P. Ivanescu (Hammer), "On the Generalized Transportation Problem," *Management Science*, 1 (1964), 188-202.

4. R. Barr, F. Glover, and D. Klingman, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes," *Mathematical Programming*, 7, 1 (1974), 60-87.

5. R. Barr, F. Glover, and D. Klingman, "Enhancements of Spanning Tree Labeling Procedures for Network Optimization," Research Report CCS 262, Center for Cybernetic Studies, University of Texas at Austin, 1976.

6. R. Barr, F. Glover, and D. Klingman, "The Alternating Basis Algorithm for Assignment Problems," Research Report CCS 263, Center for Cybernetic Studies, University of Texas at Austin, 1977.

7. G. Bhaumik, *Optimum Operating Policies of a Water Distribution System with Losses*, Unpublished Dissertation, University of Texas at Austin, August, 1973.

8. G. Bhaumik and P. Jenson, "A Computationally Efficient Algorithm for the Network with Gains Problem," Working Paper, Department of Mechanical Engineering, University of Texas at Austin, 1974.

9. G. Bradley, "Survey of Deterministic Networks," *AIIE Transactions*, 7, 3 (1975), 222-234.

10. G. Bradley, G. Brown, and G. Graves, "Design and Implementation of Large Scale Primal Transshipment Algorithms," Technical Report NPS55BZBW76091, Naval Postgraduate School, Monterey, California, 1976.

11. A. Charnes and W. Cooper, *Management Models and Industrial Applications of Linear Programming*, Vols. I and II, Wiley, New York, 1961.

12. R. Crum, "Cash Management in the Multinational Firm: A Constrained Generalized Network Approach," Working Paper, The University of Florida, Gainesville, Florida, 1976.

13. G. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.

14. K. Eisemann, "The Generalized Stepping Stone Method for the Machine Loading Model," *Management Science*, 11, 1 (1964), 154-177.

15. J. Elam, F. Glover, and D. Klingman, "A Strongly Convergent Primal Algorithm for Generalized Networks," Research Report CCS 288, Center for Cybernetic Studies, University of Texas at Austin, 1977.

16. G. Gilliam and J. Turner, "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. F. Glover, J. Hultz, D. Klingman, and J. Stutz, "A New Computer-Based Planning Tool," Research Report CCS 289, Center for Cybernetic Studies, University of Texas at Austin, 1977.

18. F. Glover, 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, 2 (1972), 171-179.

19. F. Glover, D. Karney, and D. Klingman, "Implementation and Computational Study on Start Procedures and Basis Change Criteria for a Primal Network Code," *Networks*, 4, 3 (1974), 191-212.

20. F. Glover, 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), 793-819.

21. F. Glover and D. Klingman, "On the Equivalence of Some Generalized Network Problems to Pure Network Problems," *Mathematical Programming*, 4, 3 (1973), 351-361.

22. F. Glover and D. Klingman, "A Note on Computational Simplifications in Solving Generalized Transportation Problems," *Transportation Science*, 7, 4 (1973), 351-361.

23. F. Glover, D. Klingman, and C. McMillan, "The NETFORM Concept," Research Report CCS 281, Center for Cybernetic Studies, University of Texas at Austin, 1977.

24. F. Glover, D. Klingman, and J. Stutz, "Extensions of the Augmented Predecessor Index Method to Generalized Network Problems," *Transportation Science*, 7, 4 (1973), 377-384.

25. F. Glover, D. Klingman, and J. Stutz, "Implementation and Computational Study of a Generalized Network Code," Presented at the 44th National ORSA Conference, San Diego, California, 1973.

26. F. Glover, D. Klingman, and J. Stutz, "The Augmented Threaded Index Method for Network Optimization," *INFOR*, 12, 3 (1974), 293-298.

27. F. Glover and J. Mulvey, "Equivalence of the 0-1 Integer Programming Problem to Discrete Generalized and Pure Networks," MSRS 75-19, University of Colorado, Boulder, Colorado, 1975.

28. B. Harris, "A Code for the Transportation Problem of Linear Programming," *JACM*, 23, 1 (1976), 155-157.

29. J. Hultz, *Algorithms and Applications for Generalized Networks*, Unpublished Dissertation, University of Texas at Austin, 1976.

30. W. Jewell, "Optimal Flow Through Networks with Gains," *Operations Research*, 10, 4 (1962), 476-499.

31. E. Johnson, "Programming in Networks and Graphs," ORC Report 65-1, University of California at Berkeley, 1965.

32. E. Johnson, "Networks and Basic Solutions," *Operations Research*, 14, 4 (1966), 619-623.

33. D. Karney and D. Klingman, "Implementation and Computational Study on an In-Core Out-of-Core Primal Network Code," *Operations Research*, 24, (1976).

34. P. Kazemersky, *A Computer Code for Refueling and Energy Scheduling Containing an Evaluator of Nuclear Decisions for Operation*, Unpublished Dissertation, Ohio State University, 1974.

35. Y. Kim, "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. D. Klingman, A. Napier, and G. Ross, "A Computational Study of the Effects of Problem Dimensions on Solution Times for Transportation Problems," *JACM*, 22, 3 (1975), 413-424.

37. D. Klingman, 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), 814-821.

38. R. Langley, *Continuous and Integer Generalized Flow Problems*, Unpublished Dissertation, Georgia Institute of Technology, 1973.

39. R. Langley, J. Kennington, and C. Shetty, "Computational Devices for the Capacitated Transportation Problem," *Naval Research Logistics Quarterly*, 21, 4 (1974), 637-647.

40. J. Maurras, "Optimization of the Flow Through Networks with Gains," *Mathematical Programming*, 3, (1972), 135-144.

41. V. Srinivasan and G. Thompson, "Accelerated Algorithms for Labeling and Relabeling of Trees with Applications for Distribution Problems," *JACM*, 19. 4 (1972), 712-726.

42. L. Tavis, R. Crum, and D. Klingman, "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. H. Wagner, *Principles of Operations Research*, Prentice-Hall, Englewood Cliffs, New Jersey, 1969.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

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

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Center for Cybernetic Studies The University of Texas at Austin	Unclassified 2b. GROUP

3. REPORT TITLE

Generalized Networks: A Fundamental Computer-Based Planning Tool

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

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

Fred Glover J. Stutz
J. Hultz
D. Klingman

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1977	34	43

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-75-C-0569;0616 b. PROJECT NO. NR047-021 c. d.	Center for Cybernetic Studies Research Report CCS 307 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

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

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

13. ABSTRACT

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.

Part I of the paper is a non-technical account of how generalized networks are used to model a diversity of significant practical problems. First 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 techniques based on NETFORM (network formulation) principles in which generalized networks form a major, but not the only, component of the model.

Part II is a technical exposition of the design and analysis of computer solution techniques for large-scale GN problems. It contains a study of GN solution strategies within the framework of specializations of the primal simplex method. Here we identify an efficient solution procedure that derives 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. The computer memory requirements of our 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.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Networks						
Generalized Networks						
Networks with Gains						
Network Computation						
Graphs						