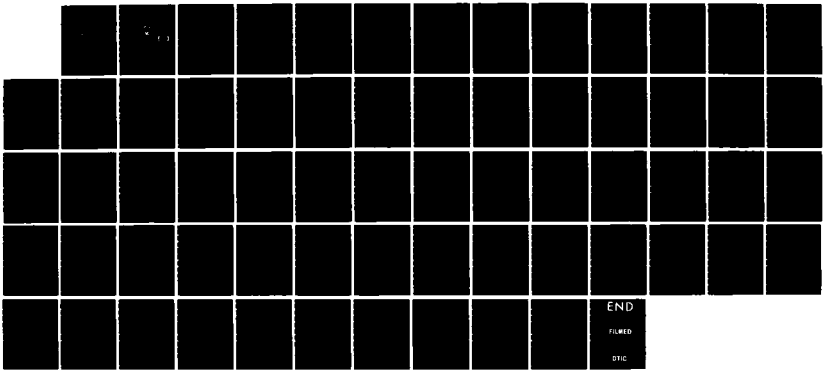
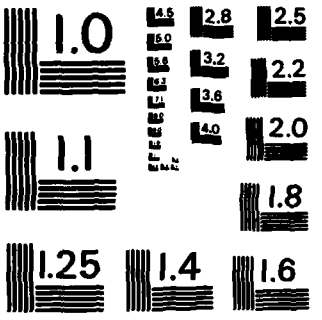
AD-A148 499 THE EFFECT OF STRUCTURE ON THE SOLUTION TIMES OF MINIMUM COST TRANSPORTAT.. (U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA W R BONWIT JUN 84 1/1

UNCLASSIFIED F/G 9/2 NL





(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THE EFFECT OF STRUCTURE ON THE SOLUTION TIMES OF MINIMUM COST TRANSPORTATION AND MULTI-ECHELON NETWORK FLOW PROBLEMS

by

Willard R. Bonwit, Jr.

June 1984

Thesis Advisor: R.K. Wood

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A148499	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Effect of Structure on the Solution Times of Minimum Cost Transportation and Multi-Echelon Network Flow Problems		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Willard R. Bonwit, Jr.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE June 1984
		13. NUMBER OF PAGES 63
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Network Generator
Structured Networks

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Researchers require benchmark test problems to evaluate the speed of computer codes designed to solve minimum cost network flow problems. To date, the only universally available test problems developed for that purpose are randomly generated. In practice, however, real-world network problems solve faster than random network problems. This thesis examines the effect on solution time resulting from applying structure, produced through simulation of real-world phenomena, to test networks. An

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (Continued)

efficient computer code, VSGEN, is developed which generates structured transportation and multi-echelon networks. Various types of structure, including unit flow cost, network topology and arc capacity, reduced the time required to solve the test networks an average of 26%, when using a primal network simplex solver, GNET.

The parameter Big M used in primal simplex algorithms may affect solution times differently in structured versus unstructured networks. VSGEN is used to investigate this possibility. A bound on the minimum Big M is first developed for bipartite networks. This bound is sharper than the default bound used in GNET, but it does not reduce solution times in either structured or unstructured problems. Even the best possible bound reduces solution times by only 10%, on average.

Approved for public release; distribution unlimited

The Effect of Structure on the Solution Times of Minimum Cost Transportation and Multi-Echelon Network Flow Problems

by

Willard R. Bonwit, Jr.
Lieutenant, United States Navy
B.S., United States Naval Academy, 1978

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 1984

Author: _____

Approved by: _____
Thesis Advisor

_____
Second Reader

_____
Chairman, Department of Operations Research

_____
Dean of Information and Policy Sciences

ABSTRACT

Researchers require benchmark test problems to evaluate the speed of computer codes designed to solve minimum cost network flow problems. To date, the only universally available test problems developed for that purpose are randomly generated. In practice, however, real-world network problems solve faster than random network problems. This thesis examines the effect on solution time resulting from applying structure, produced through simulation of real-world phenomena, to test networks. An efficient computer code, VSGEN, is developed which generates structured transportation and multi-echelon networks. Various types of structure, including unit flow cost, network topology and arc capacity, reduced the time required to solve the test networks an average of 26%, when using a primal network simplex solver, GNET.

The parameter Big M used in primal simplex algorithms may affect solution times differently in structured versus unstructured networks. VSGEN is used to investigate this possibility. A bound on the minimum Big M is first developed for bipartite networks. This bound is sharper than the default bound used in GNET, but it does not reduce solution times in either structured or unstructured problems. Even the best possible bound reduces solution times by only 10%, on average.

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

Modern high speed computers have dramatically increased our ability to solve the large minimum cost network flow problems arising in industrial, governmental and military settings. The minimum cost network flow problem is the problem of transmitting a given supply of a single commodity through a network to meet a specified demand at the lowest cost. Flow is directed through the network on arcs with linear cost functions which represent the effort required to transmit flow on a given arc. These problems arise through many sources including inventory, scheduling, distribution, assignment, and other problems. The broad applicability of research on this topic has resulted in numerous competitive computer codes which solve the minimum cost network flow problem. Users and developers of these codes require the ability to compare competing algorithms and codes for problems with a variety of structures. Accurate comparisons can result only from testing the competitors on a set of standard test problems, but up to now, only one attempt has been made to derive such a set of test problems. This attempt is a network generator code, NETGEN, written by Klingman, Napier, and Stutz [Ref. 11] in 1974. Since their

initial work, a large number of articles, proprietary network solution codes, and texts have used networks generated by NETGEN as benchmark problems [Refs. 4, 8, 9, 14].

NETGEN constructs test problems of three types: transportation networks, assignment networks, and general minimum cost flow networks. The problems generated are essentially unstructured; the user controls the maximum arc cost, percentage of arcs with bounds on capacity, the number of source nodes, the total number of nodes m and the total number of arcs n, but the scheme by which the nodes are interconnected, the cost assigned to each arc, and the distribution of supply and demand are completely random. NETGEN has given network flow research a standard set of test problems, but whether these random test problems accurately reflect the performance of competitive codes of real problems is questionable.

The original NETGEN paper acknowledges that problem structure influences the effectiveness of different algorithms. Bradley, Brown and Graves [Ref. 2] specifically state that their code, GNET, "solves real network models faster than random NETGEN problems of nominally comparable size and structure, suggesting that much remains to be learned from further investigation of special problem structures." More recently, in discussing shortest path problems, a special case of minimum cost network flow

problems, Dial et. al. [Ref. 4] indicates that "the most efficient solution procedure depends on the topology of the network and the range of the arc length coefficient." The purpose of this thesis is to explore the range of effects that certain types of network structure have on speed of network solution codes, and to develop a network generator which allows researchers to exercise their algorithms and codes on networks exhibiting different types of structure.

Real-world networks contain various types of structure which are readily apparent. For example, distribution networks sometimes exhibit a geographic echelon structure which has special topologies and dependencies among the flow costs. A hypothetical example of this might be a distribution system in which a commodity enters the continental U.S.A. through ports on the west coast. From the warehouses surrounding the ports of entry, the good is shipped to retail outlets in various regions across the country, but to reach the furthest region, the good must be shipped via distribution centers in intermediate regions (echelons). Assuming the cost of shipping is proportional to the distance over which it must be shipped, the distribution of costs inherits structure from the distribution of destinations.

Structure in a network model may also exist as a result of "gravity" modeled in traffic engineering [Ref. 16]. The

gravity model can determine with surprising accuracy the amount of rail traffic, commercial trucking, or even usage of the telecommunications networks between two cities. The gravity model is:

$$\frac{\text{(Population of city A) x (Population of city B)}}{\text{Distance between cities A and B}} \propto \text{Amount of trade between cities A and B}$$

Although this model represents a phenomenon of industry rather than a well-defined physical law, it does imply a certain amount of structure in industrial network flows.

Given that real-world problems are more accurately represented by test problems with simple structural assumptions, exploratory research needs to be done to ascertain which types of structure affect solution efficiencies and in what manner. The author chose to investigate patterns in cost, topology, and geographic echelons in this thesis. The number of variations on network structure is limited only by the imagination, so for this study, several representative selections are made. The intent here is not to exhaust all possibilities, but to gain insight into the design of test problems which may more accurately reflect performance of network flow algorithms on real problems.

If these new networks exhibit advantages over random test networks, then a decision must be made as to whether a new set of standard problems should be distributed or whether the generator code itself should be distributed, allowing researchers to produce their own networks. The advantages to a single set of test problems are obvious. A standard group of benchmarks encourages comparisons on identical networks and gives users reliable reference points. Additionally, transferring data sets on magnetic tape rather than codes essentially eliminates problems of machine independence and guarantees reproducibility of results. However, providing a comprehensive set of problems for the countless types of structure is impossible. Passing of a generator to fellow researchers would allow them to tailor test data to algorithms which may be written with specific networks in mind. This article presumes that until algorithm developers decide on what a representative group of structures is, it is better to allow researchers to select their own network structures.

B. COMPUTATIONAL METHODOLOGY

1. The Bounded Variable Primal Simplex Algorithm for Networks

The minimum cost network flow problem may be viewed as a specialized linear program.

$$\begin{aligned} &\text{Minimize} && cx \\ &\text{subject to} && Ax = b \\ & && 0 \leq x \leq u \end{aligned}$$

where A is a node arc incidence matrix with exactly one +1 and one -1 in each column.

Several methods are presently available for the solution of large minimum cost network flow problems including the primal simplex, dual simplex, primal-dual, out-of-kilter, and more [Refs. 2, 5, 13]. One of the most efficient primal network simplex algorithms is developed by Bradley, Brown and Graves in their GNET code [Ref. 2]. Because of its high speed and reliability, this code shall be used to compare the solution times of the structured networks produced in this thesis and the random networks created by NETGEN. Summarizing from the GNET paper, the primal simplex methods solves the linear programming problem in the following manner.

Manipulation of the matrix A (with addition of unit vectors, representing artificial variables) may allow A to be partitioned into A = (B,N), where B is an m x m matrix of linearly independent columns called a basis. Given a basis B, there will exist a unique $\hat{x}$ such that

$$B\hat{x} = b \qquad (1)$$

If $\hat{x} \geq 0$, a basic feasible solution to the original problem is $x^0 = \begin{pmatrix} \hat{x} \\ 0 \end{pmatrix}$. Assume such a solution is known. Partitioning c in the same manner as A results in

$$cx^0 = (c_B, c_N) \begin{pmatrix} \hat{x} \\ 0 \end{pmatrix} = c_B \hat{x} \tag{2}$$

A solution which satisfies the constraints of the original problem may be written

$$Ax = (B,N) \begin{pmatrix} x_B \\ x_N \end{pmatrix} = Bx_B + Nx_N = b \tag{3}$$

Since B is a basis, there exists a transformation Z such that

$$BZ = N \tag{4}$$

Algebraic manipulation of equations (3) and (4) yields

$$B(\hat{x} - Zx_N) + Nx_N = b \tag{5}$$

The general solution to equation (5) is then

$$x = \begin{pmatrix} x_B \\ x_N \end{pmatrix} = \begin{pmatrix} \hat{x} - Zx_N \\ x_N \end{pmatrix} \tag{6}$$

In this form the value of x is easily compared to the current solution x^0 and improved solutions are readily

identified when they exist. From equation (8) and the original objective function

$$cx = c_B\hat{x} + (c_N - uN)\, x_N \tag{7}$$

where u, called the "dual variables" or "simplex multipliers," is the solution of

$$uB = c_B \tag{8}$$

From (7) and the constraint $x_N = 0$, a necessary condition for an improved solution is that there exist a column of N, N^K such that

$$c_K - uN^K < 0 \tag{9}$$

Given there is at least one column in N corresponding to the variable x_K, a candidate variable chosen from all those satisfying (9) for entry into the basis, and a basic variable is selected to exit the basis by way of the ratio test [Ref. 2]. Having selected the variable for entry into the basis, an algebraic process called pivoting is performed to exchange the columns corresponding to the entering and exiting variable. If inequality (9) is not satisfied by a non-basic variable, then optimality has been achieved.

Otherwise, the search for an improved solution continues and another iteration is performed.

The network specialization of the primal simplex algorithm exploits the fact that all bases from A can be put into upper triangular form and represented very compactly. In the standard simplex algorithm $BZ^k = N^k$ in (4) is solved for Z^k by computing $Z^k = B^{-1}N^k$, and $uB = c_B$ in (8) is solved for u by computing $u = c_B B^{-1}$. This requires the expensive storage and updating of B^{-1} at each iteration. In contrast, with B in upper triangular form, $BZ^k = N^k$ and $uB = c_B$ can be solved directly via back substitution and forward substitution, respectively. The representation of B in 0(m) space, which is used instead of a full m x m matrix, speeds these solutions in addition to being space-efficient. Further advantages of the network simplex method are that all-integer arithmetic can be used if c and b are integer, and that specialized network data structures allow very efficient pivoting, retriangulation of the basis and updating of solutions.

2. Procedure for Comparison

The comparison used to evaluate the effects of various types of structure on solution of the minimum cost flow problem is the time required to achieve optimality. Although the number of pivots required to achieve optimality could also be used to compare these effects, the reliability

and consistency of that measure is highly suspect. All pivots are not of equal difficulty (difficulty defined as the number of machine operations required to complete a pivot in a computer implementation). It is possible for a solution which requires many pivots to be obtained in a shorter amount of time. (For example, see Goldfarb and Reid [Ref. 6] and their experiments with the "steepest-edge" variant of the primal simplex algorithm applied to general linear programs.) Time costs the computer-user money and is therefore a pertinent measure and is the measure which will be used for comparisons in this thesis. Unfortunately, time comparisons are difficult on a virtual memory machine like the IBM 3033AP since there may be significant variation in run times for identical problems. To minimize this effect, the time comparison is accomplished by solving each test network five times and noting the mean of the five solution times. The code used to solve the test problems in all cases was GNET. Solution by a single code permits accurate comparison of the solution times with regard to the influence of structure, but is not necessarily indicative of the performance of all algorithms on such problems. Future analysis should include other algorithms to reveal the influence of structure on those methods as well as the primal simplex method.

The test networks generated ranged in size from 200 to 4,000 nodes and from 1300 to 20,000 arcs. For each size, NETGEN was used to produce a random network version and each of two structured network generators was used to construct problems of various structural types. The first generator produces networks structured with respect to cost only, but the second generator yields networks with a variety of structure including structured supply, demand, cost, capacity and topology.

C. THE INFLUENCE OF BIG M ON SOLUTION TIME

A structured network generator is a tool for evaluating minimum cost network flow solution techniques. Other network research has indicated Big M [Ref. 1] is a parameter of the solution technique which can affect solution time [Ref. 7]. The version of GNET used in this research utilizes the Big M variant of the primal simplex method. In Chapter IV, an experiment evaluating the effect of Big M on solution times of network flow problems is performed which compares networks generated by the structured generators and by NETGEN. To facilitate discussion of Big M, the network linear programming problem shall be defined as before:

$$
\begin{array}{lll}
\text{(P)} & \text{Minimize} & cx \\
 & \text{Subject to} & Ax = b \\
 & & x \geq 0
\end{array}
$$

An initial basis matrix is required to initiate the simplex solution method used in existing network algorithms. If such a matrix is not readily apparent in the A matrix, an artificial vector, x_a, is introduced to give a convenient starting point for the simplex method. When an artificial vector is introduced, the initial basic feasible solution is given by $x_a = b$ and $x = 0$. Modification of the constraints requires modifying the objective function to reflect large penalties for non-zero values of the artificial variables in the problem solution. The new problem produced by these changes is as follows:

$$
\begin{array}{lll}
P(M) & \text{Minimize} & cx + Mx_a \\
 & \text{Subject to} & Ax + Ix_a = b \\
 & & x, x_a \geq 0
\end{array}
$$

where M is a very large number representing the penalties.

In the network simplex method, these penalties are assigned only to variables associated with flow into sink nodes. Even though x_a is a feasible solution to P(M), the design of the simplex method will force the artificial variables to zero in a search for the optimal solution to P, is such a solution exists [Ref. 1].

The disadvantage of the Big M method lies in attempting to select a value for Big M a priori. Too large a value will dominate the other cost coefficients in the objective function and may result in serious round-off errors in a computer or, in the case of the network simplex algorithm, problems with representation of large integers. However, too small a value will not force all the artificial variables to zero. In searching for the appropriate value for Big M, one must also be aware of the time lost to locating exactly the right value. It is believed that a close upper bound on the minimum acceptable Big M will be sufficient to reduce the CPU time required to run the simplex method without wasting time fine-tuning the estimate of Big M. In Chapter IV, a bound in bipartite networks based on the minimum and maximum cost arcs leaving the sink nodes is developed for this purpose.

An alternative approach to investigating the effects of Big M is also used. A dual formulation of P(M) is as follows:

$$\text{Maximize} \quad b^T u^T$$

$$\text{Subject to} \quad A^T u^T \leq c$$

$$I \; u^T \leq M$$

The second set of constraints implies:

$$u \leq M$$

i.e., M is an upper bound on the dual variables.

From this it is also true that if the optimal solution is known, then one can say that the $\max_i u_i$ would have been an excellent estimate for Big M. Although in Chapter IV this research attempts to determine and use a sharper bound on Big M prior to finding the solution, an analysis of M = $\max_i u_i$ dual variable in previously solved problems, may produce insight into possible differences in solution-time behavior of structured versus unstructured problems, and might lead to better estimates for Big M in future problems.

Several structured and unstructured networks of comparable sizes were tested to examine the change in solution time resulting from changes in Big M values. In each instance, the test problems were first solved using the maximum dual value previously obtained, and then solved numerous times with incrementally larger values of Big M until solution time did not change any further. The results of this investigation are included in Chapter IV.

II. NETWORKS WITH STRUCTURED COSTS

A. PHILOSOPHY

The most basic approach to the problem of generating a structured network is to create a network exhibiting structure in a single aspect, e.g., a network with random costs, random capacities, random supplies and demands, but structured topology. In this way, changes in solution efficiencies can be investigated with respect to changes in a single, isolated type of structure. Thus, a simple scheme to generate "singly" structured networks might be to take the feasible but random networks produced by NETGEN, and modify these networks to exclusively structure costs, or supplies and demands, or capacities, or topology.

NETGEN usually produces feasible networks, which is desireable, but it produces random costs, capacities and topologies. This chapter details initial attempts to produce networks structured in a single aspect. However, due to the generation methodology utilized in NETGEN, this is not easily done except with respect to costs. Consequently, using cost as the single structural aspect, arc flow costs are structured to simulate those costs which might occur in a physical distribution network. In such a system, arc flow cost is often a function of the distance

between the arc's head node, j and tail node, i [Ref. 12]. For k=1 and p=2, Euclidean distance becomes a special case of a function developed by Love and Morris [Ref. 12],

$$d_{ij} = k\,[(x_i - x_j)^p + (y_i - y_j)^p]^{1/p}$$

which an be used to estimate the actual road and shipping distances between two points. In this chapter, arc cost is then made a simple linear function of this distance since this seems to represent real-world structure in certain instances [Ref. 10].

A FORTRAN program, TRANS, was developed to take the arcs listed in the SHARE formatted output from NETGEN and replace arc costs with costs exhibiting the structure described above. The only user-defined input is the length to width ratio (r:1) of the rectangle into which the nodes are placed. TRANS assigns an (x,y) coordinate to each node generating y as a uniform (0,1) random deviate and x as a uniform (0,r) random deviate. (The uniform random number generator used for this purpose is the LRND portion of LLRANDOM II, a machine specific random number generator developed at the Naval Postgraduate School [Ref. 14]. Arc costs are then created by scaling the Euclidean distance between head and tail nodes to lie between 0 and the user-defined value, maximum cost. The output from TRANS is

identical to that from NETGEN with respect to node-arc connections arc capacities, and supplies and demands. Because the networks generated in this way are structured with respect to cost only, and other aspects remain random, these networks shall be referred to as "pseudo-structured" networks.

After the pseudo-structured networks are generated, GNET is used to solve each network five times and the mean solution time was recorded.

Three variations of the basic structure were produced. Nodes were randomly placed in a square and in rectangles with length to width ratios of 3:1 and 20:1 in an attempt to determine if any of the shapes and the resulting structures would significantly affect solution times. If any substantial change was observed, then further structure in that direction could be explored.

B. RESULTS

A representative sampling of the computational results for the pseudo-structured networks and NETGEN problems of similar size are contained in Table I. As evidenced by these values, the variation in the solution time ranged from 34% less time to 37% more time required to solve the TRANS networks than the NETGEN networks. In most cases solution time appeared comparable. The small and inconsistent changes observed here indicate that if structure does affect

TABLE I

Solution Times for Pseudo-Structured Networks

# sources	# sinks	# arcs	% capacitated	Total Supply	Average Solution Time (sec.)				Random Number Seed	
					NETGEN	TRANS			NETGEN	TRANS
						square	rectangle			
							3:1	20:1		
100	100	1300	0	100000	.18	.18	.22	.16	13502460	12345678
100	100	1300	100	100000	.18	.19	.23	.23	13502460	12345678
100	100	2000	0	100000	.24	.26	.23	.24	13502460	12345678
100	100	2000	100	100000	.17	.23	.28	.29	13502460	12345678
150	150	6300	0	150000	.93	.83	.83	.85	13502460	12345678
150	150	6300	100	150000	.67	.84	.89	.90	13502460	12345678
50	350	3000	0	20000	.51	.45	.46	.55	12345678	12345678
50	350	3000	20	20000	.53	.54	.51	.59	12345678	12345678
50	350	3000	100	20000	.58	.58	.56	.64	12345678	12345678
50	350	10000	0	20000	1.37	.90	1.16	1.19	12345678	12345678
50	350	10000	20	20000	1.02	.94	1.05	1.08	12345678	12345678
50	350	10000	100	20000	1.27	1.08	1.14	1.17	12345678	12345678
100	900	5000	0	100000	1.90	1.91	1.92	1.99	12345678	12345678
100	900	5000	20	100000	1.89	1.82	1.98	2.11	12345678	12345678
100	900	5000	100	100000	2.03	1.90	1.99	1.98	12345678	12345678
100	900	10000	0	100000	3.03	2.76	2.80	2.76	12345678	12345678
100	900	10000	20	100000	2.54	2.53	2.68	2.71	12345678	12345678
100	900	10000	100	100000	2.41	2.61	2.56	2.95	12345678	12345678

solution times of network algorithms significantly, the cost structure utilized by TRANS is inadequate to exhibit this, or the reduction in solution time is not achieved through cost structure alone.

III. A STRUCTURED NETWORK GENERATOR

A. INTRODUCTION

Pseudo-structured costs alone do not reveal any advantage to structured test problems over randomized networks. However, by providing networks with more profound and complex structure, which more closely approximates structure found in real-world networks, a reduction in solution times may be accomplished. Characteristics which can be structured include supply and demand, arc flow capacities, echelon structures of the nodes, and a wide range of indegrees (number of incoming arcs) and outdegrees (number of outgoing arcs) for individual nodes. This chapter addresses the development of a completely new network generator which provides various attributes of structure to the feasible (or infeasible, if desired) transportation and multi-echelon test problems which it creates.

Ideally, a structured network generator would provide the user with the ability to choose any one of several alternatives for the amount and type of structure in the desired network. These alternatives might include the following:

(a) type of network (transportation, assignment, multi-echelon)

(b) number of capacitated arcs

(c) tightness of capacity constraints

(d) number of node echelons

(e) number of nodes

(f) types of nodes (source, sink, transshipment source, transshipment sink, pure transshipment)

(g) amount of supply and demand

(h) choice of distributions with which supply, demand, and costs are allocated.

Generation of test problems may be a major expense in the testing of minimum cost network flow solvers. For instance, NETGEN problems can take about five times more computer time to generate than to solve with GNET [Ref. 2]. Thus, another important design criteria for this generator is efficiency in regard to computer time and storage requirements. Random number generation can be extremely time-consuming. The process of constructing test networks requires generation of random numbers, which can be very time consuming. Therefore, it is imperative that efficient methodology be used in random number generation. In this vein, it is better (faster) to create the random numbers in large groups and store them in an array rather than to call a random number subroutine each time another number is required. (This is the technique used in NETGEN.) The very nature of large industrial networks implies that test problems designed to simulate such networks will require

considerable computer storage. However, it is not necessary to store all the information generated. The arcs can be written directly to data files eliminating the need for arc-length arrays. Additional savings can be obtained by using arrays for several purposes rather than creating new arrays for each new requirement. A good example of this is reusing the arrays in which the random numbers are stored.

Another factor affecting time usage is the number and types of operations performed. A large portion of the generation time in network programming would be contained in determining flow patterns. A possible scheme for simulating network flow would be to generate a distribution to determine the likelihood and amount of flow between nodes. These distributions could be based on node attributes such as size, location, and many others. However, empirically generating such distributions by examining all possible pairs of nodes implies performing $O(m^2)$ operations. To maintain the generator's efficiency, a compromise is required between achieving real-world structure and generation speed. Performing $O(m^2)$ operations is extremely time-consuming and may require many orders of magnitude more operations than a method which requires $O(n)$ operations, n being the number of arcs. A more efficient method would be to pick head and tail node by some other method, even randomly (a loss of some structure seems unavoidable), and

determine flow based on that choice. The number of required operations is reduced to O(n) in this way.

It would be impossible to provide a network generator that could produce every possible type of structure. A more realistic approach is to build a framework from which researchers can develop test problems which meet specific needs. The framework needs to be general enough to be able to readily accept user supplied subroutines for structure beyond the capability of the basic generator. VSGEN is one implementation of a framework that meets the requirements set forth here. It is a FORTRAN program designed to efficiently create structured transportation and multi-echelon networks. Storage and time requirements are considered in the development, and the procedure used easily allow expansion.

1. Detailed Implementation

The methodology used in VSGEN is uncomplicated but effective. The program generates structure through the simulation of some real-world phenomena. Each node is assigned a set of attributes which include their rank, and a population based on that rank and location. The method for assigning node location developed in TRANS is also used here. Populations are assigned by using a phenomenon known as Zipf's Law [Ref. 16].

$$(\text{population}) \times (\text{rank}) \propto (\text{constant}).$$

Since the node with the largest population will be ranked one, this relationship indicates that the constant is approximately equal to the maximum population over all node populations. The maximum population used in VSGEN is min {100 x m, 100,000}. The ceiling of 100,000 is utilized to prevent difficulties with large integer arithmetic on the computer and the associated storage problems. Each node's population is defined by a random variable which is normally distributed about a mean of maximum population divided by node rank. The standard deviation associated with each population is assumed to be one-tenth of the node population. The LNORM portion of the LLRANDOM II random number generator [Ref. 14] was used to produce the necessary normal random variables.

The total supply is then distributed among the supply (source) nodes based on the population at each source. This pattern for supply allocation is used because it is reasonably assumed that nodes with larger populations have larger supplies in real-world networks. The portion of supply at source i is determined by

$$\text{supply at source } i = \frac{(\text{total supply})(\text{population at source } i)}{\text{total of source node populations}}$$

After supply has been allocated, the program uses the following methodology to build networks. Multi-echelon

(more than two echelons) networks are created by concatenating transportation networks, i.e., two-echelon networks together. Thus the transportation network subroutine is called (k-1) times by the multi-echelon routine to create k-echelon networks.

B. TRANSPORTATION NETWORKS

The transportation (two-echelon) algorithm proceeds by first determining the outdegree for a given source node in proportion to (1) the total number of arcs, and (2) the square root of the supply at that source. The relationship used is based on

$$\text{outdegree at source } i = \frac{(\text{total no. of arcs}) \times (\text{supply at source } i)^{1/2}}{\text{total supply}}$$

The reason for the use of square root of supply at source i is speculation by the author. If the outdegree is made directly proportional to the supplies, then in smaller problems the nodes with small supplies are occasionally assigned an outdegree of only one or two. This seems unlikely in the structure of the real world. Making outdegree a function of the square root of the supplies effectively reduces the severely skewed nature of the distribution of arcs and produces more intuitively appealing results. Additional constraints on the number of arcs

originating at any source prevent the outdegree from being less than one or greater than the number of sinks. Having at least one arc coming from each source helps insure feasibility; precluding the number of arcs from being greater than the number of sinks helps reduce the number of parallel arcs, i.e., the number of arcs with the same head and tail nodes.

After the number of arcs emanating from a source is determined, each of those arcs is randomly assigned a sink by choosing a number from the discrete uniform distribution on $(1,m_2)$ where m_2 is the number of sinks. As each destination is chosen, the arc cost is determined as a function of the Euclidean distance between source and sink node. The results of using the Euclidean distance did not reveal any gain in solution times over uniformly distributed costs. In an attempt to determine if other cost structures would produce reductions in solution times, cost was made proportional to the square root of Euclidean distance to simulate a distribution system with decreasing marginal cost per mile.

The gravity model discussed in Chapter I is utilized myopically in determining the additional demand to be placed at the newly chosen sink node. The population at the source and sink are multiplied, and the product is divided by the Euclidean distance between the nodes. This

value and the total supply at the current source are then combined to define the proportion of flow merited in the current arc. The methodology used here avoids the $O(m^2)$ operations which would be required to empirically determine a flow distribution as previously discussed in Section III-A. At this point, only arc capacity remains to be defined.

Arc capacity is determined by multiplying a user-defined value by the amount of flow just obtained. This feature allows the user to control the "tightness" of the arc capacities, and consequently, to explore the effect of capacity upon solution time. For generation of feasible networks, the only requirement is that the capacity multiplier must be greater than or equal to one. Values less than one will result in infeasible networks because arc capacities will not allow enough flow to satisfy demand. Actually, the ability to create networks with varying degrees of infeasibility is a useful property of VSGEN. Infeasible problems are not uncommon in practice and the testing of new solution codes should include infeasible problems.

Arc costs and capacities are created in the above manner until all supply has been allocated. Integer truncations occasionally prevent a small percentage of the requested number of arcs from being generated, but in all cases the total supply is completely distributed. Accurate monitoring

of the amount of supply remaining also precludes the generator from allocating more than the total amount of supply.

C. MULTI-ECHELON NETWORKS

Individual echelons in the multi-echelon networks are generated in the same manner as those in the two echelon networks. Each node is assigned population attributes exactly as before. Arc costs and capacities are assigned using the previous methodology also. The difference between producing two echelon and multi-echelon networks occurs in generating the location of each node and in the number of arcs between each echelon of nodes.

In this research, location attributes are assigned using two separate procedures. The first method assigns positions to all nodes inside one rectangle, regardless of echelon, just as TRANS did in Chapter II. The second method for assigning locations allows the researcher to evaluate any reduction in solution times available through a geographic echelon structure. The total area over which the network is defined remains unchanged. Source nodes are located at one end of the rectangle and sink nodes at the opposite end. Those nodes account for two of the requested echelons in each problem. The other nodes (transshipment nodes) are assigned to regions in the interior of the rectangle between sources and sinks. The region size is equal for all

echelons including those of supply and demand, and consequently, the width of a region is inversely proportional to the number of echelons requested by the user. The height of each region remains constant for all cases.

Flow progresses through the network from one echelon to the next with flow permitted only between adjacent echelons. Each unit of flow must transit through all echelons sequentially; i.e., no echelon may be bypassed and flow does not backtrack into previously transitted echelons. The feature is patently different from the transshipment procedure utilized in NETGEN. Although NETGEN allows the user to request transshipment nodes, the generator does not treat those nodes as belonging to a set of one or more echelons. In proceeding from a pure source to a pure sink node, flow may pass through transshipment nodes, but it is not required to do so. The totally random nature of NETGEN allows flow on arcs between any two nodes except between two pure sources or two pure sinks. Using such a scheme for network generation precludes analysis of geographic echelon structure.

The manner in which flow is directed through each network generated by VSGEN is simple. Although the number of nodes in each echelon is user-defined, the number of arcs between echelons is not. For simplicity, that number is assumed to be proportional to the product of the number of

nodes in the two adjacent echelons, and the total number of arcs to be generated. The generator then utilizes the number of arcs between echelons as input to the subroutine used to create a two echelon network. All flow in the current echelon is passed to the next echelon before any flow is passed on to subsequent echelons. The current and immediately subsequent echelons are treated as a two echelon network unto themselves; the current echelon being the supply nodes and the subsequent echelon being the demand nodes. When the flow between those two regions is complete, the subsequent echelon is then designated as the supply echelon and its successor is designated the demand echelon. This process is continued until flow has passed completely through the network to the true demand nodes.

D. OUTLINE OF VSGEN

This section specifies the required input to VSGEN and outlines the VSGEN algorithm. The ability to structure the test networks in several aspects results in slightly more complicated input compared to NETGEN. Table II shows the required input for VSGEN.

The output format utilized is the SHARE format, the same as that produced by NETGEN, because this is probably the most widely used network format [Refs. 3, 5, 11, and 15].

VSGEN, the algorithm for creating structured transportation and multi-echelon networks is outlined as follows:

TABLE II

VSGEN Input Specifications

Variable Name	Description	Value	Data Type	Card Number	Card Columns	Remarks
ITYPE	Assignment Problem Transportation Problem General Transshipment Problem Multi-echelon Problems 3-9 Echelons	1 2 3 13-19	Integer	1	1-2	Available for Future Research Two Echelon Networks Available for Future Research
ICAP	Capacitated or Uncapacitated	0,1	Integer	1	3-4	0 = Uncapacitated 1 = Capacitated
LECHLN (I)	Number of Nodes in Echelon I	1-99999	Integer	2	1-45	Not Used in Two Echelon Problems
NNOD	Total Number of Nodes	1-999999	Integer	1	6-15	
NARC	Total Number of Arcs	1-9999999	Integer	1	16-25	
NPSRC	Number of Pure Sources	1-999999	Integer	2	1-5	Used for Two Echelon Networks Only
NPSNK	Number of Pure Sinks	1-999999	Integer	2	6-10	Used for Two Echelon Networks Only
ITSUP	Total Supply	1-9999999999	Integer	3	1-10	
MINCAP	Minimum Capacity	1-9999999999	Integer	4	1-10	Available for Future Research
MAXCAP	Maximum Capacity	1-9999999999	Integer	4	11-20	Available for Future Research
CAPMUL	Capacity Multiplier	0.0-9999999.99	Real	4	21-30	
MAXCST	Maximum Cost	1-99999999	Integer	5	11-20	
MINCST	Minimum Cost	1-MAXCST	Integer	5	1-10	

VSGEN

Input: NNOD = Total number of nodes
NARC = Total number of arcs
ITSUP = Total supply
MAXCST = Maximum allowable cost on any arc
MINCST = Minimum allowable cost on any arc
ITYPE = Type of network; 2 = transportation
13-19 = multiechelon (3-9 echelons)
LECHELN(e), e=1,...,NECHELN,= Number of nodes in each echelon e
CAPMUL = Capacity multiplier, CAPMUL > 0
ICAP = Capacitated network indicator; 0 = uncapacitated
1 = capacitated

Output: Feasible network in SHARE format if CAPMUL $\geq$ 1 or ICAP = 0, else an infeasible network in SHARE format.

(1) Read input.

Define N = set of nodes, i = 1,...,NNOD .

N_e = set of LECHELN(e) nodes in echelon e, e=1,...,NECHELN

RATIO = Ratio of x to y in rectangle containing nodes

If ITYPE $\geq$ 13, NECHELN = ITYPE - 10

Otherwise, NECHELN = 2.

(2) Assign node attributes

(a) (i) If a geographic echelon structure is used, then for each echelon e, and for each node $i \in N_e$, randomly assign coordinates X(i) and Y(i) in rectangle bounded by the coordinates

$$\left(\left[(e-1) \times \frac{\text{RATIO}}{\text{NECHELN}}\right], 0\right), \quad \left(\left[(e-1) \times \frac{\text{RATIO}}{\text{NECHELN}}\right], 1\right)$$

$$\left(\left[e \times \frac{\text{RATIO}}{\text{NECHELN}}\right], 0\right), \quad \left(\left[e \times \frac{\text{RATIO}}{\text{NECHELN}}\right], 1\right)$$

and let MAXDIST = $\left[\left(2 \times \frac{\text{RATIO}}{\text{NECHELN}}\right)^2 + 1^2\right]^{1/2}$

(ii) Otherwise, for each node ieN, randomly assign coordinates X(i) and Y(i) in rectangle bounded by the coordinates

(0,0), (0,1), (RATIO,0), (RATIO, 1)

and let MAXDIST = $[\text{RATIO}^2 + 1^2]^{1/2}$

(b) For each node iϵN, randomly assign node rank RANK(i).

(c) Let MAXPOP = min{100xNNOD, 10^5} be the maximum node population.

(d) For each iϵN, randomly assign node population POP(I) using normal distribution having mean MAXPOP/RANK(i) and standard deviation 0.1xMAXPOP/RANK(i), but truncated below 1.

(3) Distribute total supply over all source nodes,

(a) Let TOTPOP = $\sum_{i \epsilon N_1}$ POP(i)

(b) For each iϵN_1, let ISUP(i) = ITSUPxPOP(i)/TOTPOP

(c) For each iϵN-N_1, let ISUP(i) = 0.

(4) For each node iϵN_1, write out source node information in SHARE format, I and ISUP(i).

(5) Determine the number of arcs to be created between echelons.

(a) If ITYPE=2, let NARC(1)=NARC and go to ( ).

(b) Let NTARC = $\sum_{e=1}^{\text{NECHELN}-1}$ LECHELN(e)xLECHELN(e+1)

(c) For e=1 to NECHELN-1, let NARC(e) = NARCxLECHELN(e)xLECHELN(e+1)/NTARC.

(6) e=1

(7) Let ISQSUP = $\sum_{i \in N_e} ISUP(i)^{1/2}$

(8) For each node $i \in N_e$

(a) Let OD = NARC(e)xISUP(i)/ISQSUP be the outdegree of node i.

(b) Randomly choose from N_{e+1} a set of OD tail nodes T_i for arcs emanating from i.

(c) For each node $j \in T_i$

(i) Let DIST = $[(X(i)-X(j))^2+Y(i)-Y(j))^2]^{1/2}$ be the distance from i to j.

(ii) Let FLOW(j) = POP(i)xPOP(j)/DIST = proportion of total flow from i going to j.

(iii) Let COST(j) = max{MINSCT,[MAXCSTx(DIST/MAXDIST)$^{1/2}$]} be the cost assigned to arc i,j.

(d) Let TOTFLOW = $\sum_{j \in T_i} FLOW(j)$.

(e) For each node $j \in T_i$,

(i) Let IASSGN = FLOW(j)xISUP(i)/TOTFLOW be the amount of flow to be assigned to node j from node i.

(ii) Let ISUP(j) = ISUP(j)+IASSGN be the current total amount of flow assigned to node j.

(iii) If ICAP=0, CAP=ITSUP, else CAP=IASSGNxCAPMUL.

(iv) Write out arc information in SHARE format, i, j, COST(j), and CAP.

(9) If e < NECHELN-1, let e=e+1 and go to (7)

(10) For each node $j \in N_{e+1}$, write out demand node information in SHARE format, j and ISUP(j).

End of VSGEN

E. RESULTS

Several types of structured networks were compared to NETGEN networks of the same size. Variations in ratio of supply and demand nodes, assignment of location attributes, number of nodes and arcs, and number of echelons were included in the evaluations. The ratio of supply to demand nodes ranged from severely skewed (few sources, many sinks) to equal numbers of sources and sinks. In no case were there more sources than sinks. The location attributes were assigned in two ways. One method assigned locations randomly inside a rectangle, similar to the methodology of TRANS. The second method assigned location according to node echelon, simulating the geographic structure described in Section II-C. The number of nodes and arcs ranged from 400 to 2,000 and 5,000 to 15,000, respectively. Two, three, and four echelon networks were evaluated. For each capacitated VSGEN network, a range of values for the capacity multiplier was tested and the values recorded in Table III reflect those versions which resulted in the fastest computation times. The capacity multiplier values ranged from 7.5 to 50.0. Those these may seem like large capacities, these values are small when compared to the capacities allowed on the NETGEN arcs which ranged between one one-hundredth and one-tenth of total supply.

TABLE III

Solution Times for Structured Networks

#nodes	#arcs	#echelons	Total Supply	%capacitated	VSGEN Capacity Multiplier	Node-Echelon Distribution		Generation Time (sec)			Average Solution Time (sec)		
						VSGEN	NETGEN	VSGEN(a)	VSGEN(b)	NETGEN	VSGEN(a)	VSGEN(b)	NETGEN
400	5000	2	20000	0	-	200x200	200x200	1.83	(c)	3.43	0.61	(c)	0.85
400	5000	2	20000	100	31.0	200x200	200x200	1.87	(c)	4.71	0.44	(c)	0.71
300	6300	3	150000	0	-	10x40x250	150x150	2.19	2.36	4.16	0.39	0.48	0.93
300	6300	3	150000	100	50.0	10x40x250	150x150	2.15	2.31	5.62	0.51	0.53	0.67
1000	5000	2	50000	0	-	500x500	500x500	2.07	(c)	5.06	1.54	(c)	1.52
1000	5000	2	50000	100	7.5	500x500	500x500	2.06	(c)	6.13	1.46	(c)	1.63
1000	10000	2	50000	0	-	500x500	500x500	3.85	(c)	8.58	2.03	(c)	2.78
1000	10000	2	50000	100	30.0	500x500	500x500	3.89	(c)	10.63	1.60	(c)	(d)
1000	5000	3	50000	0	-	300x300x400	500x500	1.94	1.98	5.06	1.61	1.57	1.52
1000	5000	3	50000	100	50.0	300x300x400	500x500	1.97	1.91	6.13	1.57	1.60	1.63
1000	10000	3	50000	0	-	300x300x400	500x500	3.55	3.51	8.58	2.15	2.20	2.28
1000	10000	3	50000	100	50.0	300x300x400	500x500	3.59	3.56	10.63	2.17	2.18	(d)
1000	5000	2	100000	0	-	100x900	100x900	2.05	(c)	5.65	1.62	(c)	1.90
1000	5000	2	100000	100	40.0	100x900	100x900	2.06	(c)	6.63	1.51	(c)	2.03
1000	5000	3	100000	0	-	10x90x900	100x900	1.98	2.00	5.65	1.60	1.73	1.90
1000	5000	3	100000	100	30.0	10x90x900	100x900	1.94	1.97	6.63	1.44	1.70	2.03
1000	10000	3	100000	0	-	10x90x900	100x900	3.58	3.63	9.74	1.87	1.83	3.03
1000	10000	3	100000	100	30.0	10x90x900	100x900	3.68	3.67	11.97	1.63	1.57	2.41
1000	5000	2	100000	0	-	200x800	200x800	2.05	(c)	5.41	1.74	(c)	1.94
100	5000	2	100000	100	8.5	200x800	200x800	2.10	(c)	6.43	1.75	(c)	2.00
100	5000	4	100000	0	-	10x40x150x800	100x900	1.90	1.92	5.65	1.56	1.63	1.90
1000	5000	4	100000	100	50.0	10x40x150x800	100x900	1.96	1.94	6.63	1.46	1.50	2.03
4000	15000	2	200000	0	-	2000x2000	2000x2000	6.61	(c)	22.10	17.50	(c)	(d)
4000	15000	2	200000	100	20.0	2000x2000	2000x2000	6.61	(c)	19.57	17.35	(c)	(d)

(a) TRANS "pseudo-structure" utilized for arc costs (nodes distributed in one large rectangle)
(b) Nodes assigned according to geographic echelons
(c) This size of network not generated with geographic echelon structure
(d) NETGEN network produced was infeasible

VSGEN revealed several interesting trends. The more efficient methodology used to produce random numbers achieved impressive reductions in the time to generate the test networks, as much as 78% less time to generate VSGEN problems than comparably sized NETGEN problems. As it should, the amount of time required by VSGEN appears to be directly proportional to the number of arcs requested. Table III shows the generation times for VSGEN and for NETGEN problems of comparable size. The "Node-Echelon Distribution" column in that table designates the number of nodes assigned to each echelon. The first number represents the number of nodes in the first echelon (sources), the last number represents the number of nodes in the final echelon (sinks) and any numbers in between represent the interior echelons (pure transshipment nodes). For example, the entry 10x40x250 indicates 10 sources, 40 transshipment nodes, and 250 sinks.

The time required to solve the structured networks was also consistently less than the random NETGEN networks. The reductions ranged from 1% to 59%; the mean was a 26% reduction in solution time. Comparisons between structured networks indicated that the ratio of sources to sinks also affects solution time. Consistently shorter times were evident for skewed networks, i.e., those with more sinks than sources. In fact, there seems to be a direct

relationship between the ratio of sources to sinks and average solution time. As the ratio decreases, the solution time also does. This relationship holds true for two, three, and four echelon networks. In the multi-echelon networks, there is no detectable difference between solution times for three or four echelon problems of approximately the same skewness. However, as with the transportation problems, the solution times were shorter for networks with fewer sources than transshipment nodes and sinks than for networks with an approximately equal number of nodes in each echelon. These comparisons are evident in Table III.

The results obtained also indicated that one of the most sensitive factors in determining solution time is arc flow capacity. Extremely tightly capacitated problems, those with a capacity multiplier close to one, can use as much as five times the amount of CPU time as the same network with uncapacitated arcs. In no case did a tightly capacitated problem solve more quickly than one with loose constraints. Figure 1 shows a sampling of capacity versus time relationships from the networks solved.

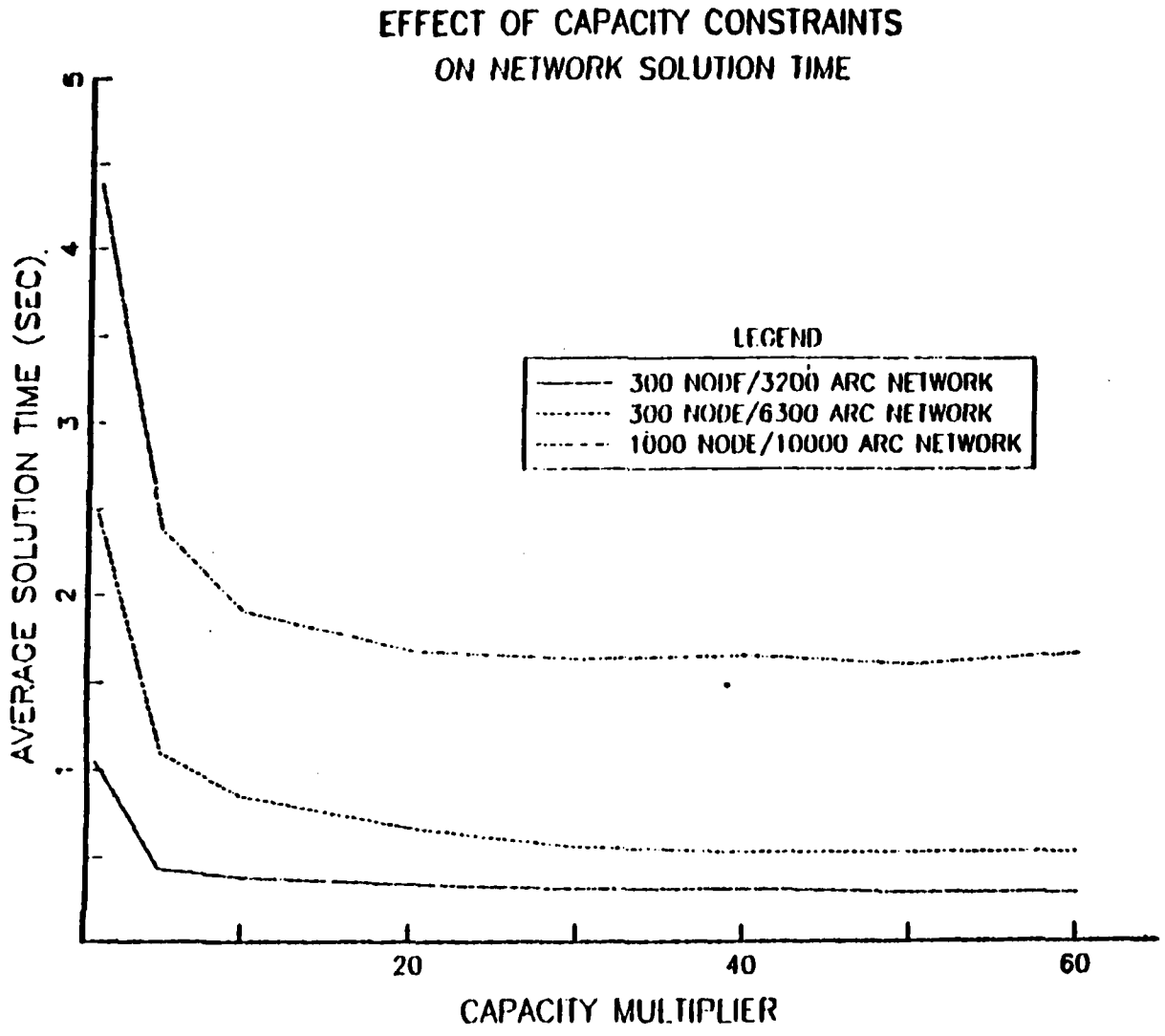


Figure 1. Effect of Capacity Constraints on Network Solution Time

IV. AN EXPERIMENT ON BIG M USING VSGEN

A. ANALYTICAL DEVELOPMENT

This chapter presents a use of VSGEN as a vehicle for comparing solution times in structured test networks and random test networks when Big M is experimentally allowed to vary. As discussed in the Introduction, small values of Big M in the primal simplex method may reduce the solution times of minimum cost network flow problems. Examining the effects of Big M presents an excellent opportunity for comparing structured and random networks. VSGEN and NETGEN are used in this chapter to generate test networks upon which the effects of varying Big M may be evaluated. A sampling of the "pseudo-structured" networks of Chapter II is also included in the evaluation.

Before generating the test networks, it is prudent to analytically examine Big M. Reductions in solution times resulting from substantially reducing the value for Big M have already been claimed by Gregoriadis [Ref. 7]. Too large a value for Big M can cause numerical difficulties. It is desirable, therefore, to find a value for Big M, M_u, which is as small as possible, yet which allows a feasible solution to be found if one exists. In this section, a bound on Big M in bipartite networks is derived which is not

computationally burdensome and this value is compared to the default value used in GNET. In the next section, the bound is also compared to the optimal value of Big M obtained by solving the minimum cost flow problem.

An $m_1 \times m_2$ bipartite network is a network with a set of m_1 source nodes S, and a set of m_2 sink nodes T, such that $m = m_1 + m_2$. Furthermore, all arcs are of the form (i,j) where $i \in S$ and $j \in T$. Transportation networks and assignment networks are examples of commonly occurring bipartite networks. Bipartite networks offer a relatively simple structure upon which to base initial calculations for the bound on Big M. Consequently, the bound developed in this section is directly applicable to bipartite networks only, but similar developments might extend the bound to more general networks.

Given that Big M must be an upper bound on the dual variables, and the duals represent the marginal cost for a change in flow to a given node, one can logically evaluate a worst case change in flow in a given network, i.e., the largest possible value for a dual variable. To determine the cost for an increase in demand of one unit at demand node j, one needs to understand the chaining effect that the increased demand might cause. For this development, a p x p bipartite network is assumed with $p = m_1 = m_2$, and it is further assumed that $c_{ij} \geq 0$ for all arcs (i,j). If demand

at node j_1 is increased by one unit, this extra unit might be supplied directly from supply node i_1 along arc (i_1,j_1). However, this may cause the reduction in flow by one unit along arc (i_1,j_2), $j_2 \neq j_1$, which in turns results in a deficit of one unit of flow at node j_2 which must be supplied from some node $i_2 \neq i_1$. This chaining effect may continue along a chain of arcs $(i_1,j_1),(i_1,j_2)\ldots(i_h,j_{h+1})$ resulting in a net increase in cost of

$$\sum_{k=1}^{h} c_{i_k j_k} - \sum_{k=1}^{h-1} c_{i_k j_{k+1}} .$$

An upper bound on the marginal cost associated with a chain using source nodes $i_1, i_2, \ldots, i_h$, in that order, is

$$\sum_{k=1}^{h} \max_{j \in T} c_{i_k j} - \sum_{k=1}^{h-1} \min_{j \in T} c_{i_k j}$$

Since we are concerned with the worst case, an upper bound on the cost associated with any chain using h sink nodes is

$$C_h = \max_{S_{h-1} \subseteq S, i_h \in S-S_{h-1}} \left\{ \sum_{i \in S_{h-1}+i_h} \max_{j \in T} c_{ij} - \sum_{i \in S_{h-1}} \min_{j \in T} c_{ij} \right\}$$

It follows that for $h < p$

$$C_h \leq C_{h+1}$$

and thus, C_p, is an upper bound on the maximum marginal cost of an increase of one unit of flow at any sink node. For computational reasons, we use an upper bound on C_p derived as follows:

$$C_p = \max_{S_{p-1} \subseteq S, i_p \in S-S_{p-1}} \left\{ \sum_{i \in S_{p-1}+i_p} \max_{j \in T} c_{ij} - \sum_{i \in S_{p-1}} \min_{j \in T} c_{ij} \right\}$$

$$= \sum_{i \in S} \max_{j \in T} c_{ij} + \max_{i_p \in S} \left\{ - \sum_{i \in S-i_p} \min_{j \in T} c_{ij} \right\}$$

$$\leq \sum_{i \in S} \max_{j \in T} c_{ij} - \sum_{i \in S} \min_{j \in T} c_{ij} + \max_{i \in S} \left\{ \min_{j \in T} c_{ij} \right\}$$

$$\equiv M_u.$$

If a feasible solution to a bipartite network flow problem exists, then M_u defined as above insures that the feasible solution will be found. M_u will be smaller than the default value for Big M used by GNET, $m \times \max_{(i,j)} c_{ij}$, since M_u will always be less than or equal to $1/2m \times \max_{(i,j)} c_{ij}$.

B. RESULTS

The sharper bound on Big M, M_u, was recorded for all random bipartite networks generated by NETGEN. Rarely was this estimate significantly more than a 50% reduction from the default value used in GNET. Similar results were

observed for the TRANS networks. For the structured bipartite networks generated by VSGEN, the bound value was as much as 70% reduction frorm GNET's default value. However, in large networks even a 70% reduction translates into a value which is quite large compared to maximum absolute cost. The values for Big M which Gregoriadis reports are necessary to reduce solution times are on the order of 1.5 x max c_{ij} to 5.0 x max c_{ij}. Sharper results might be obtained for an m_1 x m_2 bipartite network where $m_1 < m_2$, since the maximum length of the chains used in the derivation of M_u would be $2m_1 - 1$. Even those bounds would be several orders of magnitude greater than the values necessary, and so the bound was not fully developed for such networks. The sharper bound here does not appear to be useful for reducing computation times, but might be helpful in avoiding numerical problems associated with handling large integer values on a computer.

Although the bound, M_u, was not sharp enough to compare with Gregoriadis' claims, the maximum dual variable obtained using GNET was recorded for the test networks and used as Big M in an attempt to validate those claims. Big M was incrementally increased from this starting point until no further reduction in solution time was evident.

For all networks, regardless of structure, the reductions in solution times were insignificant over the

entire range of values of Big M for which feasibility was maintained, contrary to the Gregoriadis's observations. As the value for Big M increased slowly from its minimum, the solution time quickly increased to equal the time of the solution which utilized GNET's default value.

For the TRANS and NETGEN networks, using the maximum dual for Big M resulted in a reduction in solution times of only 1% to 9% over the solution time achieved using the default value of Big M. The results of reducing Big M in solving networks created by VSGEN were only slightly better (faster). In all cases, the reductions in solution time were less than 15% and the mean maximum reduction was 10%. Figure 2 illustrates the effects of varying Big M in solving random and pseudo-structured networks. Figure 3 illustrates the results for the structured networks generated by VSGEN. In both figures $M_u = \alpha \times \max c_{ij}$.

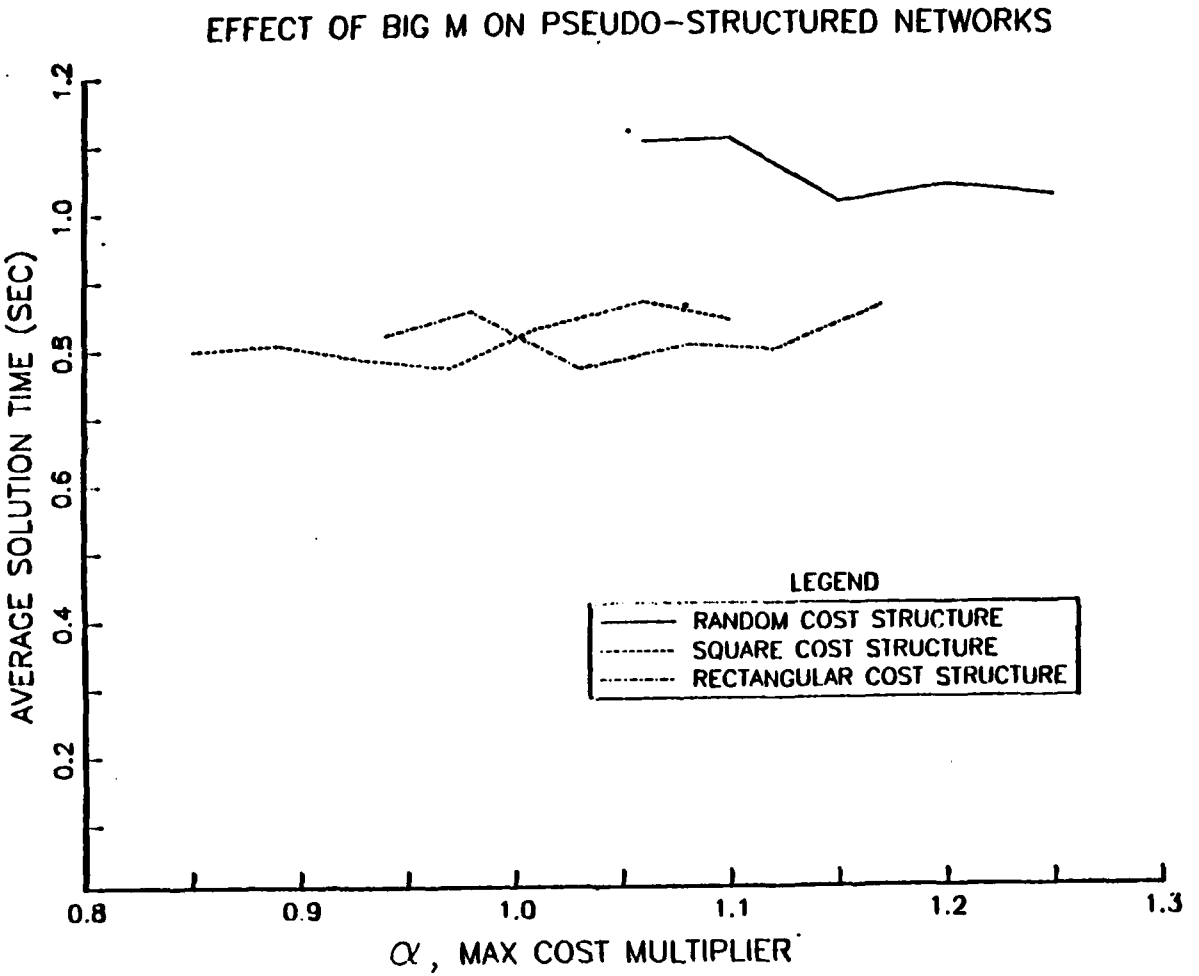


Figure 2. Effect of Big M on Pseudo-Structured Networks

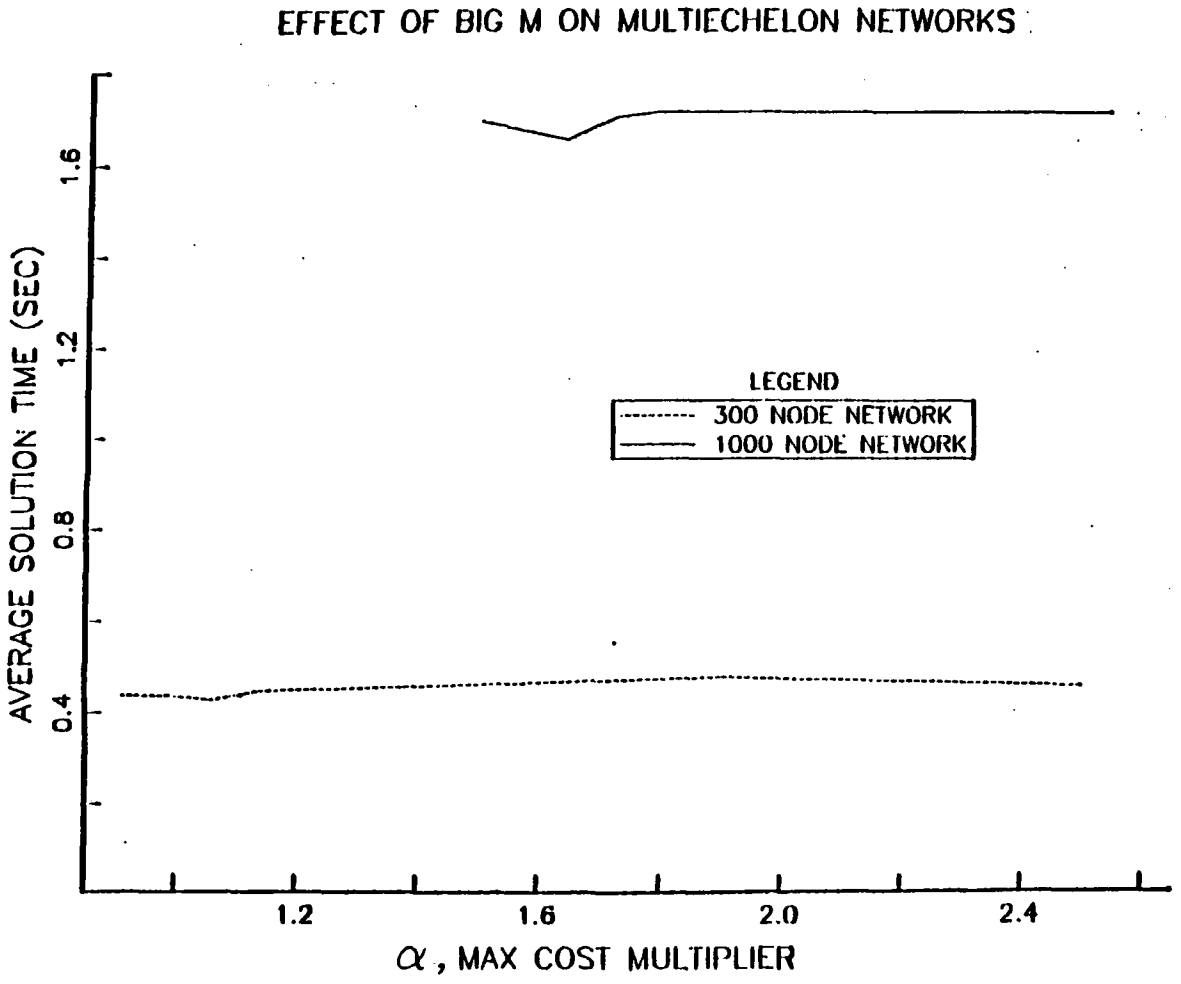


Figure 3. Effect of Big M on Multiechelon Networks

V. SUMMARY AND DIRECTIONS FOR FURTHER RESEARCH

A. SUMMARY

1. Methodology

The research presented here was initiated with the intent of determining the effect of structure on the time required to solve minimum cost network flow problems in relation to the solution time required for random networks of the same type and size. The type and complexity of structure required to cause changes in solution times were to be explored, as well as the type of structure to which solution time was most sensitive. To perform the analysis it was necessary to construct the framework for a structured network generator that was efficient and easily expandable to various structural specifications. Further, a sharper bound on Big M was desired in order to examine claims that lower values of Big M reduced network solution time significantly.

The framework for a structured network generator has been successfully created in VSGEN. The program forms feasible, structured transportation and multi-echelon test networks quickly and reliably. It does not yet have the capability to produce assignment or general transshipment problems. At its present stage of development, VSGEN allows

the user to apply structure to the arc flow costs, arc flow capacities, and node-echelon distribution. Further, the user controls total supply, total number of nodes, total number of nodes in each echelon of a multi-echelon problem, tightness of arc capacities, and maximum unit flow cost. The costs are structured in one of two ways. The first method assigns node locations randomly within a rectangle. The second method assigns nodes to sections of the rectangle in direct relation to the echelon number to which a node is assigned. In each case the cost for flow is proportional to the square root of the Euclidean distance between nodes. The amount of flow assigned is determined based on node attributes including location and node population. The population of each node is determined by application of Zipf's Law.

To test the hypothesis that structured networks solve more quickly than random networks generated by NETGEN, a wide variety of problems were solved by GNET, an efficient primal simplex code for minimum cost network flow problems. The mean of five solution times for each network was used in the time comparisons. The network parameters that were varied were total number of nodes and arcs, ratio of sources to sinks, node-echelon distribution, total supply, and arc capacity.

The sensitivity of network solution time to the parameter Big M was also tested by initially setting that parameter equal to the maximum dual variable at optimality. Big M was incremented upward from that value until no further change in solution time was evident. In addition, a bound on Big M sharper than the default value used in GNET, m x max c_{ij}, was developed.

2. Findings

VSGEN produces test networks faster than NETGEN. In some cases, generation of networks by NETGEN requires more than three times the amount of computer time required by VSGEN for comparable networks. Further, VSGEN consistently produces feasible networks when they are requested in contrast to NETGEN, which occasionally generates unrequested infeasible networks (negative demands). These infeasibilities can be avoided at times by changing the random number seed in NETGEN. A more reliable method for insuring feasibility is to input a total supply several orders of magnitude greater than the number of nodes.

VSGEN's only apparent difficulty is in generating exactly the requested number of arcs. Some instances require the user to inflate the input to obtain the desired number of arcs. The cause of this is a combination of the method of arc-to-node allocation and integer truncation. NETGEN does not exhibit this error and in most cases

produces a small percentage greater than the number of arcs requested.

The structured networks are consistently solved more quickly than random networks when capacities are not too restrictive. However, different types of structure affect solution times to varying degrees. The cost structure used in the TRANS program in Chapter II, defined as being in direct proportion to the distance between nodes, does not appear to have any significant effect on solution times. Structured transportation networks generated by VSGEN with an equal number of sources and sinks solve faster than random networks of the same size. This indicates that the combination of the different cost structure and methodology which assigns flow in VSGEN does result in reduced solution times. The individual contributions of cost and flow structure have not been determined. The most influential factors are node-echelon distribution and arc capacity. A severely skewed node-echelon distribution produces much shorter solution times than networks with equal numbers of nodes between the echelons. The addition of one or two more echelons to a skewed network did not reveal any significant change. Likewise, changing from random placement of nodes to geographic echelon structure in the multi-echelon problems did not result in reduced solution times.

The most sensitive parameter affecting solution time is arc capacity. Tightly capacitated networks may require more than five times as much time to solve as uncapacitated networks. Some capacitated problems do solve more quickly than the same networks uncapacitated, at certain levels of capacitation.

In contrast to the information presented by Gregoriadis, Big M did not significantly reduce network solution time. The largest reductions achieved averaged approximately 10%. Additionally, the sharper bound on Big M analytically determined in Chapter IV is not of any practical significance at the present time. The bound achieves gains between 50% and 70% over GNET's default value, which might be useful in avoiding numerical difficulties with large integers in computer storage, but this bound in still several orders of magnitude greater than the values necessary to influence solution times.

B. DIRECTIONS FOR FURTHER RESEARCH

VSGEN should be expanded to allow user selection of a wider variety of structures. Subroutines to handle assignment problems and other structures encountered in network research should be developed. Suggestions for additional subroutines include the following:

(1) subroutines which would allow the user to generate assignment and general transshipment problems.

(2) a structured distribution for determining which nodes may communicate or be connected. Although random selection of sink nodes is simple and efficient, it may be desirable to assign sinks to sources based on the relative location of the head and tail nodes and other node attributes as well. As an intermediate step in achieving such structure, one may wish to simply assign nodes to classes and allow communication between only specified classes. Assignment problems would be created effectively this way.

(3) a subroutine which would assign location by region instead of echelon; that is, divide the rectangular area in which the nodes have been placed, horizontally as well as vertically. Region could become an attribute on which to base the distribution discussed in (2). This would allow creation of general transshipment problems which are not bipartite.

(4) a subroutine that would allow replication of a network over multiple time periods and create inter-connections between these "temporal echelons" depending on the time required to travel between nodes.

(5) a subroutine that would allow the user to define the function of Euclidean distance from which unit flow cost is determined.

Other portions of this thesis warrant further effort, also. Some simple modifications are required to improve

NETGEN and VSGEN. The procedure for assigning outdegree in VSGEN should be refined so that the user may be assured of producing networks with the requested number of arcs. The random number generator in NETGEN should be replaced by updated versions to examine the possibility that the old random number subroutine is the cause of NETGEN's slower generation times. In any case, NETGEN's procedure should be revised to reflect the more efficient method of generating random numbers in groups rather than one at a time. Finally, and most significantly, NETGEN's method of allocating supply needs to be changed to insure generation of feasible networks. The infeasible problems produced without warning result in wasted time and effort.

The most important direction for further research is to broaden the base of comparison developed in this thesis. It is clear that structure affects solution time, but the contributions of the various aspects of structure are yet unclear. Future study should compare networks with isolated types of structure to reveal individual contributions. Different topologies, such as those found in inventory problems, should be explored. Although substantial demonstrations of the difference between structured and random networks have been presented, statistical significance in this research can be achieved only through extensive effort in varying network structure.

LIST OF REFERENCES

1. Bazaraa, M.S. and Jarvis, J.J., Linear Programming and Network Flows, pp. 154-163, John Wiley & Sons, Inc., 1977.

2. Bradley, G.H., Brown, G.G. and Graves, G.W., "Design and Implementation of Large Scale Primal Transshipment Algorithms," Management Science, Vol. 23, No. 1, p. 23, 1977.

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

4. Dial, R., Glover, F., Karney, D, and Klingman, D., "A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees," Networks, Vol. 9, pp. 215-248, 1979.

5. Fulkerson, D.R., "An Out-of-Kilter Method for Minimum Cost Flow Problems," SIAM Journal of Applied Mathematics, Vol. 9, No. 1, 1961.

6. Goldfarb, D. and Reid, J.K., "A Practical Steepest-Edge Simplex Algorithm," Mathematical Programming, Vol. 12, No. 3, pp. 361-371, June 1977.

7. Gregoriadis, M.D., "Minimum Cost Network Flows, Part I: An Implementation of the Network Simplex Method," Laboratory for Computer Science Research, Rutgers University, LCSR-TR-37, September 1982.

8. Hatch, R.S., "Bench Marks Comparing Transportation Codes Based on Primal Simplex and Primal-Dual Algorithms," Operations Research, Vol. 23, No. 6, pp. 1167-1172, 1975.

9. Helgason, R.V., and Kennington, J.L., Algorithms for Network Programming, pp. 255-256, John Wiley & Sons, Inc., 1980.

10. Institute of Traffic Engineers, Transportation and Traffic Engineering Handbook, pp. 559-598, Prentice-Hall, Inc., 1976.

11. Klingman, D., Napier, A. and Stutz, J., "NETGEN: A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," Management Science, Vol. 20, No. 5, pp. 814-821, January 1974.

12. Love, R.F., and Morris, J.G., "Modelling Inter-City Road Distances by Mathematical Functions," Operations Research Quarterly, Vol. 23, No. 1, 1972.

13. McGinnis, L.F., "Implementation and Testing of a Primal-Dual Algorithm for the Assignment Problem," Operations Research, Vol. 31, No. 2, 1983.

14. Naval Postgraduate School, Technical Report NPS55-81-005, The New Naval Postgraduate School Random Number Package LLRANDOMII, by P.A.W. Lewis and L. Uribe, February 1981.

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

16. Zipf, G.K., Human Behavior and the Principle of Least Effort, pp. 370-390, Hafner Publishing Co., Inc., 1949.

DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Professor R. Kevin Wood Code 55Wd Naval Postgraduate School Monterey, California 93943	5
4.	Professor Richard E. Rosenthal Code 55Ro Naval Postgraduate School Monterey, California 93943	1
5.	Professor Gerald G. Brown Code 55Bw Naval Postgraduate School Monterey, California 93943	1
6.	LT Willard R. Bonwit, Jr., USN 1464 Rydal Road Rydal, Pennsylvania 19046	2

END

FILMED

1-85

DTIC