The Design of Minimum-Cost Survivable Networks

KENNETH STEIGLITZ, MEMBER, IEEE, PETER WEINER, MEMBER, IEEE, AND D. J. KLEITMAN

Abstract—We consider the problem of designing a network which satisfies a prespecified survivability criterion with minimum cost. The survivability criterion demands that there be at least r_{ij} node disjoint paths between nodes i and j, where (r_{ij}) is a given redundancy requirement matrix. This design problem appears to be at least as difficult as the traveling salesman problem, and present techniques cannot provide a computationally feasible exact solution.

A heuristic approach is described, based on recent work on the traveling salesman problem, which leads to a practical design method. Algorithms are described for generating starting networks, for producing local improvements in given networks, and for testing the redundancy of networks at each stage. This leads to networks which are locally optimum with respect to the given transformation. Randomizing the starting solution ensures that the solution space is widely sampled. Two theorems are given which allow great reduction in the amount of computation required to test the redundancy of a network. Finally, some design examples are given.

I. Statement of the Problem

AN IMPORTANT consideration in the design of a communication or transportation network is the degree to which connectivity between given pairs of nodes is vulnerable to the failure or destruction of other parts of the network. The introduction of redundancy in the network will increase its survivability, but at the same time will increase the cost of construction and maintenance. This leads to the problem of designing at minimum cost a network to meet prespecified redundancy requirements.

Specifically, we assume that we are given n nodes and the cost matrix $C = (c_{ij})$, where c_{ij} is the cost of building and maintaining a branch between nodes i and j. The treatment will be restricted to undirected graphs, so that C will be assumed to be symmetric. A set of paths between nodes i and j will be called *node disjoint* if they share no nodes other than i or j. The redundancy r_{ij} is defined as the maximum number of node disjoint paths between nodes i and j. A redundancy of r_{ij} between nodes i and j means that at least r_{ij} nodes (other than i and j) and/or branches must be destroyed before i and j are disconnected. This concept of redundancy is independent of probabilistic considerations, and determines in a sense a minimum enemy effort required to disrupt a strategic communication network.

Manuscript received July 31, 1968; revised February 7, 1969. This work was supported principally by the Office of Emergency Preparedness, Executive Office of the President, and also by the U. S. Army Research Office, Durham, N. C., under Contract DAHC 04-69-C-0012.

K. Steiglitz is with the Department of Electrical Engineering, Princeton University, Princeton, N. J. 08540.

P. Weiner was with the Department of Electrical Engineering, Princeton University, Princeton, N. J. 08540. He is now with the Department of Computer Science, Yale University, New Haven, Conn.

D. J. Kleitman is with the Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

The problem mentioned above can then be stated precisely as follows. Find an undirected network that has the following properties.

1) Feasibility—The redundancy between any two nodes i and j is at least r_{ij}, where $R = (r_{ij})$ is a given matrix,
2) Optimality—No network which satisfies 1) has lower cost.

Notice that the network redundancy is required to be at least, but not precisely, r_{ij}. Hence the question of the exact realization of a given R does not arise.

When the cost matrix C is uniform (c_{ij} = constant for all i and j), the problem becomes one of designing a network with the minimum number of branches. In the special case of uniform C and R, a complete explicit solution has been given [1]–[4]. On the other hand, the synthesis problem with uniform C and arbitrary R has not been completely solved, although the problem has received attention and many special cases have been worked out [5]. In addition, design problems with different definitions of redundancy have been considered [6].

II. Heuristic Approach

The general design problem as stated above can be formulated as an integer programming problem, but the number of constraint equations quickly becomes astronomical for even small problems. In this respect the problem is similar to the traveling salesman problem, which has long resisted attempts at analytical solutions that are computationally practical. Another similarity stems from the fact that a minimum length closed tour satisfies a redundancy requirement of 2 between every pair of nodes. Hence the survivable network design problem with arbitrary cost and a uniform redundancy requirement of 2 has a good if not optimal solution provided by the solution to the traveling salesman problem. For these reasons, plus the difficulties encountered in the uniform cost case, it appears that present analytical methods cannot provide a practical exact solution to our problem. Recent work on approximate methods for the traveling salesman problem [7], [8] has been quite promising, however, and the approaches used for that problem suggested the techniques which we shall use here to develop practical algorithms for designing low-cost survivable networks.

By a "feasible" solution we shall mean one which satisfies condition 1) above. By an "optimal" solution we shall mean a feasible one which satisfies condition 2). Our goal is to present a practical method for finding feasible solutions with costs close to optimal. By a "practical"

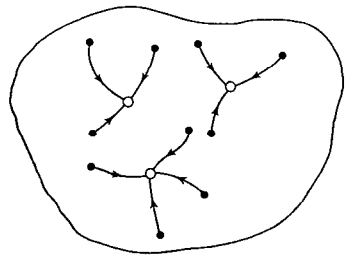

Fig. 1. Diagrammatic representation of the hill-climbing procedure.

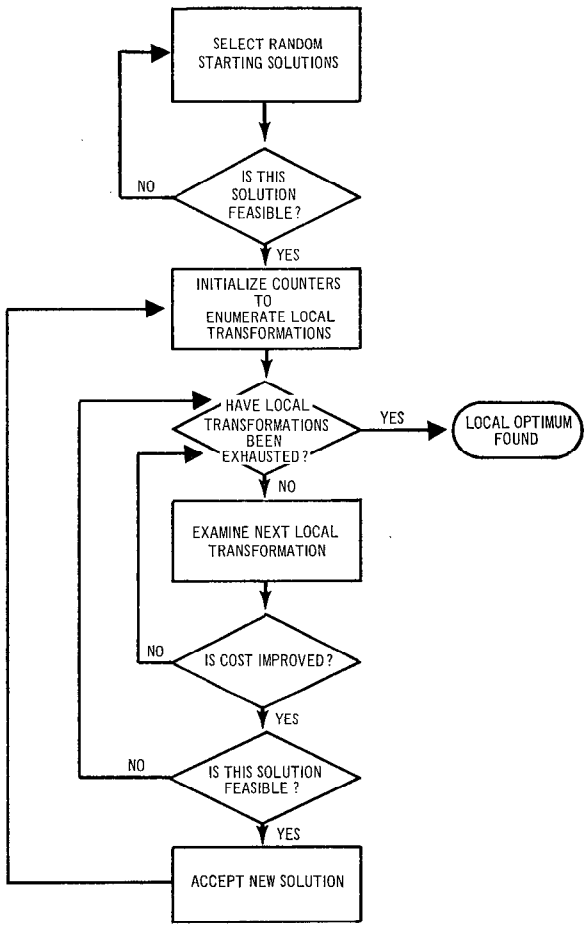

Fig. 2. Flow diagram of the hill-climbing procedure.

method we mean one which will handle realistically large problems in reasonable computation time.

The method to be described here has two main parts, which may be called the *starting routine* and the *optimizing routine*, respectively. The starting routine generates a feasible solution. The optimizing routine then searches networks generated by local transformations for a similar network with a lower cost. When such a local improvement is found that is also feasible, the improvement is adopted and the search continued from this solution, until no further local improvements can be made. In this way there is eventually reached a feasible network which is "locally optimal" in the sense that no local transformations of the type considered result in a feasible network with lower cost.

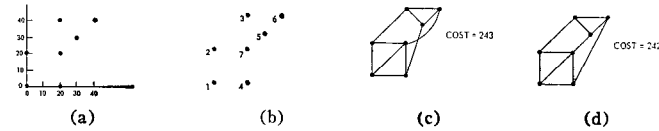

Fig. 3. A seven-node illustrative example.

Once a locally optimum solution is found, the entire process is repeated using the starting routine again. This starting routine is randomized, and by hill climbing to local optima from different random starts a variety of solutions can be found. Fig 1 shows a diagrammatic representation of this process, where the space of feasible solutions is represented by the entire figure, random starting solutions are represented by dark circles, and local optima by light circles. The practicality of the approach is based on the assumption that random local optima are, with high probability, close in cost to the global optimum. This assumption is sensitive to the particular local transformation used and must be verified experimentally in a given application.

Fig. 2 shows a flow diagram of the method. The procedure will be described in detail in the next sections.

III. The Randomized Starting Routine

The purpose of this algorithm is to generate a network that has a reasonable probability of being feasible, and that at the same time has a relatively low cost. Its operation is based on the observation that the degree of node i must be at least $\max_j r_{ij}$ if there are to be r_{ij} node disjoint paths between nodes i and j. This gives a lower bound on the number of branches at each node. Branches are added to the network one at a time between the node with the highest (updated) requirement and a node with the next highest (updated) requirement. Of all those nodes with the next highest requirement, that one is chosen that results in the smallest increase in cost, with the constraint that no parallel branches are allowed. All ties are resolved by choosing the node highest on the list of nodes, and the process terminates when the maximum requirement is zero.

The routine is made nondeterministic by ordering the nodes uniformly at random at the start of each execution. This has been observed experimentally to produce a rich variety of candidate networks, often with different numbers of branches.

The operation of the starting routine will now be illustrated by the seven-node example shown in Fig. 3(a), where the cost matrix is obtained from the Euclidian distance in the plane, truncated to an integer, and the requirement matrix is uniformly 3. Fig. 3(b) shows the random node ordering. We keep track of the updated requirements at each stage by writing

node: 1 2 3 4 5 6 7
requirement: 3 3 3 3 3 3 3.

The highest requirement is 3, and therefore the first branch to be added will start from node 1. The next

highest requirement is also 3, and of all those nodes with this requirement, node 2 (tied with 4) is closest to node 1. The updated requirements become

node:	1	2	3	4	5	6	7
requirement:	2	2	3	3	3	3	3.

Next are added in turn the branches[1] (3, 5), (4, 7), (6, 5), (1, 4), (2, 7), (3, 6), (1, 7), (2, 3), (4, 5), and (6, 7). The result shown in Fig. 3(c) is feasible and has a cost of 243. For this problem the starting routine required, on the average, about three trials to produce a feasible solution.

Fig. 3(d) shows the optimum solution, with a cost of 242, obtained from the hill-climbing routine. The global optimality was verified by an exhaustive program which guarantees optimality but which is limited by computation time to problems with about eight nodes. The next section contains a description of the local transformation used to hill-climb.

In some applications it might be desirable to use some other method to obtain a starting network. When the problem can be represented graphically, for example, human suggestions might be more effective than the heuristic procedure described here.

IV. The Local Transformation

The particular local transformation selected for this problem is called an X-change, and networks that are local optima with respect to X-changes in the space of feasible networks are called X-opt. This transformation is motivated by similar transformations for the traveling salesman problem such as Croes' inversion [7] and Lin's generalization to λ-change [8].[2]

If a network N has two branches, say (i, m) and (j, l) such that (i, l) and (j, m) are not in N, then an X-change on the network N, producing a network N', is defined by the operation of removing the branches (i, m) and (j, l), and adding the branches (i, l) and (j, m) (see Fig. 4).

If

$$c_{il} + c_{jm} < c_{im} + c_{jl}$$

we say that the X-change is favorable, since the network N' has a smaller cost than N. The presence or absence of the branches (i, j) or (m, l) is irrelevent.

An X-change has the property that the degrees of all nodes are preserved. The fact that N is feasible does not imply, however, that N' is feasible. The network of Fig. 3(d) can be obtained from that of Fig. 3(c) by the X-change replacing branches (4, 5) and (6, 7) by (4, 6) and (5, 7). The X-change in this case is favorable, preserves feasibility, and in fact produces the globally optimal solution.

[1] The symbol (i, j) will represent the undirected branch between nodes i and j.

[2] In the traveling salesman problem, an *inversion*, or a *2-change*, is defined by removing two branches from the tour, and adding two other branches to restore the graph to a tour. Similarly, a *3-change* involves replacement of three branches. The X-change described below is in fact a generalization of the 2-change, and reduces to a 2-change if it is applied to a tour and if it preserves feasibility.

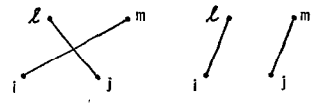

Fig. 4. Diagram of an X-change.

V. Testing Feasibility of a Random Starting Solution

The method used to calculate the redundancy between a given pair of nodes is due to Frisch [9]–[12] and will not be described here. The algorithm is similar to the labeling algorithm of Ford and Fulkerson [13] and requires computation time linear in the number of nodes n.

The feasibility of a candidate solution can be checked, of course, by calculating the redundancy between all possible pairs of nodes. This requires $n\ (n-1)/2$ applications of Frisch's labeling algorithm. However, not all of these checks are necessary in a given situation, and the number of redundancy calculations required can be reduced significantly, especially in the case of uniform R. To see this, we prove the following result, where r_{ij} denotes a redundancy requirement, and r'_{ij} actual redundancy.

Theorem 1

If

$$r'_{am_i} \geqq r_{ab} \tag{1}$$

$$r'_{bm_i} \geqq r_{ab} \tag{2}$$

for nodes $m_1, m_2, \cdots, m_{r_{ab}}$, distinct from a, b, and each other, then

$$r'_{ab} \geqq r_{ab}.$$

Proof: Suppose $r'_{ab} < r_{ab}$, that is, that the redundancy requirement between a and b is not satisfied. Assume first that the branch (a, b) is not present. Then there exists a node cutset Q with $k < r_{ab}$ nodes, which separates the network into two or more disconnected subgraphs $G_1, G_2, \cdots$, with $a \in G_1$ and $b \in G_2$ (see Fig. 5). Node m_i must lie in the cutset, for if it lay in G_1, condition (2) would be impossible; if it lay in G_2, condition (1) would be impossible; and if it lay in G_i, $i > 2$, both conditions would be impossible. Hence, there are r_{ab} distinct elements in the cutset, which is a contradiction. If the branch (a, b) is present, the argument is essentially unchanged. Q.E.D.

This result enables one to eliminate the need for checking r_{ab} whenever the conditions of the theorem hold. In such a case, we say that the entry r_{ab} of the matrix R is *dominated* by previous checks. In the case where R is uniformly k, we need only perform the following redundancy checks: first, between any node m_1 and all remaining nodes; second, between any node m_2 and all remaining nodes except m_1; $\cdots$; and so on, up to m_k. This is a total of

$$(n-1) + (n-2) + \cdots + (n-k)$$

$$= kn - k(k+1)/2$$

tests, which for large problems represents a considerable saving over $n\ (n-1)/2$.

In the nonuniform R case it is not clear how to order the redundancy tests so as to take maximum advantage of Theorem 1. The following *ad hoc* procedure was programmed and seems to perform well. It reduces to the procedure described above when R is uniform.

1) Select the largest unchecked entry r_{ij} of R.
2) If it is dominated, consider it checked.
3) If it is not dominated, calculate r'_{ij} and check the entry if $r'_{ij} \geqq r_{ij}$.
4) Go to 1) until every element of R is checked.

Of course, when a network is being tested for feasibility and a given redundancy requirement is not met, the checkout need not be continued further. Empirically, it has been observed that certain pairs of nodes in a network are more likely to reveal such infeasibility than others. It is also true that infeasibility is usually revealed at an early stage of a checkout, and that computation time is determined largely by the cost of a complete checkout of a feasible solution.

VI. Testing Feasibility After an X-Change

If an X-change is performed on a feasible network, a different economy can be effected in checking feasibility of the new network. We need the following result, where r_{ij} denotes a redundancy requirement and r'_{ij} denotes the actual redundancy after an X-change.

Theorem 2

If an X-change on a feasible network destroys feasibility by reducing the redundancy between nodes a and b below the requirement r_{ab}, and if the X-change resulted in the removal of branches (i, m) and (j, l), then either

$$r'_{im} < r_{ab}$$

or

$$r'_{jl} < r_{ab}.$$

Proof: Assume first that the branch (a, b) is not present in the original network. After the X-change a and b have a redundancy of less than r_{ab}, so that there is a node cutset Q with $k < r_{ab}$ nodes, separating the network into two or more disconnected subgraphs $G_1, G_2, \cdots$, with $a \in G_1$ and $b \in G_2$ (see Fig. 6). Either i and m are disconnected by the cut, or j and l are; for otherwise, a and b would not have had the required redundancy before the X-change. Assume without loss of generality that i and m are disconnected by Q. Then

$$r'_{im} \leqq k < r_{ab}.$$

The argument is not essentially changed if (a, b) was present in the original network. Q.E.D.

From this theorem it follows that whenever

$$r_{ab} \leqq \min\ (r_{im}, r_{jl}),$$

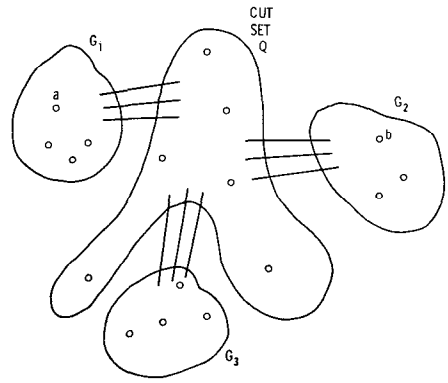

Fig. 5. Diagram illustrating the construction of Theorem 1.

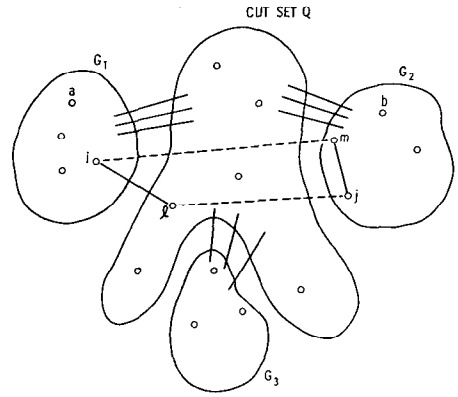

Fig. 6. Diagram illustrating the construction of Theorem 2.

any infeasibility will be discovered after an X-change by checking r'_{im} and r'_{jl}. In addition to r'_{im} and r'_{jl}, then, we need only check redundancy between those pairs a and b such that

$$r_{ab} > \min\ (r_{im}, r_{jl}).$$

When R is uniform, only two redundancies need be calculated after an X-change. This is in contrast with naïvely calculating all $n\ (n-1)/2$ redundancies, and results in an enormous saving. The saving in the nonuniform case will depend on the particular X-change and the distribution of values in R.

VII. Examples

1) The first example has already been discussed to some extent, and is shown in Fig. 3. As mentioned, the solution with a cost of 242 has been verified to be optimum. To illustrate the randomized aspect of the method, 100 local optima were found, with the following results:

Cost	Frequency
242	39
245	9
250	20
251	7
253	6
258	1
265	9
270	9

In this simple example the global optimum is found more frequently than the other local optima. The average computation time was about 0.2 second per local optimum on the Univac 1108.

2) To illustrate the method when applied to a problem with nonuniform redundancy requirements, the network shown in Fig. 7 was analyzed. This network represents the first stage in the construction of the Continental United States Automatic Voice Network (CONUS AUTOVON) [14], [15]. This network contained ten switching machines (nodes) and existed at the end of 1965. The calculated redundancy matrix is given below, and the cost is 67522 (the units are airline miles multiplied by $\sqrt{10}$).

0	3	4	3	4	3	3	3	3	3
	0	3	3	3	3	3	3	3	3
		0	4	6	5	5	4	4	4
			0	4	4	4	4	4	4
				0	5	5	5	4	5
					0	5	5	4	5
						0	5	4	5
							0	4	5
								0	4
									0

This matrix was used as a requirement matrix and 100 local optima found with the following result:

Cost	Frequency
65692	5
65770	10
65822	9
65882	14
66409	8
66593	1
66647	1
66701	6
66711	17
66780	16
67035	2
67044	5
67468	3
68313	1
69373	1
70978	1

The network with the best score of 65692 is shown in Fig. 8, and has a cost about 2.7 percent lower than the network of Fig. 7 with at least as much redundancy between each pair of cities. It also has one less branch. Note that in this example the most frequently obtained local optimum is not the lowest in cost. This problem required about 3.4 seconds per local optimum on the Univac 1108.

3) The last example illustrates the method on a rather large-scale problem, a mosaic representing 58 switching centers of CONUS AUTOVON. Cost was determined by distance in this regular pattern. In a real application actual geographical distances or cost in dollars could be used. The redundancy requirement was taken to be uniformly 6. Fig. 9 shows the best solution obtained after ten local optima were found, each taking about 12 minutes on the Univac 1108. The solutions obtained are superficially similar to the polygrid pattern used in CONUS AUTOVON, but have uniform redundancy. Thus, connections with cities near the edge of the pattern have the same survivability as those with geographically interior cities.

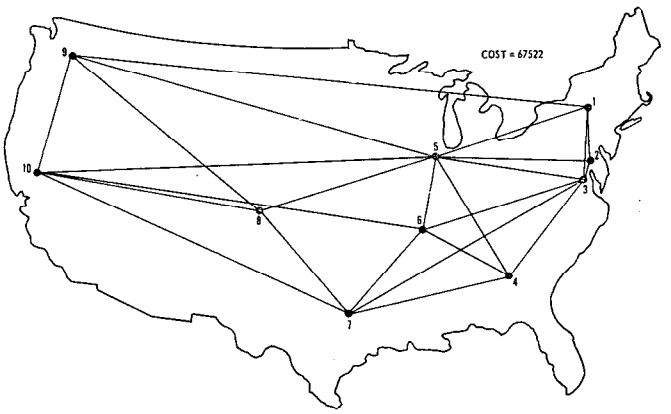

Fig. 7. CONUS AUTOVON in 1965.

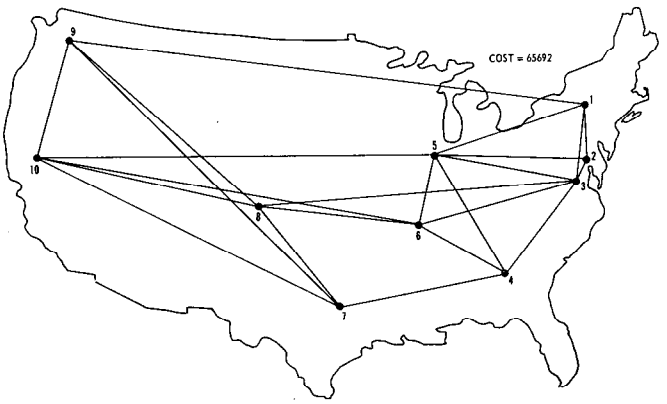

Fig. 8. A ten-city network designed in Example 2.

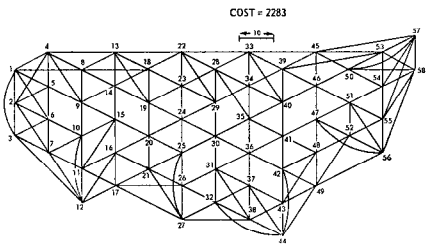

Fig. 9. A 58-city network designed in Example 3.

VIII. COMMENTS

A practical method has been described for designing low-cost survivable networks, a problem for which present analytical methods fail. The approach is based on recent advances in heuristic programming, and especially Lin's work on the traveling salesman problem. In addition, recently developed tools of network analysis, such as Frisch's flow algorithm for calculating redundancy, have been indispensable.

The heuristic approach described here depends very much on interaction between the human and the computer. The selection of starting solutions and local transformations relies both on human intuition and computational verification. In this paper, for example, the randomized starting routine and X-opt were arrived at only after examining many alternative schemes.

One advantage of the heuristic approach is that of producing a number of distinct solutions with near optimal cost. This allows one to take into account nonanalytic constraints when selecting the final solution.

Finally, we mention that further effort along the lines described here will very probably be rewarded by improved performance, in the sense that better solutions will be obtained in less time. A local transformation which replaces three branches is a natural generalization of the X-change, just as 3-opt is a generalization of 2-opt. Further economies are possible in the feasibility checkout.

Acknowledgment

The authors are indebted to Dr. D. M. Rosenbaum, who suggested this problem, and who encouraged this work at every stage. Thanks are due also to N. J. Ray and D. L. Parrish for programming help.

References

[1] D. Rosenbaum and J. B. Friedman, "Redundant networks," MITRE Corp., Bedford, Mass., Rept. TM-3195, October 1961.
[2] F. Harary, "The maximum connectivity of a graph," *Proc. Natl. Acad. Sci.*, vol. 48, pp. 1142–1146, July 1962.
[3] F. T. Boesch and R. E. Thoams, "Optimal damage resistant nets," *Proc. IEEE Iternatl. Conf. on Communications*, vol. 4, pp. 688–693, 1968.
[4] S. L. Hakimi, "An algorithm for the construction of the least vulnerable communication network," Northwestern University, Evanston, Ill., 1968 (unpublished paper).
[5] H. Frank, private communication.
[6] ——, "Vulnerability of communication networks," *IEEE Trans. Communications Technology*, vol. COM-15, pp. 778–789, December 1967.
[7] G. A. Croes, "A method for solving traveling salesman problems," *Operations Res.*, vol. 6, pp. 791–812, 1958.
[8] S. Lin, "Computer solutions of the traveling salesman problem," *Bell Sys. Tech. J.*, vol. 44, pp. 2245–2269, December 1965.
[9] I. T. Frisch, "An algorithm for vertex pair connectivity," *Internatl. J. Control*, vol. 6, pp. 579–593, 1967.
[10] ——, "Analysis of the vulnerability of communications nets," *Proc. 1st Ann. Princeton Conf. on Systems Science* (Princeton, N. J., March 1967).
[11] F. T. Boesch and I. T. Frisch, "On the smallest disconnecting set in a graph," *IEEE Trans. Circuit Theory* (Correspondence), vol. CT-15, pp. 268–288, September 1968.
[12] I. T. Frisch, "Flow variation in multiple min-cut calculations," *J. Franklin Inst.* (to be published).
[13] L. R. Ford, Jr. and D. R. Fulkerson, *Flows in Networks*, Princeton, N. J.: Princeton University Press, 1962.
[14] L. W. Helke, "CONUS AUTOVON network and traffic routing plan," presented at the 1966 IEEE Internatl. Communications Conf., Philadelphia, Pa., June 15–17, 1966, paper CP 19CP66-1173.
[15] "Network plan—CONUS AUTOVON," AT & T Co., December 1, 1967.

Computation of DC Solutions for Bipolar Transistor Networks

HAROLD SHICHMAN, MEMBER, IEEE

***Abstract*—The dc state vector is often the desired initial condition for the solution of a system of first-order differential equations that characterize network dynamics. A computer algorithm is described in this paper that computes the dc solution of first-order differential equations that characterize networks containing transistors, diodes, capacitors, inductors, resistors, and voltage sources with neither cut sets nor closed loops of either junctions, capacitors, or inductors. Networks containing current sources were not considered. The Newton–Raphson iteration function is the basis of the dc solution algorithm. The unique feature of the solution procedure is the use of upper bounds on the solution to avoid slow convergence and difficulties in computing exponential functions. Derivation of the upper solution bounds is discussed in detail.**

I. Introduction

THIS PAPER is concerned with the calculation of dc solutions of the system of first-order differential equations characterizing the dynamic behavior of networks of resistors, bipolar transistors, and diodes, capacitors, and inductors assuming quiescent excitation of the system [1].[1] These dc solutions can, of course, be useful end products of analysis. More importantly, in transient network analysis, the dc solution can give the initial conditions for the transient solution of the system of the differential equations [4]–[6].

The numerical algorithm described in this paper is based upon the Newton–Raphson iteration function [2], [3]. Other workers who have used the Newton–Raphson function to compute dc solutions for networks containing bipolar transistors reported difficulties in evaluating exponential functions, slow convergence of the sequence of approximate solutions generated by the iteration function, and the necessity of excluding problems that contain cut sets of capacitors and/or closed loops of inductors [5]–[7], [14].

Manuscript received November 7, 1968.
The author is with Bell Telephone Laboratories, Inc., Murray Hill, N. J.

[1] Networks containing current sources were not considered.