1958

AN ALGORITHM FOR SOLVING "TRAVELING-SALESMAN" AND RELATED NETWORK OPTIMIZATION PROBLEMS

Frederick Bock
Armour Research Foundation

Presented at the Fourteenth National Meeting
of the Operations Research Society of America

St. Louis, October 24, 1958

AN ALGORITHM FOR SOLVING "TRAVELING-SALESMAN" AND RELATED NETWORK OPTIMIZATION PROBLEMS

ABSTRACT

The algorithm has two complementary phases which are applied alternately in the solution of network optimization problems of the "traveling-salesman" type. Phase 1 proceeds from circuit to circuit by one-step transformations, always decreasing the circuit value, until either a relative or an absolute minimum value is attained. Phase 2 operates on a table of links arranged in order of increasing link values, combining conditionally minimum-valued circuit fragments in a systematic and exhaustive search for a complete circuit having a value less than the value previously attained in phase 1. The algorithm terminates when phase 2 fails to find an improved circuit. The algorithm has been programmed for a digital computer. Computational experience is reported.

AN ALGORITHM FOR SOLVING "TRAVELING-SALESMAN" AND RELATED NETWORK OPTIMIZATION PROBLEMS

I. INTRODUCTION

The algorithm described in the present paper solves the following "traveling-salesman" problem: given the $m(m-1)/2$ link values of an m-node simplex, to find that circuit including all m nodes which has minimum value. A simplex is a network with a direct and symmetrical connection (link) between each pair of nodes. The integers $1,\ldots,m$ are assigned arbitrarily to the m nodes for the purpose of indexing these points. A link is indexed by the pair of nodes which it connects. A circuit is represented by a sequence of nodes, the first and last nodes of the sequence being identical. The value of a circuit is defined as the sum of the values of the component links.

The basic notation employed in the statement of the algorithm is explained in Table 1. Additional notation will be clear from the context. The "replace" operator (←) signifies that the stored or computable numbers on the right become the values of the variables on the left. The detailed statement of the algorithm is given in the form of flow diagrams (see Figures 2-6).

II. THE TWO COMPLEMENTARY PHASES

The algorithm has two complementary phases which are employed alternately in the solution of problems of the type described above. Phase 1 examines a given circuit to determine whether an improved circuit can be obtained by performing any one of a set of elementary transformations. If an improved circuit is found, it in turn is examined for possible improvement. Phase 1 terminates whenever the examination of a circuit is completed without finding an improvement. At this point one proceeds to the employment of

Table 1

NOTATION

m	The number of nodes in a simplex
i	Range: 1,...,m. Node index
j	Range: 1,...,m. Node index. Nodes adjacent to given node i in a circuit or partial circuit are denoted by $j_1(i)$ and $j_2(i)$
n	The number of links in a simplex, i.e., (m(m-1))/2
D	The value of a link (positive real number). $D(i_1,i_2)$ is the value of the link connecting nodes i_1 and i_2; $^1D(k)$ is the value of link k
k	Range: 1,...,n. Index of links arranged in order of increasing value
h	Range: 1,...,(m+1). Index of the nodes of a complete circuit arranged in standard form, i.e., beginning and ending with node 1
U	Value of a complete circuit; the sum of the values of the component links
X	Range: 1,...,(m-1). The number of links in a partial circuit
x	Range: 1,...,X. Index of the links in a partial circuit in the order of their introduction
F	The value of a partial circuit
B	For given X and given k>k(x=X), B(k) is the sum of the values of links k,...,(k+m-X-1)

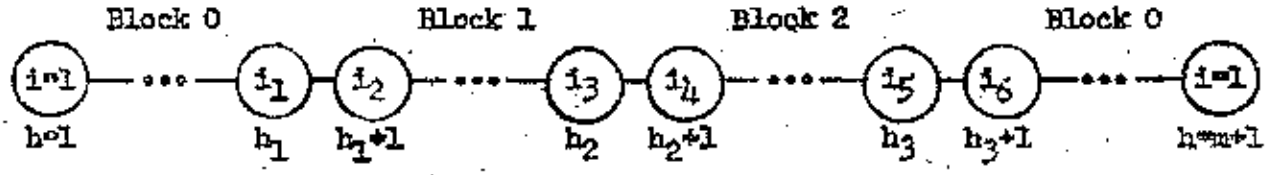

Fig. 1. Diagram of a circuit with indexing used in phase 1 of the traveling-salesman algorithm

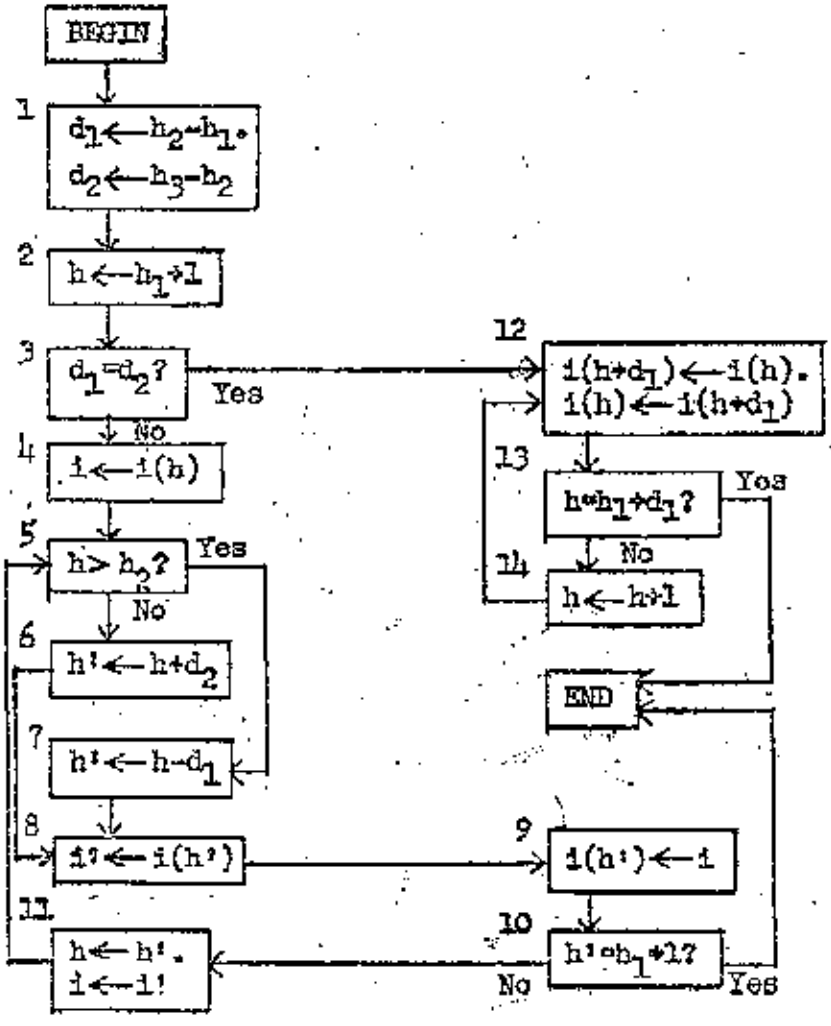

Fig. 2. Flow diagram, interchange of blocks 1 and 2, phase 1 of traveling-salesman algorithm

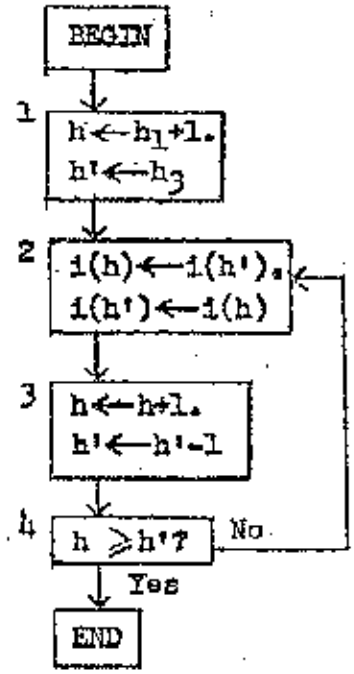

Fig. 3. Flow diagram, reversal of blocks 1 and 2 as a unit, phase 1 of traveling-salesman algorithm

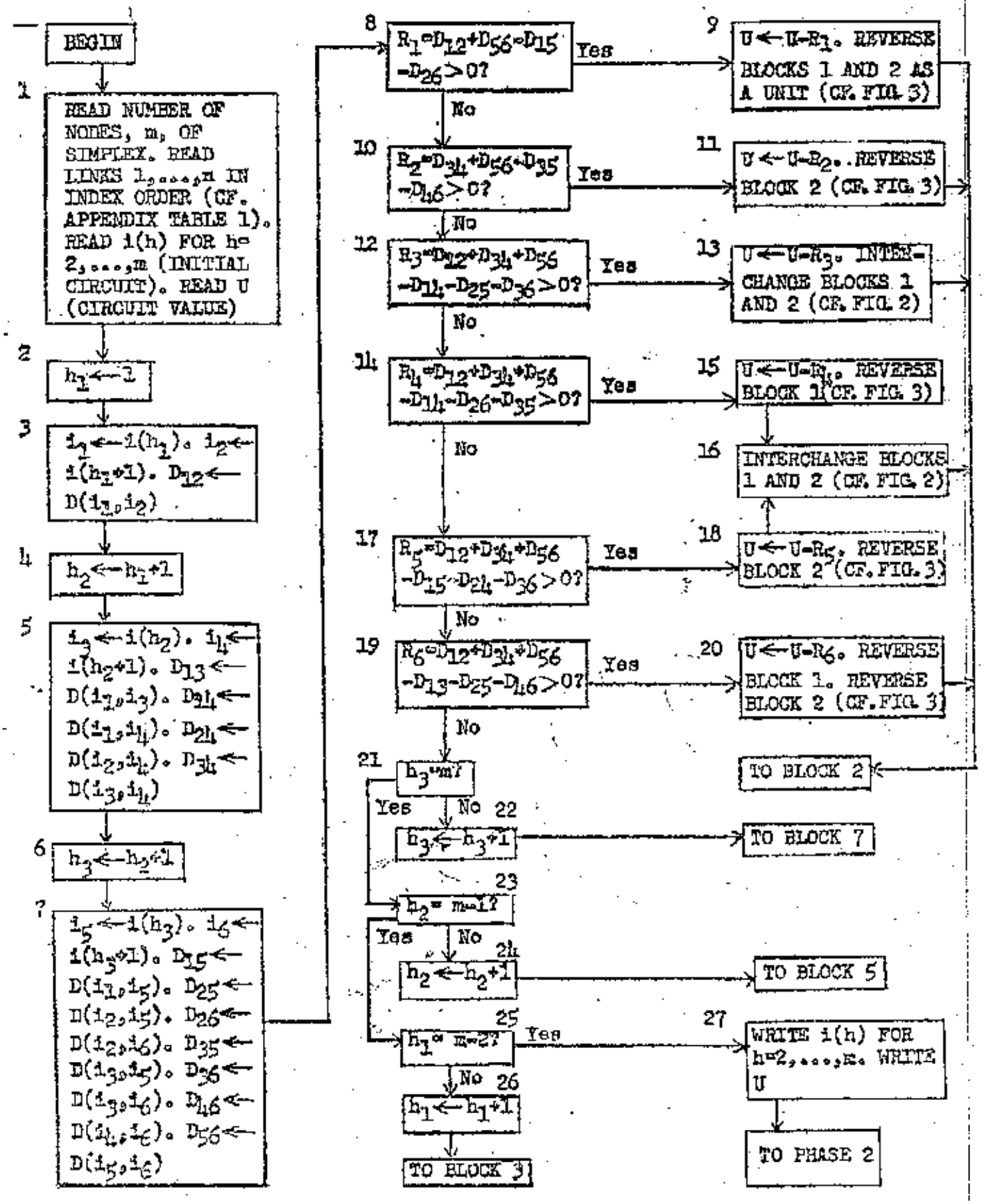

Fig. 4. Overall flow diagram, phase 1 of traveling-salesman algorithm

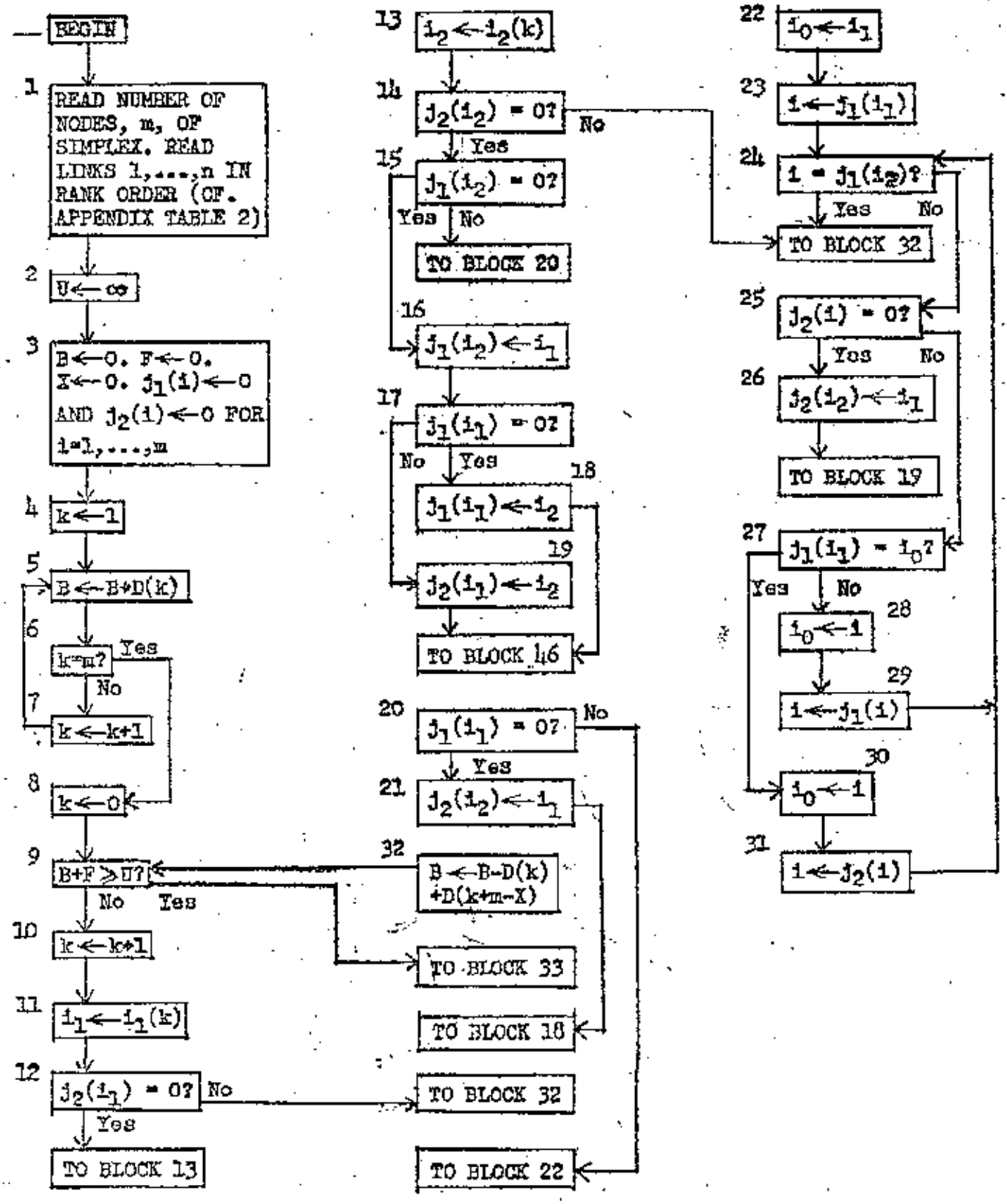

Fig. 5. Flow diagram, phase 2 of traveling-salesman algorithm (first part)

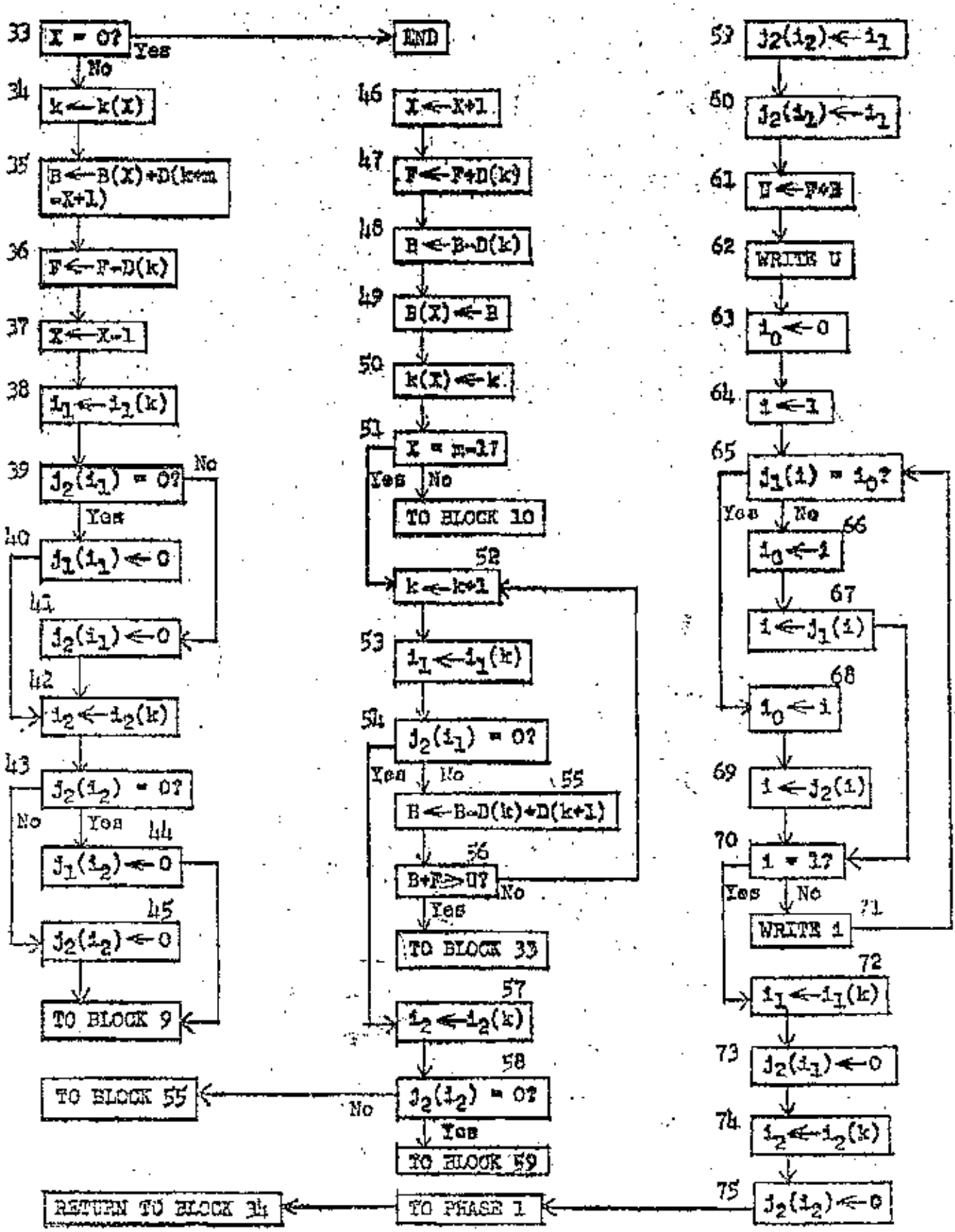

Fig. 6. Flow diagram, phase 2 of traveling-salesman algorithm (last part)

phase 2.

Phase 2 examines a list of the links arranged in order of increasing link values in a search for a combination of links forming a complete circuit and having a value less than the smallest value previously attained in phase 1. When such a combination is found, it is used as input to phase 1. Combinations increase, and also decrease, in number of links by the introduction or deletion of one link at a time. In the absence of an effective upper bound on the minimum circuit value phase 2 would generate all possible complete circuits.

The algorithm may be initiated either by entering phase 1 with any known circuit or by entering phase 2 with an artifically large number as the value of the best circuit known. The algorithm terminates when phase 2 fails to find an improved circuit.

III. PHASE 1, DETAILED DESCRIPTION

The set of elementary transformations of a circuit, with respect to which the search for improvement is conducted, consists of all rearrangements of the nodes of the circuit which are made possible by deletions of three links at a time. Let $h = 1,\ldots,(m+1)$ be an index of the nodes of a circuit arranged in standard form. Let h_1, h_2, and h_3 be three particular values of h such that $h_1<h_2<h_3$. Then the pairs of nodes at positions h_1 and h_1+1, h_2 and h_2+1, and h_3 and h_3+1 define three links of the circuit (see Fig. 1). Deletion of these three links (but not the nodes which are their endpoints), resulting in the division of the circuit into three blocks (block 0, block 1, and block 2 of Fig. 1), allows seven possible rearrangements of the circuit:

1. Reverse block 1 and block 2 as a single unit (i.e., turn these blocks around so that node i_5 is adjacent to node i_1 and node i_2 is adjacent to node i_6)

2. Reverse block 2

3. Interchange blocks 1 and 2

4. Reverse block 1; then interchange blocks 1 and 2

5. Reverse block 2; then interchange blocks 1 and 2

6. Reverse block 1 and also reverse block 2

7. Reverse block 1.

Consequent on any of the rearrangements the three gaps existing in the circuit are closed by introduction of the links defined by the corresponding pairs of nodes. (The net result of transformations 1, 2, and 7 is the deletion of two links and the insertion of two other links.)

Procedures for the interchange of two blocks, and for the reversal of a block, are illustrated by the flow diagrams of Figs. 2 and 3.

Figure 4 is the overall flow diagram for phase 1 of the algorithm. Systematic variation of h_1, h_2, and h_3 generates all possible combinations of the m links of the circuit taken three at a time. $R_1,\ldots,R_6$ are the amounts by which the circuit value U would be decreased or increased if rearrangements $1,\ldots,6$ respectively were carried out (rearrangement 7 is redundant in the context of the complete set of threefold link combinations). After all possible improvements are made with respect to the set of transformations, then either an absolute or a local minimum has been found. The value of U that has been attained is subsequently utilized in phase 1 as an upper limit on values of sets of links to be examined for feasibility.

IV. PHASE 2, DETAILED DESCRIPTION

The flow diagram is shown in two parts (Figs. 5 and 6). Links are arranged in order of increasing value and, in this order, are indexed by $k = 1,\ldots,n$. The two nodes connected by link k are designated $i_1(k)$ and

$i_2(k)$. Two additional lists are set up. For each of nodes i = 1,...,m the identity of the adjacent nodes, $j_1(i)$ and $j_2(i)$, in the current (partial) circuit is recorded. A zero value for $j_2(i)$ alone indicates that node i is at the moment directly linked to just one node, i.e., to $j_1(i)$. A zero value for both $j_1(i)$ and $j_2(i)$ indicates that node i is at the moment linked to no node.

The second current list is of the X links constituting the partial circuit, arranged in the order of their introduction and indexed by x = 1,...,X. A link is identified by the value of its index, k. In addition to k(x), B(x) is also recorded. If X is the number of links in a partial circuit with total value F, and if link k>k(x=X) is about to be examined for possible introduction into the partial circuit, then B is the sum of the values of the m-X links k,...,(k+m-X-1). If link k is introduced into the partial circuit, then X is increased by one and B is stored as B(x=X). The initial value of B is computed in blocks 4-7 of Fig. 5.

At block 9 of Fig. 5 the test is made whether B + F equals or exceeds the value, U, of the best circuit so far attained. If so, then either link k(x=X) is deleted from the partial circuit and the search of the link table is resumed at that point (blocks 33-45 of Fig. 6) or, if X = 0, the algorithm terminates because the best circuit so far attained has been proved to have minimum value. If B + F is less than U, then link k is tested for compatibility with links k(x) and, if compatible, is introduced into the partial circuit (blocks 10-32 of Fig. 5 and Blocks 46-51 of Fig. 6). Introduction of an incompatible link would result in either a branch point or a subcircuit containing fewer than m nodes. In the special case where the partial circuit contains m-1 links, the subcircuit test is not made (blocks 52-58 of Fig. 6).

Blocks 59-75 of Fig. 6 insert the final link in a circuit, enter the reduced circuit value, trace the completed circuit in the list of $j_1(i)$ and $j_2(i)$, and delete the final link just introduced. The algorithm then normally transfers to phase 1, although it is optional to continue with phase 2. On the return to phase 2 from phase 1, the search of the link table is resumed at the point where it was previously broken off.

V. COMPUTER PROGRAM AND COMPUTATIONAL EXPERIENCE

The algorithm has been programmed for a digital computer (IBM Type 650) and tested on a number of examples of traveling-salesman problems. Sample experience will be related.

Ex. 1. Robacker has presented solutions to ten 9-node traveling-salesman problems (Ref. 1). The minimum circuits were obtained by him by means of Dantzing's linear-programming method (Ref. 2). Utilizing the computer to solve the first of the ten examples, the minimum circuit (1-2-6-3-8-5-4-7-9-1 with a value of 232) was obtained and proved minimum in about a minute. Initial entry was into phase 2. This provided a starting circuit of value 408 for phase 1. The minimum circuit was found by phase 1 through six improvements.

Ex. 2. Barachet has presented a conjectured solution to a 10-node problem (Ref. 3) using a graphical procedure. His conjectured solution (1-2-3-4-5-10-9-8-6-7-1 with a value of 378) was proved to be indeed optimum. Computing time: about 40 minutes. Initial entry was into phase 2 which provided a starting circuit of value 425 for phase 1. The minimum circuit was obtained by phase 1 through four improvements.

Ex. 3. Transformed road distances between ten cities (Akron, Atlanta, Baltimore, Birmingham, Bismarck, Boston, Buffalo, Cheyenne, Chicago, and Cincinnati, corresponding to i = 1,...,10 respectively) are presented in

Tables A-1 and A-2 of the Appendix. The road distances (Table A-3) are from a Rand-McNally atlas. Let $S(i_1,i_2)$ be the road distance between city i_1 and city i_2. Then the transformation employed is

$$D(i_1,i_2) = 2.S(i_1,i_2) - \min_j (S(i_1,j)) - \min_j (S(i_2,j))$$

where $\min_j (S(i,j))$ is the road distance between city i and the closest city. The minimum circuit is 1-7-6-3-2-4-8-5-9-10-1 with a value of 4142 which transforms into 5344 miles. Computation time was about 15 minutes. Initial entry into phase 2 provided a starting circuit for phase 1 of value 4786. The minimum circuit was found by phase 1 through one improvement.

On the basis of computational experience with these and other examples it is concluded that the algorithm in its present form provides a practicable method for solving traveling-salesman problems of moderate size. The speed of convergence to a proved minimum circuit is highly sensitive to the structure of the individual problem. For example, a twenty-node example in which the twenty smallest links happened to form a circuit was solved immediately upon entry of the data into phase 2. On the other hand, it was concluded from the impracticably slow rate of convergence in the case of Dantzig's 42-city example (Ref. 2) that improvement of the algorithm lies principally in the speeding up of phase 2 by utilizing additional criteria to curtail the search procedure. Transformations such as that employed in Ex. 3 may also speed up the rate of convergence by grouping the set of links forming the minimum circuit toward the start of the table of ranked links.

VI. EXTENSION TO RELATED NETWORK OPTIMIZATION PROBLEMS

The algorithm can be generalized to handle a much broader class of network optimization problems than the classical form of the traveling-salesman problem which has been treated here. The consideration of a

generalized network (instead of a simplex with symmetrical links) and the consideration of minimum paths (as well as minimum circuits) meeting specified conditions are suggested.

V. REFERENCES

1. J. T. Robacker. Some Experiments on the Traveling-Salesman Problem. The RAND Corporation: Research Memorandum RM-1521 (1955)

2. G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a Large Scale Traveling-Salesman Problem. Operations Research 2: 393-410 (1954)

3. L. L. Barachet. Graphic Solution of the Traveling-Salesman Problem. Operations Research 5: 841-845 (1957)

APPENDIX

DATA OF EX. 3

Table A-1

TRANSFORMED ROAD DISTANCES, $D(i_1,i_2)$, BETWEEN TEN CITIES

2	3	4	5	6	7	8	9	10	i_2 / i_1
998	112	1058	1619	714	0	1827	191	20	1
	880	0	2309	1630	1420	2161	969	548	2
		1120	2169	82	190	2349	755	450	3
			2245	1850	1484	1979	889	610	4
				2707	1987	0	826	1473	5
					296	2929	1293	1110	6
						2187	567	422	7
							1048	1549	8
								59	9

Table A-2

LINK VALUES OF TABLE A-1 ARRANGED IN RANK (NON-DECREASING) ORDER

Rank k	Link indices i_1	i_2	Link value D	Rank k	Link indices i_1	i_2	Link value D	Rank k	Link indices i_1	i_2	Link value D
1	1	7	0	16	1	6	714	31	8	10	1549
2	2	4	0	17	3	9	755	32	1	5	1619
3	5	8	0	18	5	9	826	33	2	6	1630
4	1	10	20	19	2	3	880	34	1	8	1827
5	9	10	59	20	4	9	889	35	4	6	1850
6	3	6	82	21	2	9	969	36	4	8	1979
7	1	3	112	22	1	2	998	37	5	7	1987
8	3	7	190	23	8	9	1048	38	2	8	2161
9	1	9	191	24	1	4	1058	39	3	5	2169
10	6	7	296	25	6	10	1110	40	7	8	2187
11	7	10	422	26	3	4	1120	41	4	5	2245
12	3	10	450	27	6	9	1293	42	2	5	2309
13	2	10	548	28	2	7	1420	43	3	8	2349
14	7	9	567	29	5	10	1473	44	5	6	2707
15	4	10	610	30	4	7	1484	45	6	8	2929

Table A-3

ROAD DISTANCES IN MILES, $S(i_1,i_2)$, BETWEEN TEN CITIES. EACH VALUE ON THE DIAGONAL MARGIN IS THE MINIMUM OF THE CORRESPONDING ROW AND COLUMN (I. E., $\min_j S(i,j)$)

	2	3	4	5	6	7	8	9	10	i_2 / i_1
215	690	327	720	1228	669	215	1332	350	235	1
167		687	167	1549	1103	901	1475	715	475	2
	327		807	1559	409	366	1649	688	506	3
		167		1517	1213	933	1384	675	506	4
			622		1869	1412	622	871	1165	5
				409		460	1980	998	877	6
					215		1512	538	436	7
						622		982	1203	8
							294		294	9
								235		