Industrial and Systems Engineering
Report Series No. J-78-11
-November, 1977

On the Detection of Negative Cycles in a Graph

by

Dennis P. Shea *

Stuart Jay Deutsch **

and

John J. Jarvis **

* Dennis P. Shea
Center for Naval Analyses
1401 Wilson Boulevard
Arlington, Virginia 22209

** Stuart Jay Deutsch and John J. Jarvis
School of Industrial Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

This work was performed under Grant No. 75NI-99-0091 from the National Institute of Law Enforcement and Criminal Justice. Points of view or opinions stated in this document are those of the authors and do not necessarily represent the official position or policies of the United States Department of Justice.

Abstract

This paper compares various algorithms for the detection and subsequent tracing of negative cycles in a graph. Both direct search and shortest path methods are examined. An experimental design is employed to evaluate the effects of algorithm, nodes, density, and arc distribution in detecting negative cycles on random networks. Extensive analysis is performed on 1) the computational time to detect a negative cycle and 2) the sum of the arc costs around the cycle. A third response variable, a quality index, defined as the absolute value of the sum of the arc costs around the cycle divided by the computational time to detect and trace the cycle, serves as an additional means of comparing the efficiency of the algorithms.

Introduction

Two important approaches to the solution of single commodity network flow problems are (1) primal approaches and (2) dual approaches. Dual approaches rely on the identification of shortest paths in a network; whereas, primal approaches depend on the detection of negative cycles in a network.

A number of papers [6,10] have compared various methods for locating shortest paths in a network. However, little work has been done on the determination of negative cycles. In the remainder of this paper we shall attempt to enumerate the important methods for locating negative cycles. We shall follow this enumeration with theoretical and computational comparisons of several of these approaches. Finally we shall conclude with a discussion of the relative difficulty of locating negative cycles as compared to that of locating shortest paths.

Methods for Identifying Negative Cycles

Literature on identifying negative cycles can generally be divided into two major classes:

1. Shortest path approaches
2. Direct search approaches.

Shortest path approaches attempt to find the shortest path in a network. If they are unsuccessful it is because a negative cycle exists. If the information in these shortest path algorithms is organized properly, it is usually a straight forward and simple matter to reconstruct a negative cycle when an indication of the existence of one is given.

Direct search methods rely on some specialized property which negative cycles possess. There methods exploit such a property to directly construct a negative cycle.

Shortest Path Approaches

Many of the important algorithms for detecting negative cycles in a graph concern locating shortest paths. Given a graph containing N nodes, a path between any pair of nodes may contain at most N-1 arcs. If a shortest path can be found which contains more than N-1 arcs, then an arc must be repeated and a negative cycle formed.

One of the first computationally efficient schemes for solving this problem was presented by Bellman [2]. Bellman proposed a dynamic programming algorithm for the problem when the associated arc distances are non-negative.

By the principle of optimality, π_j (the optimal distance from node 1 to node j) must satisfy the nonlinear system of equations.

$$\pi_j = \min_{i \neq j} [c_{ij} + \pi_i] \quad j = 2, 3, \ldots, N$$

$$\pi_1 = 0$$

Bellman suggested the following iterative algorithm

$$\pi_j^{(o)} = c_{1j} \quad j = 1, 2, \ldots, N$$

$$\pi_j^{(k+1)} = \min_{i \neq j} (c_{ij} + \pi_i^{(k)}) \quad j = 2, 3, \ldots, N$$

$$\pi_1^{(k+1)} = 0$$

$$k = 0, 1, 2, \ldots$$

$\pi_j^{(k)}$ represents the minimum distance for a path passing through at most k intermediate nodes. Termination with the optimal solution occurs when $\pi_j^{(k)} = \pi_j^{(k-1)}$, j = 1, 2, ..., N. Convergence is guaranteed in no more than N-2, i.e. k = N-2, iterations since no path contains more than N-1 arcs.

Ford and Fulkerson [9] presented a similar algorithm which extended to the more general problem where some c_{ij} are negative. When the procedure is applied to a problem with some negative distances c_{ij}, either convergence will occur on or before the (N-1)st iteration, indicating no negative cycles exist and the solution is optimal; or a change in some π_j will occur on the Nth iteration, indicating the existence of a negative cycle.

By processing the nodes alternately forward and backwards, and minimizing only over nodes previously treated, Yen [12], [13], [14] has produced a dynamic programming algorithm which reduces the amount of computational effort to $N^3/2$. This number is half that required by the original dynamic programming algorithms of Ford-Fulkerson and Bellman. This algorithm also indicates a negative cycle when a node label (functional equation) changes value on the last iteration.

Dantzig proposed the first simplex algorithm for the shortest path problem [5]. Bennington [3] refined this algorithm to show that the set of basic feasible solutions could be restricted to the set of arborescences centered at the origin. An arborescence centered at the origin consists of N-1 arcs which form a unique path from the origin to all remaining nodes without forming any cycles. Bennington also introduced a test for negative cycles into the simplex algorithm. Being a simplex based method, the Bennington algorithm is of exponential order in the worst case.

Direct Search Approaches

The only direct search method to locate negative cycles is due to Florian and Robert [7]. The algorithm is based on a property of negative partial sums of finite sequences. The only disadvantage of this algorithm can be shown to be its excessive computational upper bound (viz. expontial order).

Selection of Algorithms for Testing

We shall restrict our evaluation to the algorithms of Yen, Bennington, Florian, and Ford-Fulkerson. The first three are considered the most efficient algorithms from the fields of dynamic programming, linear programming, and direct search methods, respectively. The Ford-Fulkerson, also a dynamic programming based algorithm, has proven its reliability over time and would act as a "control algorithm" with which to compare the performance of the remaining three algorithms. The following section describes the algorithms in detail including a flowchart depicting the sequence of operations for each algorithm.

Flowcharts for the four algorithms are presented in the appendix.

Experimental Design

An experiment was designed to study the effect of several factors on the detection of negative cycles in a graph. Four independent variables were selected for the experiment. The factors consisted of (1) the specific algorithm used to locate the cycle, (2) the number of nodes in the network, (3) the density of the network, and (4) the distribution of arc cost.

A list processing approach was employed so the effect of network representation would be common to all four algorithms. Under list processing, only those arcs which exist in the network are stored in computer memory. Three lists are used to store the originating nodes i, the terminating nodes j, and the cost associated with arc (i,j) respectively. Two additional lists serve as pointers to indicate arcs entering or leaving a desired node.

A classical Fixed-Effects factorial design consisting of four factors, algorithm, nodes, density, and arc distribution was selected with each factor evaluated at (4,4,4,3) levels, respectively. Networks were generated and searched for negative cycles using all possible combinations of the following factors:

A. Type of algorithm--the algorithm used to detect and trace negative cycles (Yen, Bennington, Florian-Robert, Ford-Fulkerson).

B. Nodes--the number of nodes in the network (25,50,75,100).

C. Density--the ratio of the number of arcs in the network to the number of possible arcs if all pairs of nodes were connected (.05,.10,.15,.20).

D. Distribution of arc costs--the costs associated with the arcs were generated from a uniform distribution with mean = 0, over three ranges corresponding to a required variance. The intervals chosen were (-25,25), (-125,125), and (-250,250) corresponding to a variance of 208, 5208, and 20833, respectively.

The problem size was restricted to 100 nodes due to limitations in computer storage. The levels of density and the field. The specific levels of the three quantitative factors were chosen at the extremes and at intermediate levels so as to cover the entire range of interest. Three observations

were taken under each of the 192 conditions, making a total of 576 runs.

Selection of Response Variables

The initial response variable considered was the computational time required to locate and trace a negative cycle in a graph. Since each algorithm could not be expected to locate the identical negative cycle, an additional response variable of a quality index was also recorded. The quality index was defined as the ratio of the absolute value of the sum of the arc costs around the cycle to the time required to locate and trace the cycle.

Additional response variables of interest include the value of the cycle (the sum of arc costs around the cycle) and the number of nodes in the negative cycle.

Generating Random Networks

The generation of networks possessing various combinations of the foregoing factors was an integral element of the experimental design. In order to obtain a valid comparison of the four algorithms, it was essential to test them under identical conditions. That is, a network would be generated with a specified number of nodes, density, and distribution of arc costs and each algorithm would receive this network as input and proceed to search for negative cycles.

To guarantee a connected graph, i.e. every pair of nodes are joined by at least one chain, the first N-1 arcs were generated to form an arborescence in the network. (Note that the arborescence was required in the application of Bennington's algorithm.) Upon completion of the arborescence, additional arcs were generated to complete the required density.

Empirical Results

The experiment was performed by generating 144 networks (corresponding to three replications of the factorial design) and applying each of the four algorithms to locate a negative cycle.

The following response variables were recorded throughout the experiment: (1) computational time to detect and trace the negative cycle, (2) value of the cycle, (3) quality of the negative cycle, i.e. $|(2)|/(1)$, (4) the number of nodes in the cycle, and (5) the distinct nodes of the cycle. A summary of the quality indices and solution times is presented in Tables 1 and 2.

An analysis of variance (Table 3) performed on the quality index data revealed a significant effect (at the 0.05 level) due to algorithm, arc distribution, and the algorithm-arc distribution interaction. A similar analysis of variance executed on computational time (Table 4) disclosed an algorithm-node interaction as the single significant effect. This algorithm-node interaction reveals that as the number of nodes in the network is increased, its effect on computational time differs among the algorithms chosen to locate the cycle. The sum of squares due to all nonsignificant effects were pooled together with the sum of squares due to error and the analysis was reperformed. The results of the ANOVA conformed with the original analysis.

We can utilize the mean response data in explaining these effects. Table 1 shows that on the average, negative cycles obtained with the algorithm of Florian and Robert have an associated quality almost six times larger than any other algorithm. There is no discernible difference in quality over the levels of nodes or density. The mean response over arc distribution shows a perceptible increase in quality over successive levels of this variable.

The results in Table 2 question the superiority of the Florian and Robert algorithm. The mean computational time for this algorithm is exceeded only by the Ford-Fulkerson algorithm.. Since quality index is inversely proportional to computational time, these figures challenge the validity of the experiment.

In addition to the contrasting results over quality and computational time with the Florian and Robert algorithm, the lack of any significant effect due to nodes is also reason for concern. Since an iteration in each algorithm is a function of the number of nodes, we had originally expected an increase in the computational time (and therefore a decrease in quality index) with an increase in the number of nodes.

Examination of Table 5 which contains the mean responses over all combinations of algorithms and nodes offers some insight into the problem. The dependency on nodes is verified by the responses from three of the four algorithms. The algorithm of Florian and Robert is unaffected by the number of nodes in the network with the exception of the mean computational time for 50 node networks which is exceptionally high. This term is responsible for the significant algorithm-node interaction uncovered by the ANOVA in Table 4.

The algorithm of Florian and Robert was removed and an analysis of variance performed on the remaining data. The mean response data is presented Tables 6 and 7 while the analyses are shown in Tables 8 and 9.

The analysis on quality produced a significant effect due to all main factors: algorithm, nodes, density, and arc distribution. In addition, all interactions involving algorithms were found to be significant. The main effects can be verified by examining the mean responses in Table 6. The Yen algorithm produces negative cycles with the highest mean quality. The difference

Table 1. Mean Response--Quality Index

Algorithm		Nodes	
Yen	7189.8	25 Nodes	12945.8
Bennington	1517.5	50 Nodes	12965.6
Florian-Robert	42137.0	75 Nodes	13322.1
Ford-Fulkerson	880.0	100 Nodes	12491.0

Density		Arc Distribution	
.05	12529.6	(-25,25)	2923.4
.10	14000.7	(-125,125)	12231.8
.15	13563.7	(-250,250)	23638.0
.20	11630.5		

Table 2. Mean Response—Computational Time (Sec)

Algorithm		Nodes	
Yen	0.085	25 Nodes	0.101
Bennington	0.809	50 Nodes	1.134
Florian-Robert	0.873	75 Nodes	0.548
Ford-Fulkerson	1.277	100 Nodes	1.260

Density		Arc Distribution	
.05	1.335	(-25,25)	0.607
.10	0.601	(-125,125)	0.563
.15	0.498	(-250,250)	1.112
.20	0.609		

Table 3. Analysis of Variance--Quality

Source	Degrees of Freedom	Sum of Squares (1)	Mean Square (1)	F-ratio
Main Effects				
Algorithm	3	167.00	55.60	191.72*
Nodes	3	0.05	0.02	0.07
Density	3	0.49	0.16	0.55
Arc Distribution	2	41.30	20.60	71.03*
Interactions				
Algorithm-Nodes	9	1.25	0.14	0.48
Algorithm-Density	9	3.01	0.33	1.14
Algorithm-Arc Distribution	6	76.40	12.70	43.79*
Nodes-Density	9	3.54	0.39	1.34
Nodes-Arc Distribution	6	2.04	0.34	1.17
Density-Arc Distribution	6	0.41	0.07	0.24
Error	519	151.00	0.29	

*Significant at $\alpha = 0.05$ level

(1) Entries are in 10^9

Table 4. Analysis of Variance--Computational Time

Source	Degrees of Freedom	Sum of Squares	Mean Square	F-ratio
Main Effects				
Algorithm	3	106.30	35.40	1.56
Nodes	3	125.00	41.67	1.84
Density	3	64.30	21.43	0.95
Arc Distribution	2	35.70	17.85	0.79
Interactions				
Algorithm-Nodes	9	426.68	47.40	2.09*
Algorithm-Density	9	281.18	31.24	1.38
Algorithm-Arc Distribution	6	127.13	21.18	0.93
Nodes-Density	9	318.37	35.37	1.56
Nodes-Arc Distribution	6	115.59	19.26	0.85
Density-Arc Distribution	6	112.36	18.72	0.83
Error	519	11729.70	22.60	

*Significant at $\alpha = 0.05$ level

Table 5. Mean Response--Algorithm-Node Interaction

	Nodes	Quality	Computational Time (sec)
Yen	25	5423.8	.062
	50	8627.2	.089
	75	7674.4	.078
	100	7033.8	.108
Bennington	25	4153.1	.056
	50	1274.2	.325
	75	410.2	.798
	100	232.4	2.058
Florian-Robert	25	40150.7	.0050
	50	41141.5	3.4760
	75	44801.4	.0046
	100	42454.5	.0054
Ford-Fulkerson	25	2055.3	.2820
	50	819.3	.6440
	75	402.5	1.3130
	100	243.1	2.8680

Table 6. Mean Response--Quality-Excluding Florian and Robert's Algorithm

Algorithm		Nodes	
Yen	7189.8	25	3877.4
Bennington	1517.5	50	3573.6
Ford-Fulkerson	880.0	75	2829.1
		100	2503.1

Density		Arc Distribution	
.05	2013.0	(-25,25)	668.7
.10	3249.6	(-125,125)	3570.8
.15	3733.7	(-250,250)	5347.9
.20	3786.9		

Table 7. Mean Response--Computational Time-Excluding Florian and Robert's Algorithm

Algorithm		Nodes	
Yen	0.085	25	0.133
Bennington	0.809	50	0.353
Ford-Fulkerson	1.277	75	0.730
		100	1.678

Density		Arc Distribution	
.05	0.621	(-25,25)	0.728
.10	0.800	(-125,125)	0.750
.15	0.662	(-250,250)	0.694
.20	0.811		

Table 8. Analysis of Variance--Quality-Excluding Florian and Robert's Algorithm

Source	Degrees of Freedom	Sum of Squares (1)	Mean Square (1)	F-ratio
Main Effects				
Algorithm	2	34.70	17.30	113.80*
Nodes	3	1.32	0.44	2.89*
Density	3	2.20	0.73	4.82*
Arc Distribution	2	16.00	8.00	52.60*
Interactions				
Algorithm-Nodes	6	4.97	0.83	5.44*
Algorithm-Density	6	5.40	0.90	5.92*
Algorithm-Arc Distribution	4	11.40	2.85	18.70*
Nodes-Density	9	1.92	0.21	1.40
Nodes-Arc Distribution	6	0.68	0.11	0.74
Density-Arc Distribution	6	0.85	0.14	0.92
Error	384	58.49	0.15	

*Significant at $\alpha = 0.05$ level

(1) Entries are in 10^8

Table 9. Analysis of Variance--Computational Time-Excluding Florian and Robert's Algorithm

Source	Degrees of Freedom	Sum of Squares	Mean Square	F-ratio
Main Effects				
Algorithm	2	103.93	51.96	108.50*
Nodes	3	150.88	50.30	105.07*
Density	3	3.00	1.00	2.09
Arc Distribution	2	0.23	0.12	0.24
Interactions				
Algorithm-Nodes	6	75.19	12.53	26.17*
Algorithm-Density	6	16.84	2.81	5.87*
Algorithm-Arc Distribution	4	0.29	0.07	0.14
Nodes-Density	9	14.21	1.58	3.30*
Nodes-Arc Distribution	6	5.56	0.93	1.94
Density-Arc Distribution	6	1.80	0.30	0.63
Error	384	183.82	0.48	

*Significant at $\alpha = 0.05$ level

in mean quality between the Yen and Ford-Fulkerson algorithms seems large enough to create an effect due to algorithm. This data shows an increasing complexity of network problems with an increase in nodes. Increasing density supplies the networks with additional arcs and leads to negative cycles of high quality. The data also indicates that distributing the arc costs over wide intervals centered at 0 will result in negative cycles with high quality.

The analysis on computational time produced a significant effect due to algorithms and nodes. The algorithm-nodes, algorithm-density and nodes-density interactions were also considered to be significant. Table 7 substantiates the effects due to algorithm and nodes. Surprisingly, there was no effect on computational time due to density. Networks with high densities contain a relatively large number of arcs and would seem to be more complex than low density networks. Obviously, the analysis has demonstrated otherwise. The lack of an arc distribution effect is understandable since the variance of arc costs would not seem to be a computational factor.

Selecting the Most Effective Algorithm

It seems reasonable to conclude that on networks for which the theoretical upper bound is not attained, the Florian and Robert algorithm will produce the superior response in terms of both quality and computational time. For the user who desires a negative cycle with high quality, the Yen algorithm is favored under most conditions.

Comparison of Negative Cycles to Shortest Paths

To complete the research on randomly generated networks, we attempted to gain a measure of the relative difficulty involved in locating negative

cycles compared to locating shortest paths. Since three of the four algorithms used to detect negative cycles are modifications of shortest path algorithms, the question arose as to which was the most difficult problem.

The results show that locating negative cycles requires less computational effort than does locating shortest paths. The answer lies in the terminating criterion of the two problems.

Recall that during the kth iteration, the Ford-Fulkerson algorithm determines the optimal path from the source node to all nodes whose shortest path consists of no more than k arcs. During the same iteration, the Yen algorithm determines the shortest path from the origin to all nodes in the kth block of increasing and decreasing sequences. At least one node must become permanently labeled during each iteration in either algorithm. Therefore, both algorithms terminate with the optimal shortest paths whenever the node labels from the successive iterations are identical for all nodes. Convergence is guaranteed in no more than N-1 arcs or blocks. Termination with a negative cycle may occur as early as the 2nd iteration.

Termination of the simplex algorithm with the shortest paths or a negative cycle is a function of the initial basic feasible solution and the manner in which nonbasic arcs are chosen to enter the basis. There is no evidence to indicate that this algorithm detects negative cycles "faster" than it locates shortest paths. Empirical results have shown the Bennington algorithm to be the most severely affected algorithm in terms of computational time in locating negative cycles as the percentage of negative arcs in the network decreased.

Application to the Minimal Cost Flow Problem

We have thus for reported computational experience in locating negative cycles on randomly generated networks. Once a negative cycle was found, the

network was discarded and a new network created with different negative value will require the tracing of fewer negative cycles in an iteration of Klein's [11] algorithm than will an algorithm which does not concern itself with the value of a negative cycle. Although this is not a statistical test of hypotheses, the result is encouraging and warrants further research.

The computational experience with the Florian and Robert algorithm was disappointing. A number of networks arose which approached the theoretical upper bound of this algorithm.

Any difficulties encountered with this algorithm on random networks are accentuated in the minimal cost flow problem since the entire problem consists of examining similar networks. That is, if while solving a minimal cost flow problem using Klein's algorithm a network arises which causes Florian and Roberts's algorithm to approach its computational upper bound, then the following network is also likely to present problems for the algorithm. The rationale is that once a negative cycle is identified on a marginal cost network, flow is sent around the cycle and the new marginal cost network is formed. The new marginal cost network will differ from the previous one in at most k ($2 \leq k \leq N$) arcs, i.e. those arcs corresponding to the negative cycle. Therefore, Klein's algorithm will retain a so called "bad" network until the minimal cost flow is obtained. This contrasts our experimental "runs" in which a network was discarded after a single negative cycle was located. This evaluation may not be a true indication of how well an algorithm will perform when applied to locate negative cycles in a practical problem. Therefore, as a final evaluation, the four algorithms were employed to locate negative cycles in Klein's algorithm for minimal cost flow problems.

Over the set of problems solved, Bennington's algorithm produced the fastest average solution time. This can be attributed to the "restart" procedure

which distinguishes the Bennington algorithm from the remaining algorithms which store no information from the previous cycles and must be "restarted" from scratch. By utilizing the "restart" technique, the initial arborescence in the Bennington algorithm need be created only once. After a negative cycle is located, an arborescence for the new marginal cost network is obtained from a modified version of the previous arborescence along with the reverse arcs from the negative cycle.

The computational experience showed that both the Yen and Ford-Fulkerson algorithms were required, on the average, to locate fewer negative cycles in a minimal cost flow problem than were the Bennington or Florian-Robert algorithms. An analysis also showed that the negative cycles detected by the Yen and Ford-Fulkerson algorithms were, on the average, "more negative" than the cycles located by the other two algorithms.

Conclusion

The empirical results show the Florian and Robert direct search method to be superior to the remaining algorithms in locating negative cycles on random networks. The application of the algorithm in Klein's minimal cost flow algorithm produces disappointing results. When a marginal cost network presents a difficulty to the direct search method, the next marginal cost network (differing from the previous network in k arcs corresponding to the negative cycle) will also cause difficulties to Florian and Robert's algorithm.

Contrasting Florian and Robert's algorithm was the Bennington algorithm. The true capability of the latter was not realized until the algorithm was applied to the minimal cost flow problem.

The Bennington algorithm is the only algorithm which retains information from previous cycles. This information was of no value to us in the experiment in which the network was discarded once a negative cycle was located. An

algorithm like the Bennington algorithm, which possesses a "restart" procedure, is intuitively appealing and its relative worth was presented in the minimal cost flow problem in which it produced the lowest average time per problem solved.

The Yen algorithm was probably the overall most efficient of the four algorithms tested. It returned a negative cycle of reasonably high quality in a fairly low average computational time over the entire experiment. The algorithm was not severely affected by any particular combination of the independent factors comprising a network. The algorithm performed favorably in the minimal cost flow problem as well as in locating the shortest paths when negative costs were allowed. Yen's algorithm has the lowest theoretical upper bound on the number of required computations and this fact coupled with the empirical evidence recommends the algorithm as the most dependable. The only drawback would appear to be in programming the algorithm for efficient use on the computer.

The Ford-Fulkerson algorithm was used primarily as a "control" algorithm against which to compare the remaining algorithms. The Ford-Fulkerson algorithm has withstood the test of time and produced respectable results in both the experiment and the minimal cost flow problem. The algorithm, while not providing the fastest computational time, did produce, on the average, the "most negative" cycle. Unlike the Yen algorithm, its strongest attribute is that the Ford-Fulkerson algorithm is easily programmed for implementation on the computer.

BIBLIOGRAPHY

1. Bazaraa, M. S. and R. W. Langley, "A Dual Shortest Path Algorithm," SIAM Applied Mathematics, Vol. 26, No. 3, (1974), pp. 496-501.

2. Bellman, Richard, "On a Routing Problem," Quarterly Applied Mathematics, Vol. 16, No. 1 (1958), pp. 87-90.

3. Bennington, G. E., "A Simplex Algorithm for the Shortest Chain Problem," Report No. 72, North Carolina State University at Raleigh, 1971.

4. Bennington, G. E., "An Efficient Minimal Cost Flow Algorithm," Management Science, Vol. 19, No. 9 (1973), pp. 1042-1051

5. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, Princeton, N.J., 1963.

6. Dreyfus, S. E., "An Appraisal of Some Shortest-Path Algorithms," Operations Research, Vol. 17 (1969), pp. 395-412.

7. Florian, M. And P. Robert, "A Direct Search Method to Locate Negative Cycles in a Graph," Management Science, Vol. 17, No. 5 (1971), pp. 307-310.

8. Florian, M. and P. Robert, Rejoinder: Direct Search Method for Finding Negative Cycles," Management Science, Vol. 19, No. 3 (1972), pp. 335-336.

9. Ford, L. R. and D. R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, 1962.

10. Golden, Bruce, "Shortest-Path Algorithms: A Comparison," Operations Research, Vol. 24, No. 6 (1976), pp. 1164-1168.

11. Klein, M., "A Primal Method for Minimal Cost Flows with Applications to the Assignment and Transportation Problems," Management Science, Vol. 14, No. 3 (1967), pp. 205-220.

12. Yen, J.Y., "An Algorithm for Finding Shortest Routes from all Source Nodes to a Given Destination in General Networks," Quarterly Applied Mathematics, Vol. 27, No. 4 (1970), pp. 526-530.

13. Yen, J.Y., "A Modified 1/2 N^3-steps Algorithm for Detecting Negative Loops or Finding all Shortest Routes from a Fixed Origin in N-node General Networks, Paper presented at the TIMS Tenth American Meeting, Atlanta, Georgia, Oct. 1-3, 1969.

14. Yen, J.Y., "On the Efficiency of a Direct Search Method to Locate Negative Cycles in a Network," Management Science, Vol. 19, No. 3 (1972), pp. 333-335.

Appendix

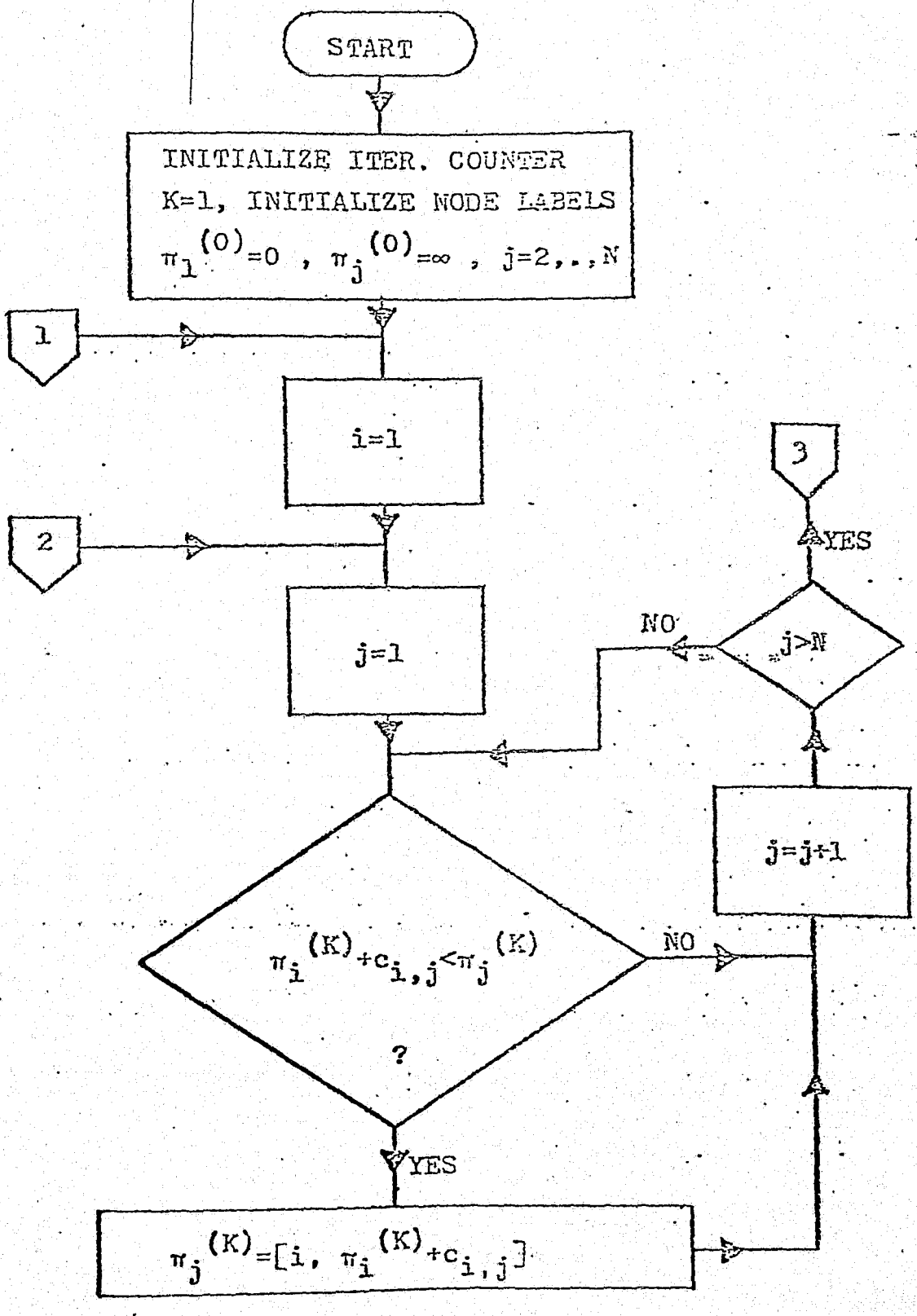

Figure A-1. Flowchart of the Ford-Fulkerson Algorithm

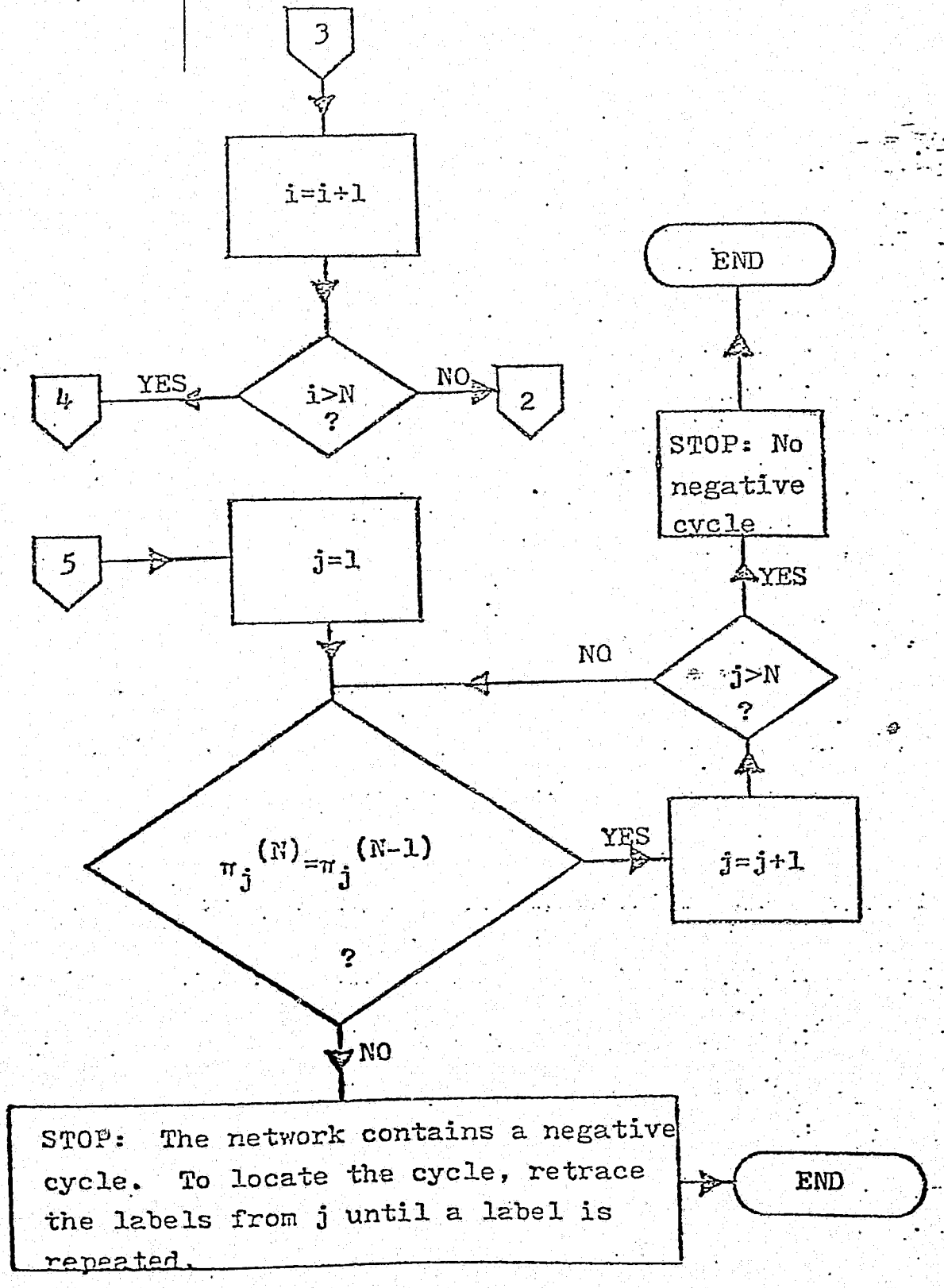

Figure A-1. (continued)

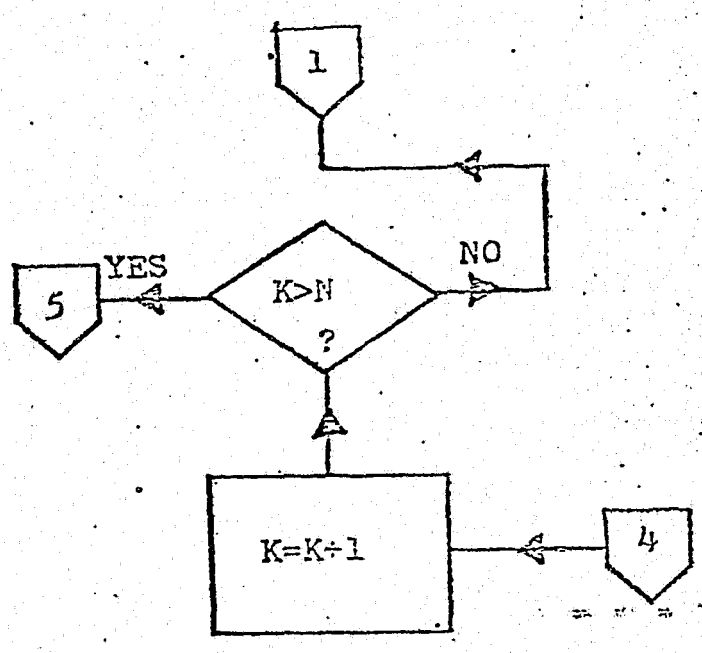

Figure A-1. (concluded)

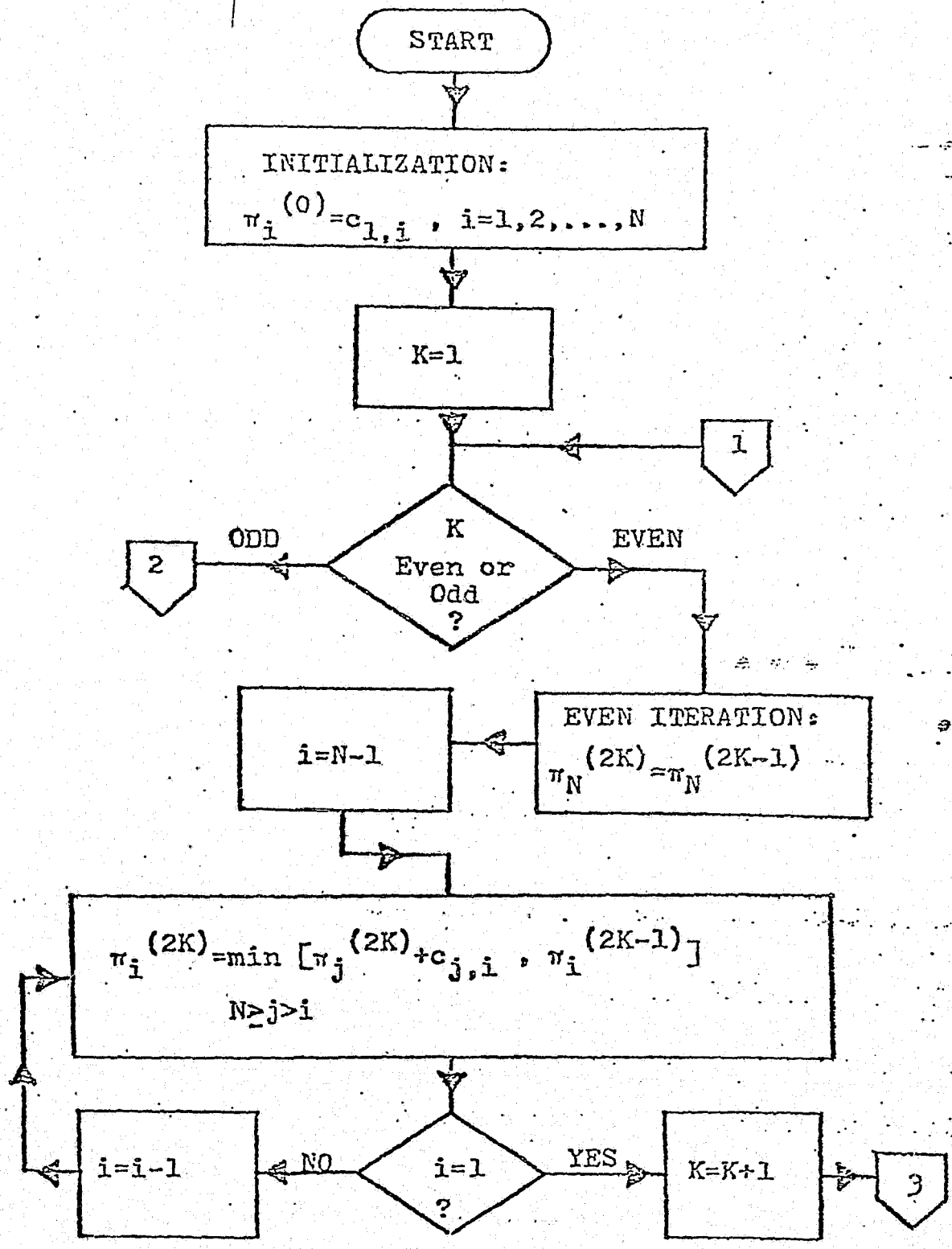

Figure A-2. Flowchart of the Yen Algorithm

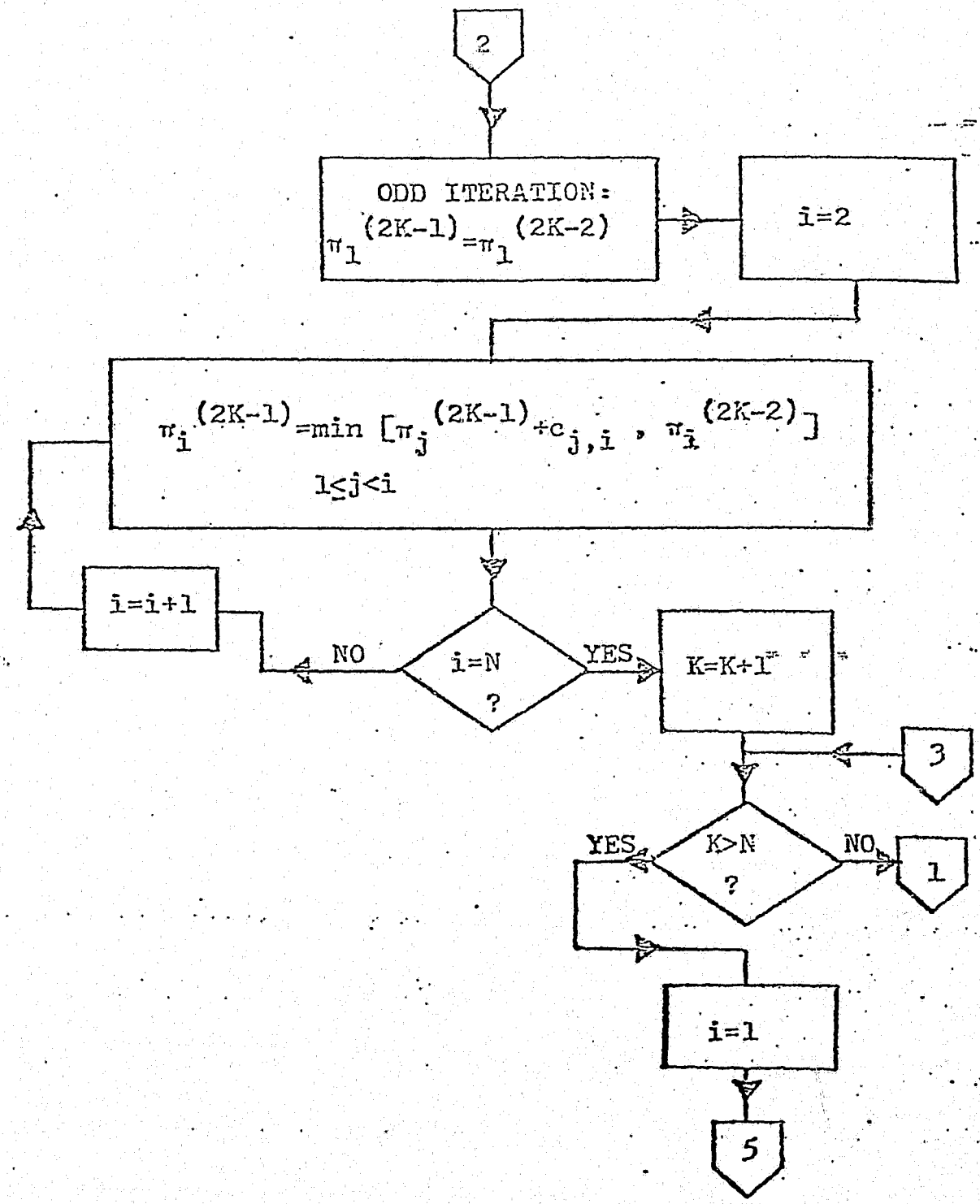

Figure A-2. (continued)

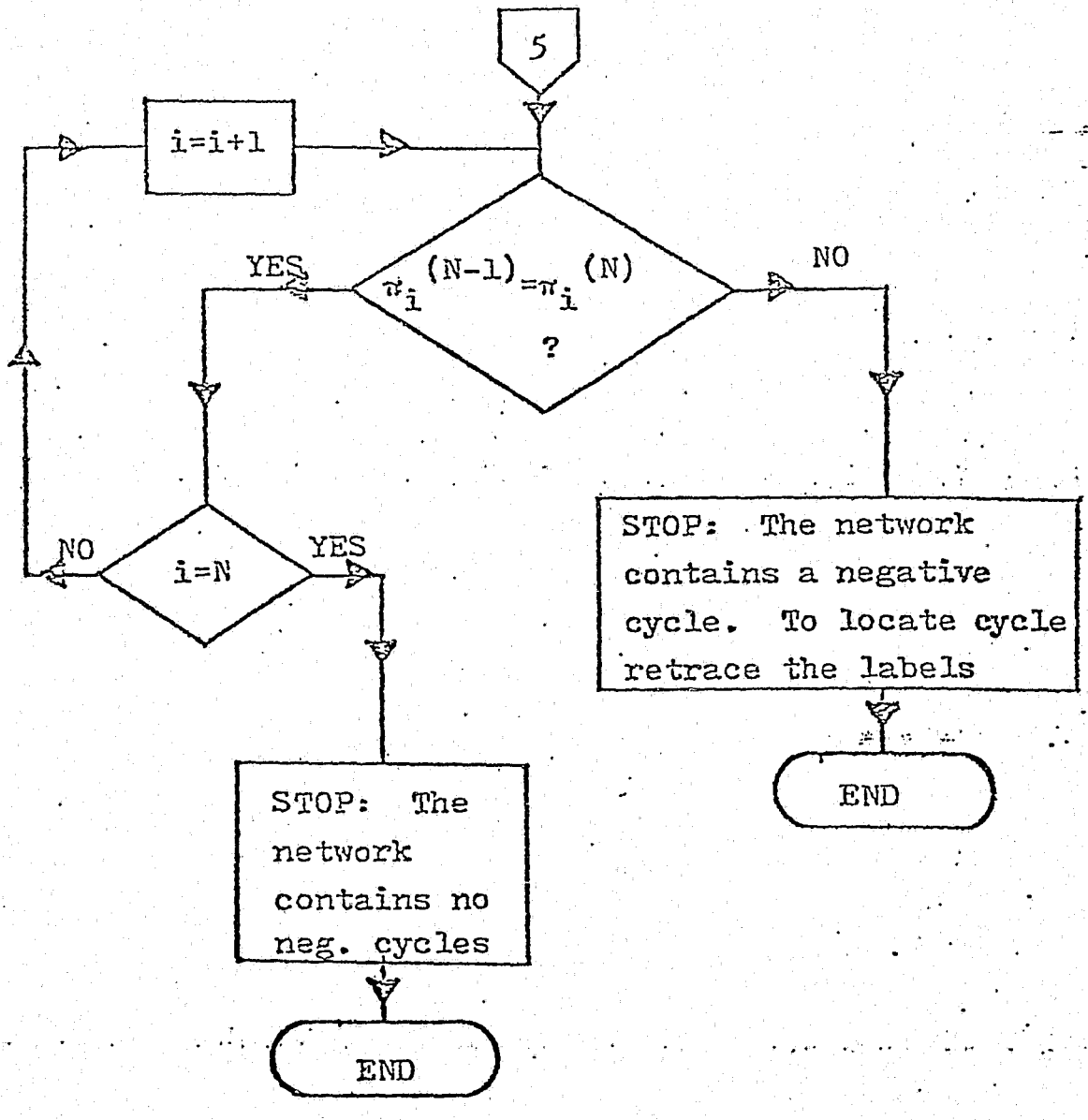

Figure A-2. (concluded)

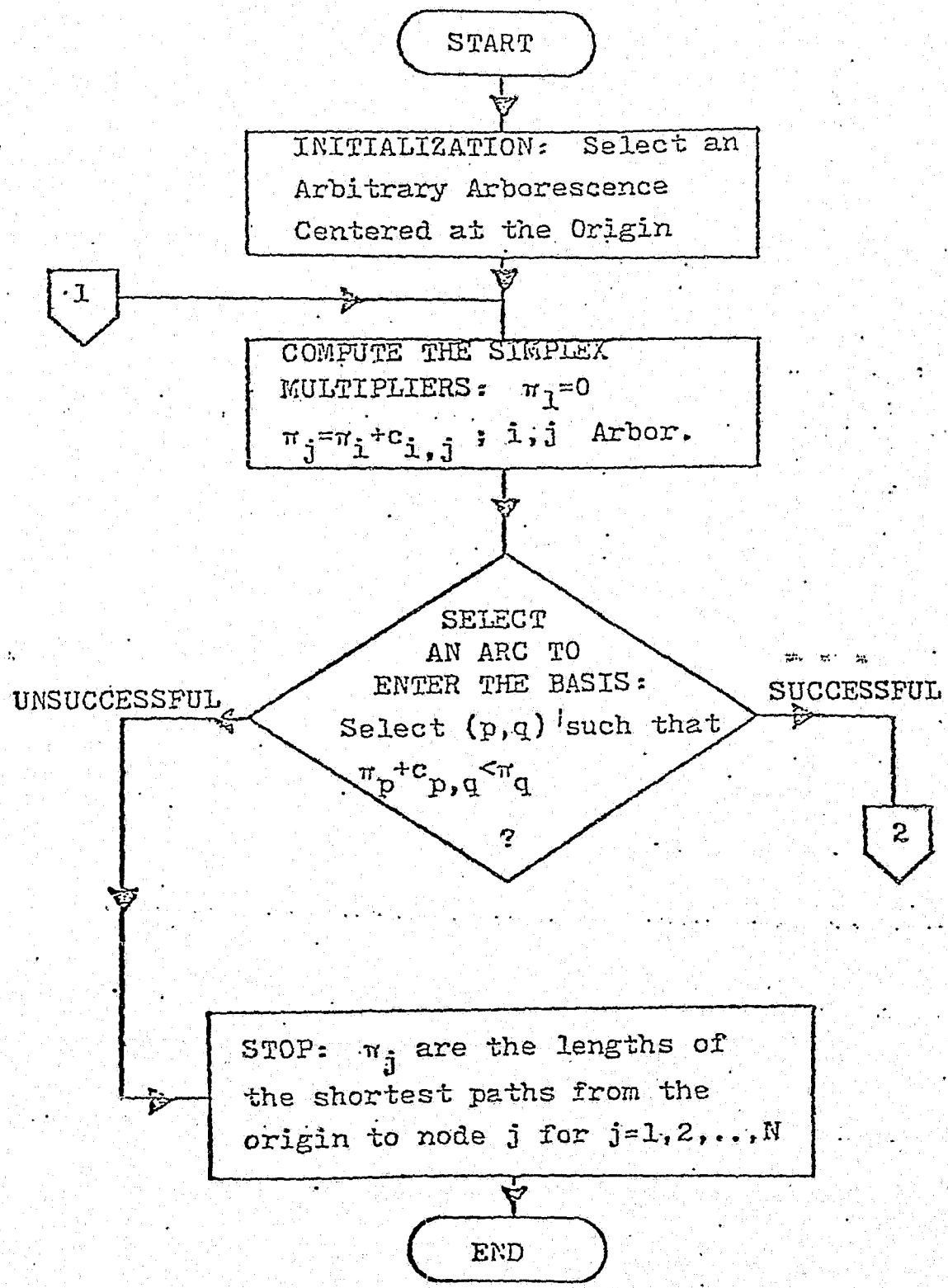

Figure A-3. Flowchart of the Bennington Algorithm

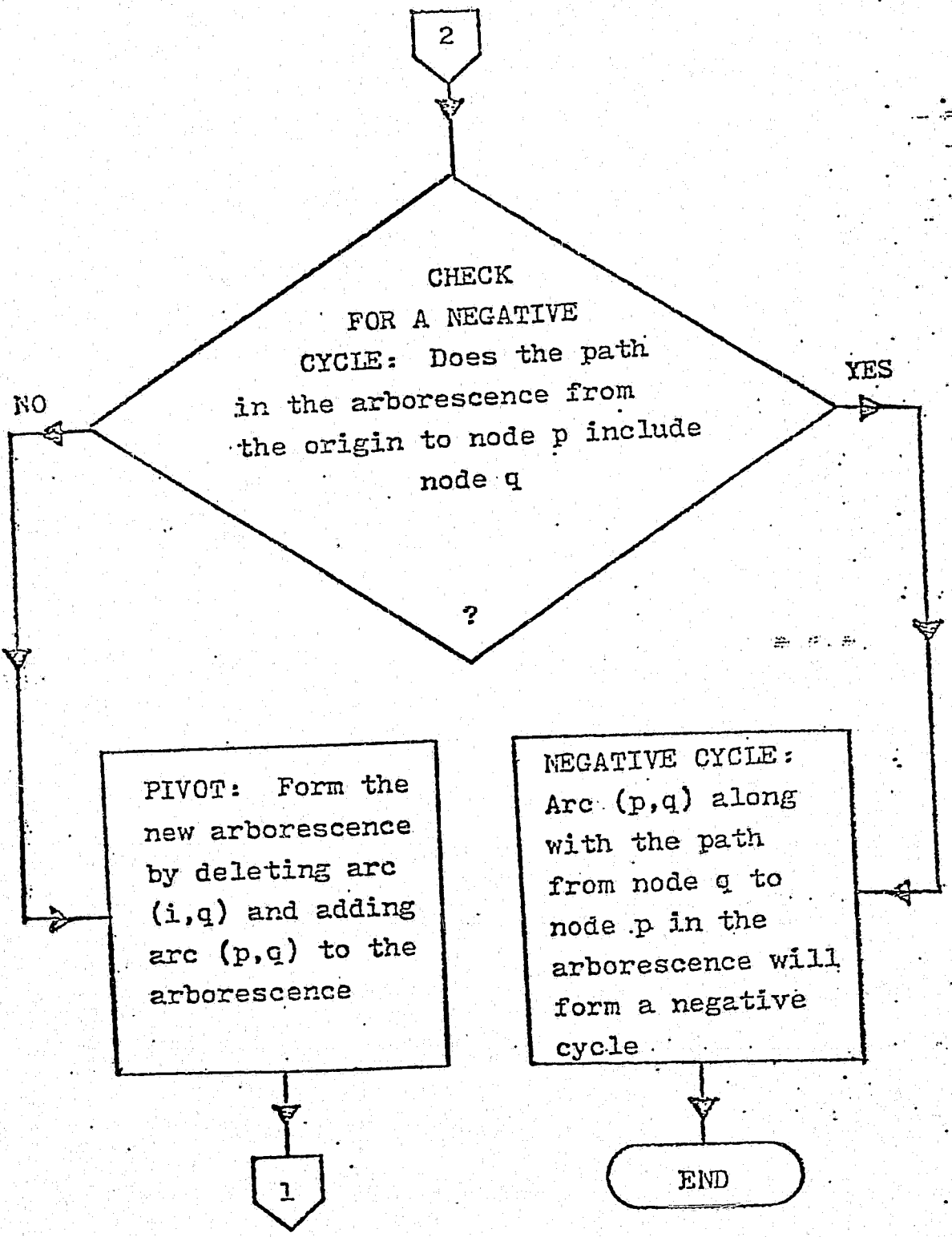

Figure A-3. (concluded)

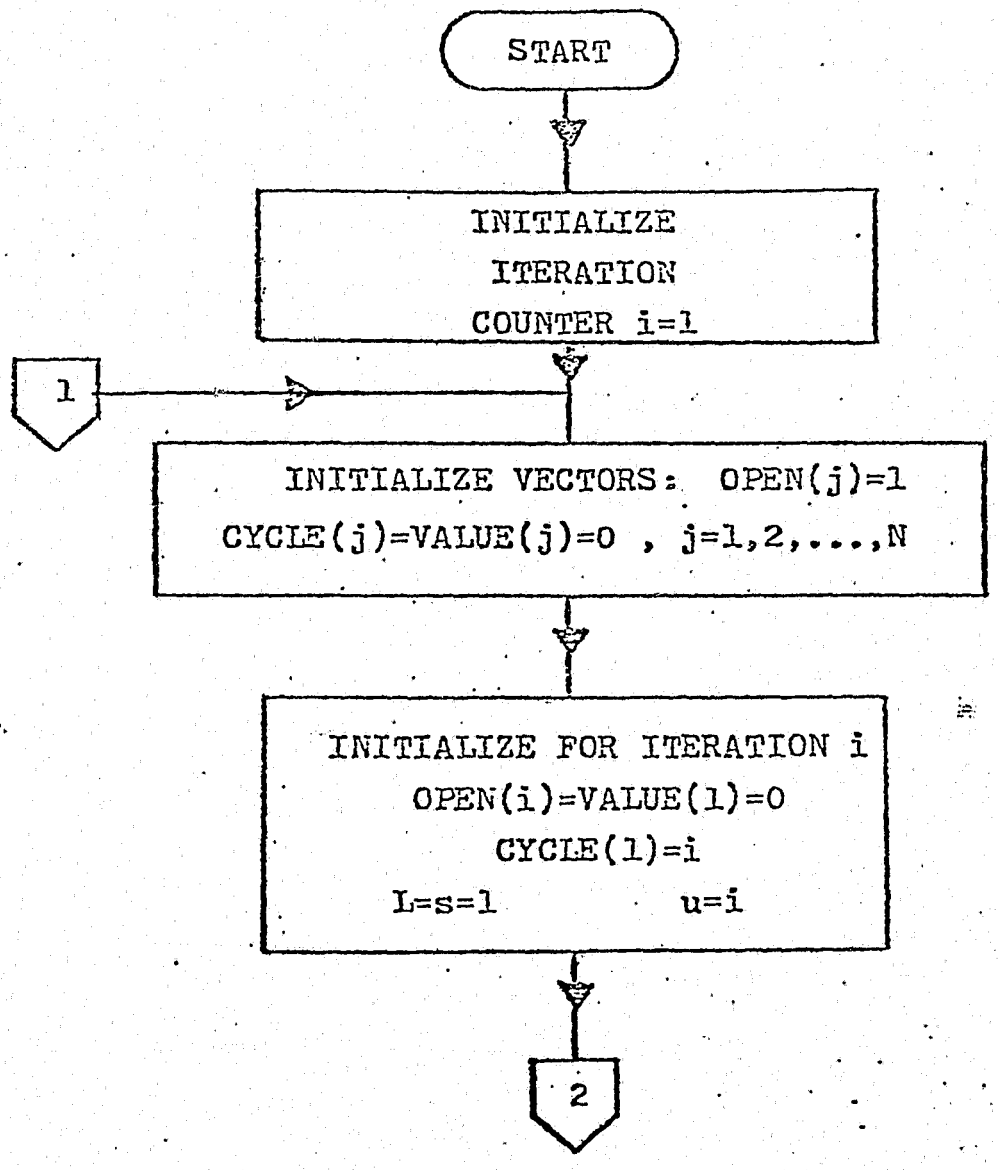

Figure A-4. Flowchart of the Florian and Robert Algorithm

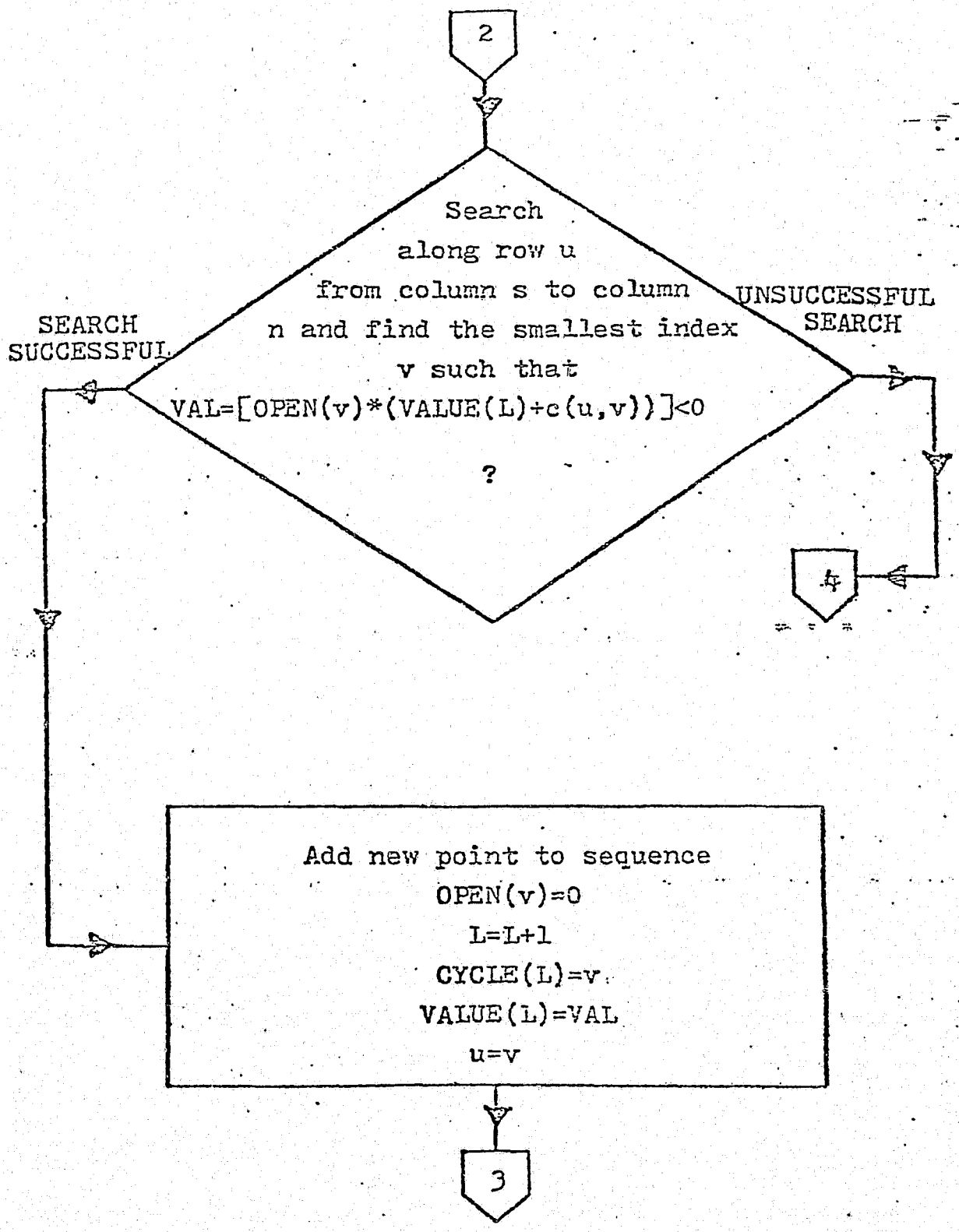

Figure A-4. (continued)

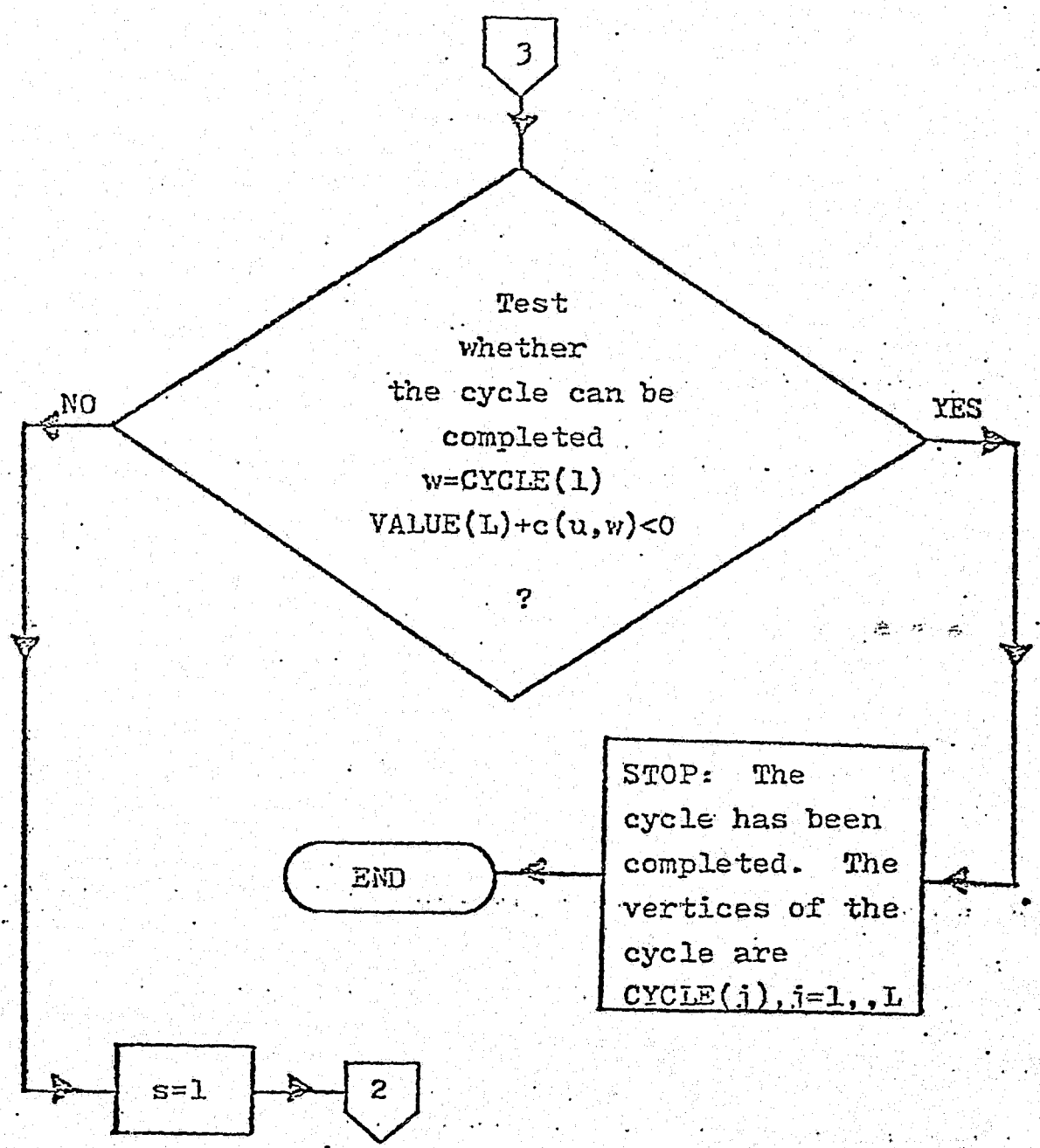

Figure A-4. (continued)

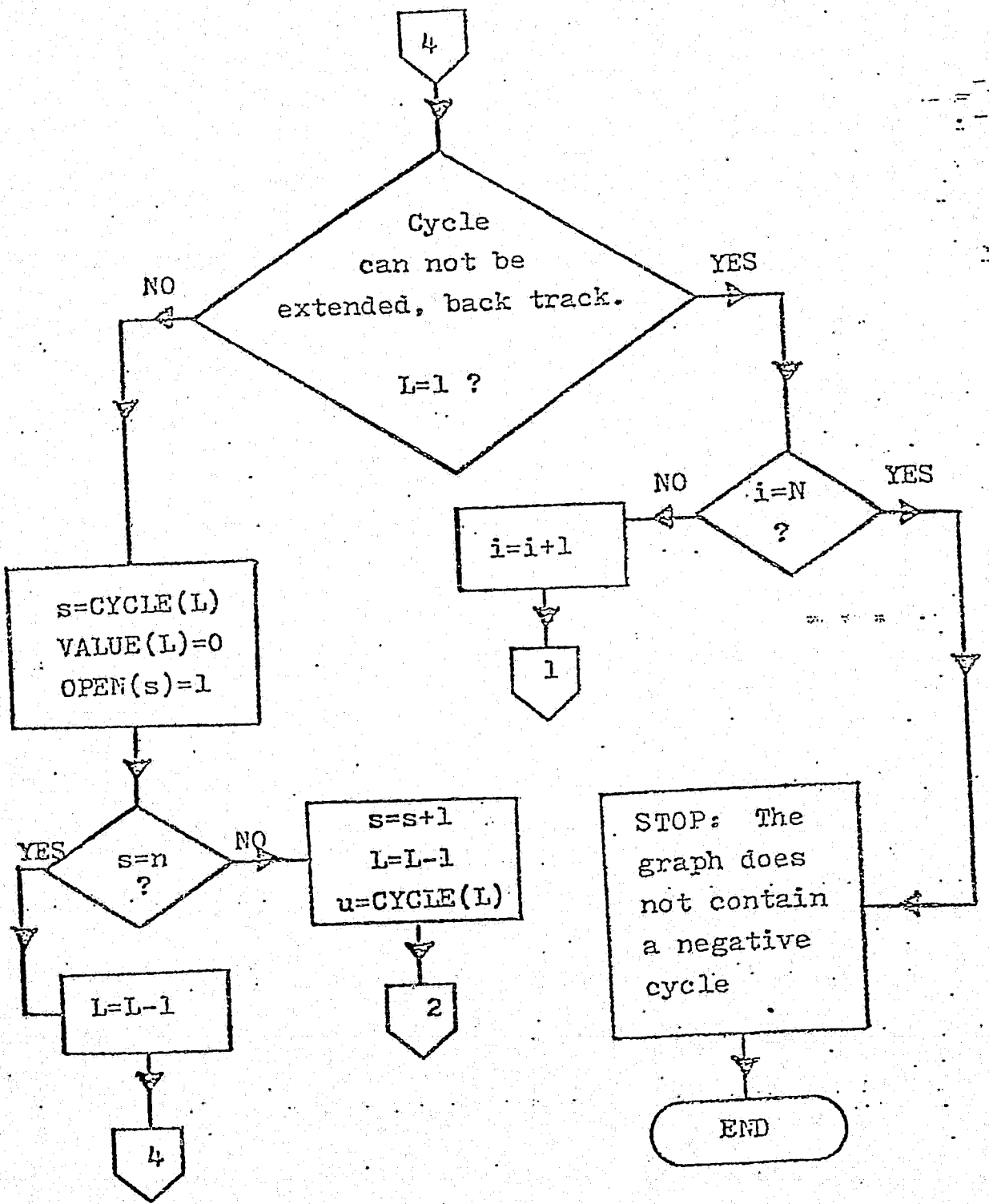

Figure A-4. (concluded)

END