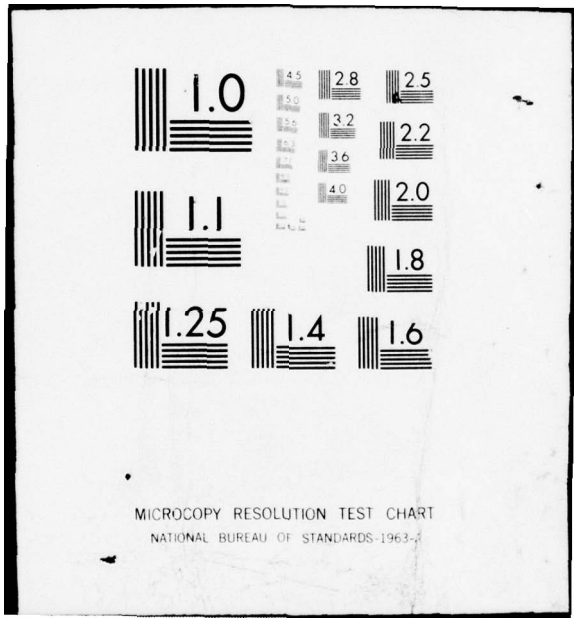
AD-A041 009 DARCOM INTERN TRAINING CENTER TEXARKANA TEX F/G 12/2
EFFICIENCY OF THE MISRA-FAIR ALGORITHM FOR THE SOLUTION OF THE --ETC(U)
APR 76 W E SMYTHE

UNCLASSIFIED DARCOM-ITC-02-08-76-021 NL

| OF |
AD
A041009

END

DATE
FILMED
7-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(14) DARCOM-
ITC-02-08-76-021

(2) NW

(6)

EFFICIENCY OF THE MISRA-FAIR ALGORITHM FOR THE SOLUTION OF THE TRAVELING-SALESMAN PROBLEM

(10) Walter E. Smythe
Maintainability Graduate Engineering Program
USA DARCOM Intern Training Center
Red River Army Depot
Texarkana, Texas 75501

(11) April 1976 (12) 36p.

(9) Final Report.

Prepared for

MAINTAINABILITY GRADUATE ENGINEERING PROGRAM
AND TEXAS A&M UNIVERSITY GRADUATE CENTER
USA DARCOM Intern Training Center - USALMC
Red River Army Depot
Texarkana, Texas 75501

392756

FORWARD

The research discussed in this report was accomplished as part of the Maintenance Effectiveness Engineering Graduate Program conducted jointly by the DARCOM Intern Training Center and Texas A&M University. As such, the ideas, concepts and results herein presented are those of the author and do not necessarily reflect approval or acceptance by the Department of the Army.

This report has been reviewed and is approved for release. For further information on this project contact Dr. Ronald C. Higgins, Intern Training Center, Red River Army Depot, Texarkana, Texas 75501.

APPROVED:

DR. RONALD C. HIGGINS, Chief
Department of Maintenance Effectiveness Engineering

FOR THE COMMANDER:

JAMES L. ARNETT, Director
Intern Training Center

ABSTRACT

Research Performed by Walter E. Smythe

Under the Supervision of Dr. Ram B. Misra

The efficiency of the Misra-Fair algorithm for solution of the Traveling-Salesman problem is investigated. A FORTRAN language computer program is written for the Misra-Fair algorithm, Little et al.'s Branch-and-Bound algorithm and the Closest-Unvisited-City algorithm. The accuracy and speed of solution of the program using the Misra-Fair algorithm is compared to the accuracy and speed of solution of the programs using the Branch-and-Bound algorithm and the Closest-Unvisited-City algorithm. Preliminary computational results and suggestions for improving the computer program are given.

ACKNOWLEDGMENTS

The author wishes to express his gratitude to Dr. Ram B. Misra for his supervision, suggestions and guidance during the progress of this research. I also wish to thank my wife, Jan, for her help with typing, for her support and especially for her patience.

During the course of this work, the author was employed by the U. S. Army as a career intern in the DARCOM Maintainability Engineering Graduate Program.

The ideas, concepts, and results herein presented are those of the author, and do not necessarily reflect approval or acceptance by the Department of the Army.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Many situations in government and industry require the optimal ordering of a number of tasks. Sequencing jobs on a machine to minimize set-up time and ordering stops on a supply or maintenance route for minimum distance traveled are two practical examples of the classical Traveling-Salesman problem. A statement of this problem is as follows:

> A salesman must visit each of n cities once and only once and return to the starting city by the shortest possible route. The distances between all cities are known and the starting city is unspecified. The distances between cities need not be symmetrical.

The solution to the Traveling-Salesman problem is normally found using such techniques as the Closest-Unvisited-City algorithm (2), the Branch-and-Bound algorithm (12) and more recently the Held-Karp algorithm (7,8). Dr. Ram B. Misra and David F. Fair propose a simplified algorithm for the solution of this problem in an unpublished paper entitled "A New Algorithm for Solving the Traveling-Salesman Problem (14)." The Misra-Fair algorithm is simple and straightforward, making application to problems almost intuitive. The algorithm quickly gives an initial solution.

This solution is not guaranteed to be optimal, but provides a quick method for obtaining a reasonable first guess. The initial solution and the lower bound which is found in the first step of the algorithm combine to give a range which contains the optimal solution. The algorithm checks the initial solution for optimality. Problems with non-optimal initial solutions are then optimized.

Presently, there is no data available giving an estimate of the percent of time that the Misra-Fair algorithm yeilds an initial solution that is optimal. No data is available giving an estimate of the relative error of any non-optimal initial solution found using the Misra-Fair algorithm. Because the Misra-Fair algorithm has not been coded for computer usage, no comparisons of speed of solution exist between this algorithm and currently used techniques for solving the Traveling-Salesman problem.

This report describes the procedure used to estimate the efficiency of the Misra-Fair algorithm. This procedure consists of writing three computer programs to solve the Traveling-Salesman problem; one using the Misra-Fair algorithm, one using the Closest-Unvisited-City algorithm and one using the Branch-and-Bound algorithm of Little et al. The computer programs are to be used to solve a number of problems. The results are to be used to determine such things as relative error in the initial solution, relative speed of solution and the general relative efficiency of the Misra-Fair algorithm.

Chapter 2 surveys the work that has been done on this problem. The Misra-Fair algorithm is described in Chapter 3. The computer program that uses the algorithm is also described. The Closest-Unvisited-City algorithm and the computer program that uses it are described in Chapter 4. Chapter 5 describes the Branch-and-Bound algorithm and the computer program that uses it. Problems and results are reported in Chapter 6. Conclusions and recommendations for further study are presented in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

Solution techniques for the Traveling-Salesman problem are divided into three classes by Bellmore and Nemhauser (1). The tour-to-tour improvement class includes iterative schemes as developed by Lin (10) and Reiter and Sherman (15). Dynamic programing (3,6) and "branch and bound" algorithms such as Little's et al. (12) and Hatfield and Pierce (5) fall into the tour building class. The third class of techniques, called subtour elimination, includes integer programing (13).

The Misra-Fair algorithm (14) is a tour building scheme. This algorithm begins just like the Branch-and-Bound algorithm of Little et al.; but immediately after finding the baseline value, the algorithm departs from the Branch-and-Bound procedure. The remainder of the Misra-Fair algorithm consists of heuristic rules for selecting path-segments in the tour. There are several procedures documented in the literature that contain heuristics, such as Krolak et al. (9) and Lin and Kerningham(11). These techniques are different and much more complicated than those used in the Misra-Fair algorithm.

Held and Karp have developed an algorithm for solving symmetric Traveling-Salesman problems that gives good computational results for large problems (7,8). However, this

algorithm seems complex and contains a "branch and bound" procedure together with an ascent method for calculation of lower bounds (4). Helbig, Hansen and Krarup report an improved Held and Karp algorithm with solution times twenty-five times faster (4). Both algorithms are complex, however.

The Closest-Unvisited-City algorithm is one of the most straightforward algorithms for solving the Traveling-Salesman problem (2). It is another tour building algorithm.

The Misra-Fair algorithm is simple and easy to understand. It is possibly the simplest and the easiest algorithm to understand of any of the algorithms for the solution of the Traveling-Salesman problem, with the exception of the Closest-Unvisited-City algorithm. If the Misra-Fair algorithm produces solution times comparable with the Branch-and-Bound algorithm, it would appear to be the one to use because of its simplicity and ease of understanding.

CHAPTER 3

THE MISRA-FAIR ALGORITHM

The Misra-Fair algorithm for solving the Traveling-Salesman problem consists of four steps. Step 1 is to obtain a Lower Bound for the solution to the problem. This is accomplished by forming a square data matrix with the city to city cost data. Then the smallest cost in each row and column of the data matrix is subtracted from the row or column in which it appears. The sum of the subtracted values is the Lower Bound for the problem. This starting procedure is the same as for the Branch-and-Bound algorithm of Little et al. (12).

Step 2 is to form a Solution Table that ranks in ascending order the costs of the various path-segments. A path-segment is the path from one city to another; the path from city A to city B is path-segment A-B. This ranking allows the consideration of the least expensive path-segments first.

Step 3 is the Solution Routine. This step uses the Solution Table made up of the ranked path-segments, starting with the zero cost path-segments and adding others to the table as needed. The Solution Rules and the Logic Rules listed below allow path-segments from a Solution Talbe to be either included in the solution or eliminated from the solution. The Solution Rules are (14):

1. When only one destination is available from an origin or there is a uniquely occuring destination, the corresponding path-segment must be entered into the solution.
2. Each path-segment in solution has a unique origin and destination.
3. A feasible solution must include all of the destinations. Any path-segment which returns a solution to its origin without visiting all of the cities must be disregarded.

The Logic Rules are (14):

1. If when a path-segment is arbitrarily eliminated from consideration and no feasible solution exists, then that path-segment must be part of the feasible solution, if one exists.
2. If a new set of path-segments is entered into a Solution Table, at least one of the members of that set of path-segments must be entered into the new solution.
3. When a path-segment is aribitrarly entered into solution to break an impasse, that is a failure of the Solution Rules to determine a solution, and no feasible solution is found to exist, then that path-segment may be eliminated from consideration.

The Logic Rules are used only when the Soution Rules fail to reach a solution or fail to determine that a solution cannot be obtained using the current Solution Table. The

Solution Routine finds an initial solution which may or may not be the optimal solution.

Step 4 checks the initial solution for optimality. This is done by examining the Solution Tables containing the various combinations of path-segments with costs that sum to a value less than the difference between the Lower Bound, L, and the initial solution, S. The examination of the Solution Tables is done using the Solution Routine. Any solution found that has a cost less than the initial solution is taken as the new initial solution. The algorithm then checks this solution for optimality. This process continues until the optimal solution is found.

To demonstrate the use of the Misra-Fair algorithm, the problem shown in Table 1 is given. Application of Step 1 of

CITY	1	2	3	4	5	6
1	-	7	5	5	5	3
2	10	-	5	6	3	2
3	11	9	-	4	4	3
4	11	1	0	-	4	0
5	9	0	1	0	-	5
6	4	6	1	5	8	-

TABLE 1. EXAMPLE PROBLEM DATA MATRIX

Misra-Fair algorithm to this example problem gives a Lower Bound of 13. The resulting reduced data matrix is shown in Table 2. Table 3 shows the path-segments of the reduced data matrix ranked according to cost, which is Step 2 of the

CITY	1	2	3	4	5	6
1	-	4	2	2	1	0
2	5	-	3	4	0	0
3	5	6	-	1	0	0
4	8	1	0	-	3	0
5	6	0	1	0	-	5
6	0	5	0	4	6	-

TABLE 2. REDUCED DATA MATRIX

the Misra-Fair algorithm. The entries in Table 3 are the destinations from each origin ranked from smallest to largest cost. For example, the path-segment from city 1 to city 6 has a cost of zero; therefore, a 6 is entered in the table

		COST							
		0	1	2	3	4	5	6	8
ORIGIN	1	6	5	3,4		2			
	2	5,6			3	4	1		
	3	5,6	4				1	2	
	4	3,6	2		5				1
	5	2,4	3				6	1	
	6	1,3				4	2	5	

TABLE 3. RANKED PATH-SEGMENTS

under the zero cost column for origin 1. Next the Solution Rules and the Logic Rules of Step 3 are used on Solution Tables formed from the ranked path-segments table. Table 4 shows the first Solution Table formed, which uses only path-segments with cost of zero. The Solution Rules quickly

indicatd for the example that this Solution Table cannot yeild a feasible solution because path-segments 2-5 and 3-5 cannot exist in the same solution. Circled destinations indicate path-segments that are included in the solution by

ORIGIN	DESTINATION
1	(6)
2	5,ø
3	5,ø
4	3,ø
5	2,4
6	1,3

TABLE 4. SOLUTION TABLE WITH COST 0

the Solution Rules, (i.e. 1-6). Destinations that have been eliminated from consideration by the Solution Rules are slashed, (i.e. 2-6, 3-6, 4-6). The Solution Routine continues adding destinations with higher costs to the Solution Table until the Solution Rules and Logic Rules can obtain a solution. For this example, the Solution Table of Table 5 which includes destinations with cost zero and one yeilds the solution 1-5-3-4-2-6-1. The cost of this solution is 17, the Lower Bound of 13 plus 4. Step 4 applies the Solution Routine to the Solution Table shown in Table 6 giving the optimal solution of 1-4-3-5-2-6-1 with a cost of 13 + 2 = 15. This solution is known to be optimal because the Solution Table contained destinations with costs of up to two and the solution of 15 is only 2 greater than the Lower Bound.

ORIGIN	DESTINATION
1	⑤, ∅
2	∅, ⑥
3	④, ∅, ∅
4	②, ∅, ∅
5	∅, ③, ∅
6	①, ∅

TABLE 5. SOLUTION TABLE WITH COST 0 & 1

ORIGIN	DESTINATION
1	∅, ④, ∅, ∅
2	∅, ⑥
3	∅, ⑤, ∅
4	∅, ③, ∅
5	②, ∅
6	①, ∅

TABLE 6. SOLUTION TABLE WITH COST 0, 1, & 2

The Misra-Fair algorithm is implemented for computer usage in a FORTRAN language computer program. Figure 1 gives a generalized flow chart of the program. Step 1 of the algorithm is accomplished by inputting the cost data matrix from data cards. The Lower Bound is found by SUBROUTINE REDUCE. A vector, COST, is established containing in ascending order the various costs of the path-segments. Starting at point 1000, the Solution Table is set up and the Solution Routine is performed. The program loops through this portion until the Solution Rules and the Logic Rules find an initial

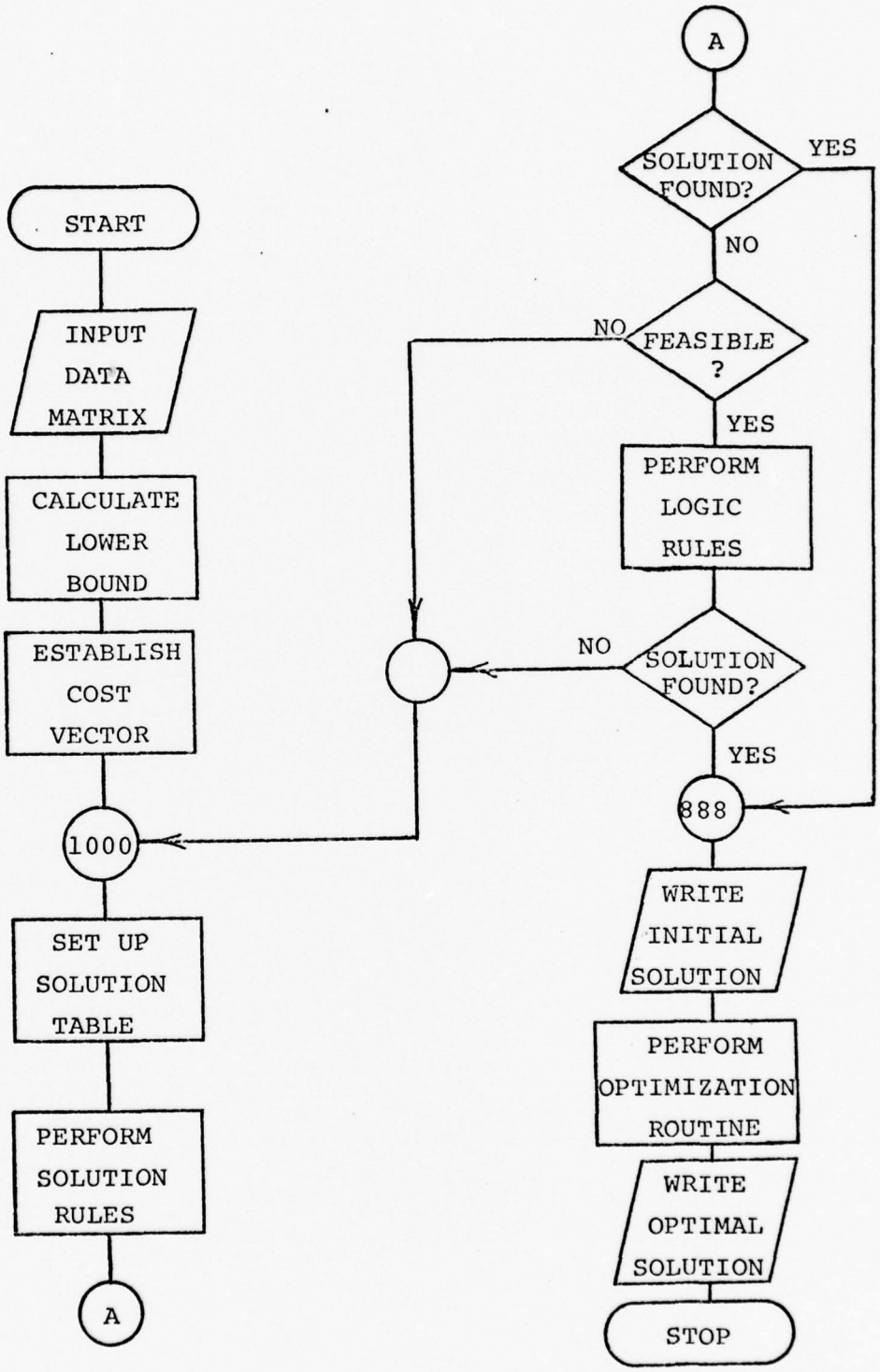


FIGURE 1. MISRA-FAIR PROGRAM FLOWCHART

solution. Each time the progran loops back to point 1000, a set of path-segments with the next highest cost is included in the Solution Table. After an initial solution is found, the program goes to point 888 and starts the optimization routine of Step 4. When the optimal solution is found, the program ends.

CHAPTER 4

CLOSEST-UNVISITED-CITY ALGORITHM

The Closest-Unvisited-City algorithm (2) is a very simple method for finding a solution to the Traveling-Salesman problem. The algorithm arrives at a solution that is not guaranteed to be optimal. For this reason, the Closest-Unvisited-City algorithm has been chosen to compare with the initial solution found by the Misra-Fair algorithm.

In the Closest-Unvisited-City algorithm, a starting city is selected. The next city in the solution tour is chosen by selecting the city closest to the starting city. This process is repeated until a complete tour is chosen. Cities already visited are not considered as the algorithm progresses. If ties for the closest city exist at any point in the algorithm, the cities that are tied are included in the solution tour one at a time. The algorithm is completed for each "tied" city.

A FORTRAN computer program was written using the Closest-Unvisited-City algorithm. Figure 2 shows a generalized flowchart fo the program. It can be seen in Figure 2 that the program finds a solution for each of the n cities in an n city problem that are used as starting cities. More than one solution for each starting city is found if a tie for the closest unvisited city occurs from the starting city. Only the best solution is outputted by the program.

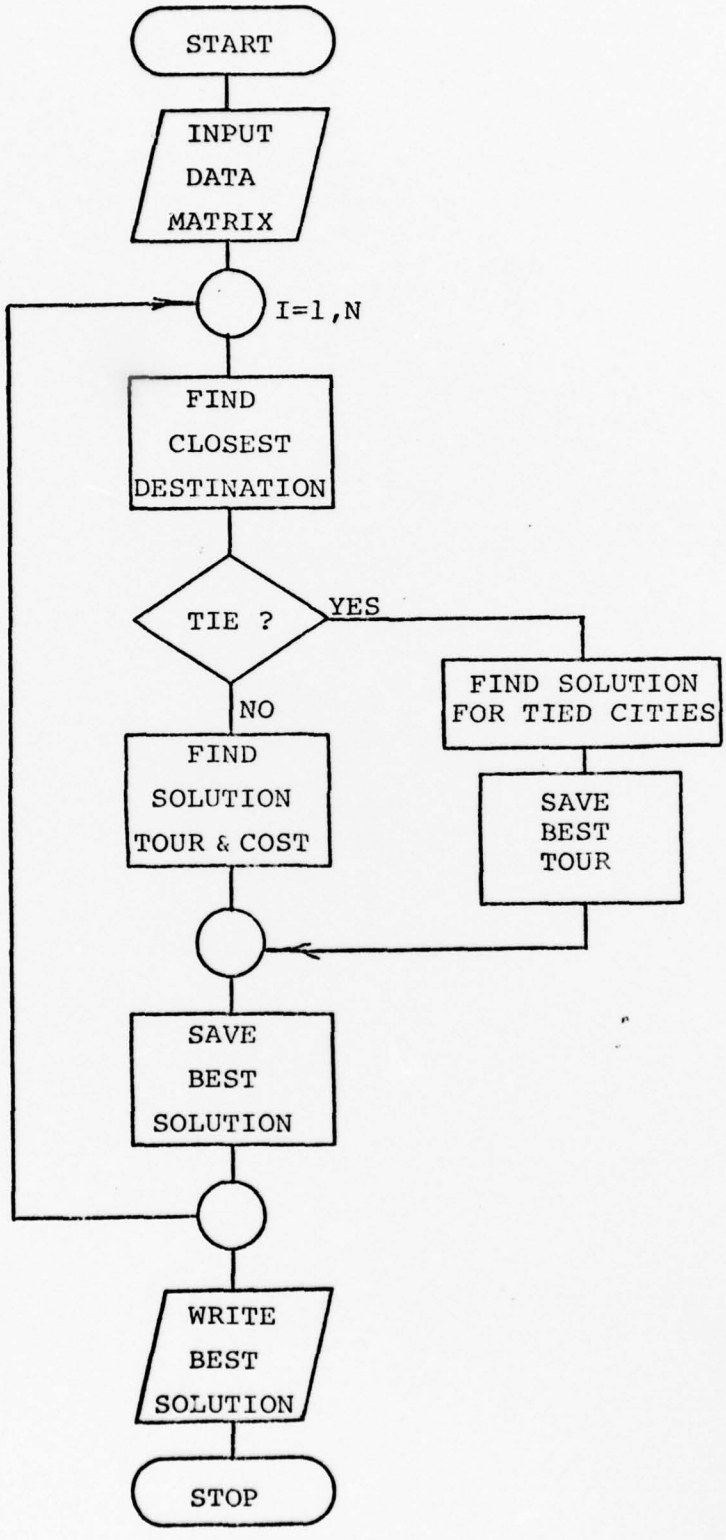


FIGURE 2. CLOSEST-UNVISITED-CITY PROGRAM FLOWCHART

CHAPTER 5

THE BRANCH-AND-BOUND ALGORITHM

The Branch-and-Bound algorithm developed by Little et al. (12) has been commonly published in scheduling theory textbooks as one of the better means of solving the Traveling-Salesman problem. This has been the reasoning for using the Branch-and-Bound algorithm as the solution method to compare the Misra-Fair algorithm against.

The Branch-and-Bound algorithm is so named because of its two main features. The bounding process is as described in Chapter 3 under Step 1 of the Misra-Fair algorithm. The cost data matrix is used and the reduced cost matrix is calculated as before by subtracting the smallest entry in each row and column from that row and column. The sum of the subtracted values gives a Lower Bound for the problem.

The branching process is a means by which the algorithm divides a problem into two new problems. The branch is made on the path-segment (path from city A to city B is path-segment A-B as before) that will cause the largest increase in the original problem's Lower Bound if that path-segment is excluded from solution. This path-segment is found by looking at all of the path-segments with zero cost in the reduced cost matrix. When this path-segment is found, one new problem is formed by excluding the path-segment from solution.

Another problem is formed by including the path-segment in the solution. This problem is one city smaller than the original problem. This process continues until a complete solution tour is obtained.

Application of the Branch-and-Bound algorithm is best illustrated by an example problem. Table 7 shows the data matrix of the example problem used in Chapter 3. The dash

CITY	1	2	3	4	5	6
1	-	7	5	5	5	3
2	10	-	5	6	3	2
3	11	9	-	4	4	3
4	11	1	0	-	4	0
5	9	0	1	0	-	5
6	4	6	1	5	8	-

TABLE 7. EXAMPLE PROBLEM DATA MATRIX

(-) entries in Table 7 indicate path-segments with infinite costs. The original cost matrix is reduced, giving the Lower Bound for the problem. Table 8 shows the reduced cost matrix for the example. The Lower Bound for the example problem is 13 as before. Figure 3 shows the tree-graph representation of the first two branching points for the problem, path-gegment 6-1 and path-segment 5-4. The Lower Bound of that solution path is the number outside the circle of the last node. This branching and bounding process continues until an optimal solution is obtained. As before, the optimal solution is the tour 1-4-3-5-2-6-1 with a cost of 15.

CITY	1	2	3	4	5	6
1	-	4	2	2	1	0
2	5	-	3	4	0	0
3	5	6	-	1	0	0
4	8	1	0	-	3	0
5	6	0	1	0	-	5
6	0	5	0	4	6	-

TABLE 8. EXAMPLE PROBLEM REDUCED DATA MATRIX

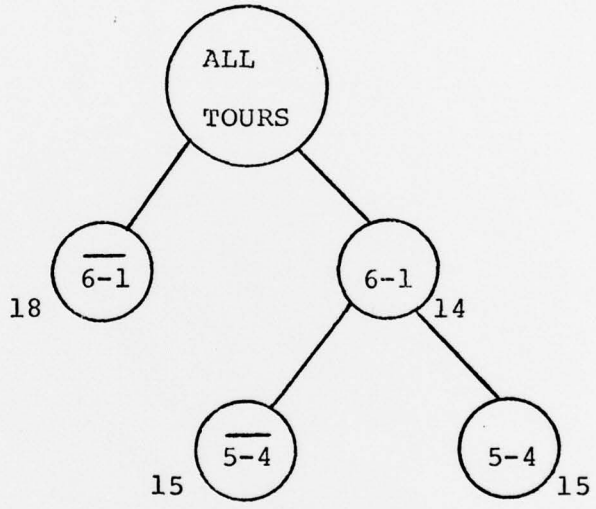


FIGURE 3. EXAMPLE PROBLEM TREE GRAPH

Figure 4 is a generalized flowchart of the FORTRAN language computer program that uses the Branch-and-Bound algorithm. The problem cost matrix is input and reduced. The path-segment to branch on is picked and the branch is made to the excluded path-segment problem. The branch including the path-segment is then made. If that forms a 2 x 2 problem, the cost is checked to see if the completed solution is

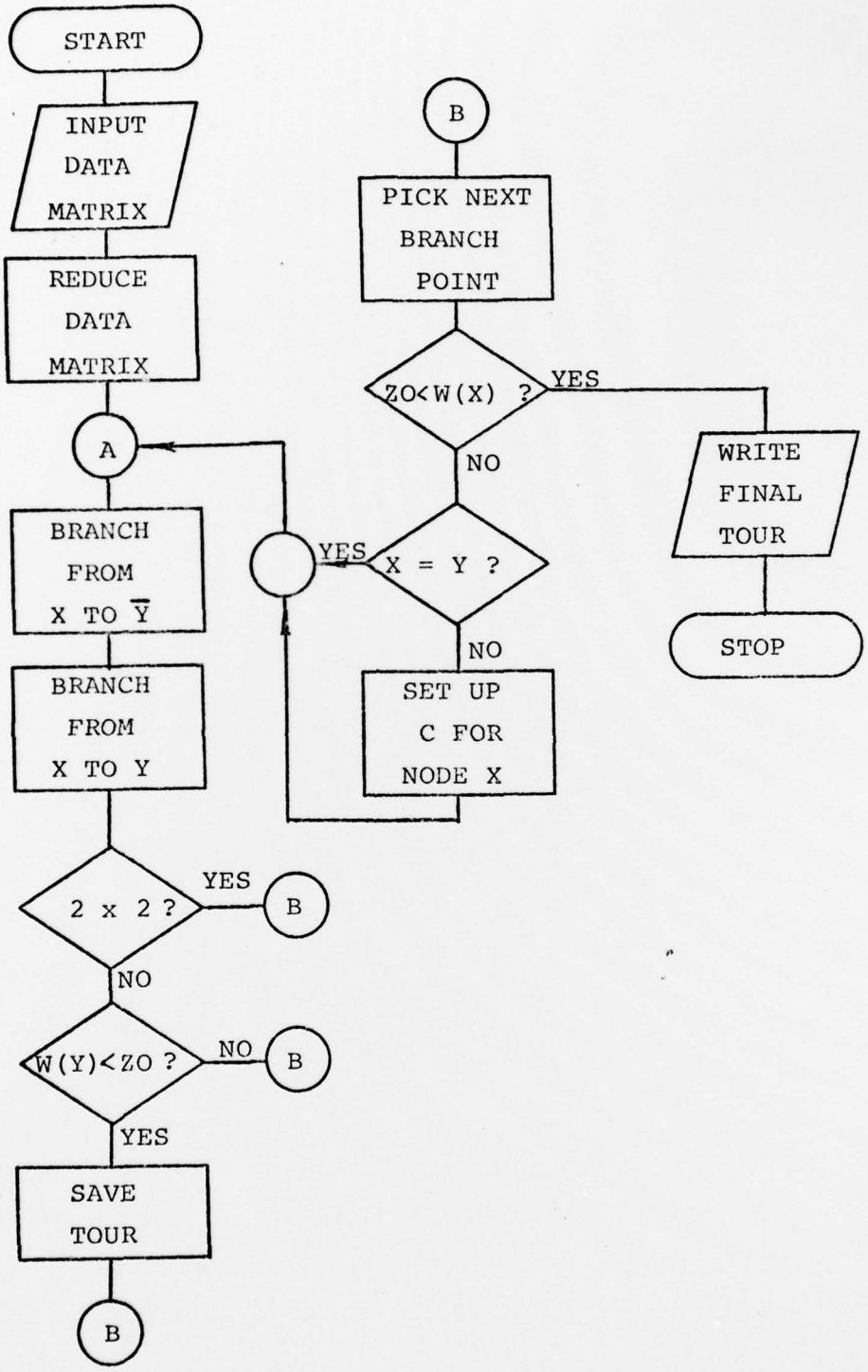


FIGURE 4. BRANCH-AND-BOUND PROGRAM FLOWCHART

better than any previous solution. Then the next branch point is picked. This process continues until any branch point picked is greater than or equal to the cost of the best finished solution.

CHAPTER 6

COMPUTATIONAL RESULTS

The three programs described in the preceding chapters are being used to solve a number of Traveling-Salesman problems. The programs are being run on the Amdahl 470 computer at Texas A&M University, College Station, Texas. The WATFIV FORTRAN compiler is being used. This compiler prints the execution time required for each run of one of the programs. From these times, an estimate of the speed of solution of the Misra-Fair algorithm can be made. This estimate can then be compared with a similar estimate of the speed of solution of the Closest-Unvisited-City algorithm and the Branch-and-Bound algorithm. The results of this comparison will give a good indication of the speed of solution of the Misra-Fair algorithm. The relative error of the initial solution found by the Misra-Fair algorithm can also be found.

To compare the Misra-Fair algorithm with the Closest-Unvisited-City algorithm, the program using the Misra-Fair algorithm was modified to stop execution as soon as the initial solution was found. Twenty problems were solved using both the Misra-Fair algorithm and the Closest-Unvisited-City algorithm. Table 9 shows the execution times, initial solutions, and actual optimal solutions for the ten problems with six cities that were solved. The average execution time for

	CLOSEST-UNVISITED-CITY ALGORITHM		MISRA-FAIR ALGORITHM		
PROB #	EXECUTION TIME	INITIAL SOLUTION	EXECUTION TIME	INITIAL SOLUTION	OPTIMAL SOLUTION
1	.03	22	.09	21	20
2	.05	18	.05	17	15
3	.04	65	.20	63	63
4	.04	15	.07	15	15
5	.03	104	.05	104	104
6	.04	199	.07	199	199
7	.03	81	.74	106	81
8	.03	197	.05	185	185
9	.03	26	.06	26	26
10	.04	20	.04	16	16

TABLE 9. EXECUTION RESULTS OF SIX-CITY PROBLEMS

the ten six-city problems was 0.036 seconds for the Closest-Unvisited-City algorithm and 0.142 seconds for the Misra-Fair algorithm. The standard deviations for the Closest-Unvisited-City algorithm and the Misra-Fair algorithm were 0.007 seconds and 0.215 seconds, respectively. The Closest-Unvisited-City algorithm found the optimal solution 50% of the time; while the Misra-Fair algorithm found the optimal solution 70% of the time. The average error for the Closest-Unvisited-City algorithm was 6.47% and the average error for the Misra-Fair algorithm was 4.91%. While the Misra-Fair algorithm was almost 4 times slower, it found the optimal solution more often and the non-optimal solutions found were closer to the optimal solution than the Closest-Unvisited-City algorithm.

Also, ten problems with 10 cities were run using the Closest-Unvisited-City algorithm and the Misra-Fair algorithm. The results obtained for these problems are summarized in Table 10. The WATFIV compiler of the Amdahl 470 computer has

	CLOSEST-UNVISITED-CITY ALGORITHM		MISRA-FAIR ALGORITHM		
PROB #	EXECUTION TIME	INITIAL SOLUTION	EXECUTION TIME	INITIAL SOLUTION	OPTIMAL SOLUTION
1	.19	22	.29	30	20
2	.15	146	.27	147	146
3	.17	9	.12	5	5
4	.12	169	2.00	-	146
5	.15	117	.23	117	117
6	.16	121	1.39	121	121
7	.16	174	2.00	-	159
8	.18	177	1.58	160	155
9	.18	126	2.00	-	126
10	.17	162	2.00	-	144

TABLE 10. EXECUTION RESULTS OF TEN-CITY PROBLEMS

a two second limit for execution time. The Misra-Fair program failed to determine an initial solution for of the ten problems before the two second time limit was exceeded. However, on the problems that were solved, 50% of the initial solutions were also the optimal solutions. The average error for the non-optimal initial solutions was 14.19% for the Closest-Unvisited-City algorithm and 8.98% for the Misra-Fair algorithm.

The ten six-city problems were solved again using the Branch-and-Bound progran and the Misra-Fair program set to obtain the optimal solution. Table 11 shows the results of running these problems. The average execution time for the

	BRANCH-AND-BOUND ALGORITHM	MISRA-FAIR ALGORITHM	
PROB #	EXECUTION TIME	EXECUTION TIME	OPTIMAL SOLUTION
1	.11	.13	20
2	.08	.12	15
3	.07	1.20	63
4	.06	.07	15
5	.05	.05	104
6	.05	.07	199
7	.10	1.57	81
8	.05	.06	185
9	.07	.11	26
10	.05	.05	16

TABLE 11. OPTIMAL SOLUTION RESULTS OF SIX-CITY PROBLEMS

Branch-and-Bound program was 0.069 seconds with a standard deviation of 0.0218 seconds. The Misra-Fair program had an average execution time of 0.343 seconds and a standard deviation of 0.556 seconds.

The ten problems with ten cities were also rerun using the Branch-and-Bound program and the Misra-Fair program set to obtain the optimal solution. The results of these runs are shown in Table 12. The Misra-Fair program found the

	BRANCH-AND-BOUND ALGORITHM	MISRA-FAIR ALGORITHM	
PROB #	EXECUTION TIME	EXECUTION TIME	OPTIMAL SOLUTION
1	.63	.74	20
2	.39	1.51	146
3	.21	.24	5
4	.62	2.00	146
5	.20	.23	117
6	.56	2.00	121
7	.28	2.00	159
8	.51	2.00	155
9	.20	2.00	126
10	.65	2.00	144

TABLE 12. OPTIMAL SOLUTION RESULTS OF TEN-CITY PROBLEMS

optimal solution for only four of the ten problems before the execution time limit of the WATFIV compiler was exceeded.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

The Misra-Fair algorithm is simple and easy to understand. Hand solution of problems of up to ten cities is quite easy and quick.

As the program for the Misra-Fair algorithm is presently written, a subroutine sets up the Solution Tables using all possible combinations of costs currently being considered. This is done in both the Solution Routine and the Optimization Routine. Computational experience with this program indicates that combinatorial set-up for the Solution Tables gives a better initial solution, but takes more time to find this better solution. Using this method, the program is slower than the Closest-Unvisited-City program, but more accurate. It is therefore recommended that the combinatorial method be used if the more accurate initial solution is desired. If a "quick and dirty" first guess is all that is desired, it is suggested that the Solution Tables be set up by including the next highest cost path-segments only in each iteration.

When the Optimization Routine is used, the Misra-Fair algorithm is much slower than the Branch-and-Bound algorithm when the combinatorial method of setting up the Solution Tables is used in the Misra-Fair program. Not enough problems

have been run to determine the relative speed of solution of the Misra-Fair algorithm when the Solution Tables are built by adding the next highest cost path-segements only during each iteration. A final decision on the value of the Misra-Fair algorithm should not be made until this modification has been implemented and many problems run.

Therefore, with the information that presently exists, the value of the Misra-Fair algorithm for solution of the Traveling-Salesman problem lies in its ease of hand solution of fairly large problems and the reasonably accurate first guess solution found using the computer program. More computational experience could prove the Misra-Fair algorithm to be valuable for computer solution of large problems when the optimal solution is desired.

REFERENCES

1. Bellmore, M. and G. L. Nemhauser, "The Traveling-Salesman Problem: A Survey," Opns. Res., Vol. 16, 538-558, (1968).
2. Conway, R. W., W. L. Maxwell and L. W. Miller, Theory of Scheduling, Addison-Wesley Co., Inc., (1967).
3. Gonzales, R. H., "Solution to the Traveling-Salesman Problem by Dynamic Programming on the Hypercube," Tech. Report #18, Opns. Res. Center, MIT, (1962).
4. Hansen Helbig, K. and J. Krarup, "Improvements of the Held-Karp Algorithm for the Symmetric Travelling Salesman Problem," Mathematical Programming, Vol. 7, 87-96, (1974).
5. Hatfield, D. J. and J. F. Pierce, "Production Sequencing by Combinatorial Programming," IBM Cambridge Scientific Center, Cambridge, Mass., (1966).
6. Held, M. and R. M. Karp, "A Dynamic Programming Approach to Sequencing Problems," SIAM, Vol. 10, 196-210, (1962).
7. ____ and ____, "Travelling Salesman Problem and Minimum Spanning Trees," Opns. Res., Vol. 18, 1138-1162, (1970).

8. ____ and ____, "The Travelling Salesman Problem and Minimum Spanning Trees: Part II," Mathematical Programming, Vol. 1, 6-25, (1971).

9. Krolak, P., W. Felts and G. Marlbe, "A Man-Machine Approach Toward Solving the Traveling Salesman Problem," CACM, Vol. 14, 327-334, (1971).

10. Lin, S., "Computer Solution of the Traveling Salesman Problem," Bell System Tech. J., Volume 44, 2245-2269, (1965).

11. ____ and B. W. Kerningham, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," Opns. Res., Vol. 21, 498-516, (1973).

12. Little, J. D. C., K. G. Murty, D. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," Opns. Res., Vol. 11, 979-989, (1963).

13. Martin, G. T., "Solving the Traveling Salesman Problem by Integer Programming," CEIR, New York, (1966).

14. Misra, Ram B. and David F. Fair, "A New Algorithm for Solving the Traveling Salesman Problem," an unpublished paper, USAMC Graduate Intern Training Center, April, (1975).

15. Reiter, S. and G. Sherman, "Discrete Optimizing," SIAM, Vol. 13, 864-889, (1965).