Pergamon

Computers Ops Res. Vol. 25, No. 11, pp. 879–885, 1998

Printed in Great Britain
PII: S0305-0548(98)00032-X
0305-0548/98 $19.00 + 0.00

IMPROVED SOLUTIONS FOR THE TRAVELING PURCHASER PROBLEM

W. L. Pearn†‡ and R. C. Chien§

Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu, Taiwan R.O.C.

(Received December 1996; in revised form February 1998)

Scope and Purpose—The traveling purchaser problem (TPP) is a generalization of the well-known traveling salesman problem (TSP), which has many real-world applications. Examples include the purchase of the required raw materials for the manufacturing factories in which the total cost has to be minimized, and the scheduling of a set of jobs over some machines with different set-up and job processing costs in which the total cost for completing the jobs has to be minimized. The TPP has been shown to be computationally intractable. Therefore, many heuristic solution procedures have been proposed to solve the TPP approximately. The purpose of this paper is to consider some variations of the existing solution procedures to improve the solutions. Computational experiments and performance comparisons among the existing algorithms as well as the proposed variations are provided, and the results are analyzed.

Abstract—The traveling purchaser problem (TPP) is an interesting generalization of the well-known traveling salesman problem (TSP), in which a list of commodity items have to be purchased at some markets selling various commodities with different prices, and the total travel and purchase costs must be minimized. Applications include the purchase of raw materials for the manufacturing factories in which the total cost has to be minimized, and the scheduling of jobs over some machines with different set-up and job processing costs in which the total costs for completing the jobs has to be minimized. The TPP has been shown to be computationally intractable. Therefore, many heuristic solution procedures, including the Search algorithm, the Generalized-Savings algorithm, the Tour-Reduction algorithm, and the Commodity-Adding algorithm have been proposed to solve the TPP approximately. In this paper, we consider some variations of these algorithms to improve the solutions. The proposed variations are compared with the existing solution procedures. The results indicate that the proposed variations significantly improve the existing solutions.

Key words: Network optimization, combinatorics, complexity

1. INTRODUCTION

There are numerous generalizations of the traveling salesman problem (TSP). Examples include the time-constrained traveling salesman problem (TCTSP), the stochastic traveling salesman problem (STSP), the multiple traveling salesman problem (MTSP), the vehicle routing problem (VRP), and many others. The traveling purchaser problem (TPP) is another interesting generalization of the TSP, which has many applications. One example is the purchase of parts and raw materials for the manufacturing factories in which the total cost has to be minimized. Since the required parts and raw materials may be carried by various warehouses located in different areas with different prices in this case, the purchaser has to select the warehouses to be visited, and plan the tour for these selected warehouses in order to minimize the sum of the purchase and the travel costs. Another example is the scheduling of a set of jobs over some machines. Each machine has a different set-up cost and a different job processing cost, and the

†To whom all correspondence should be addressed. Tel.: +35-714261/35-712121; E-mail: roller@cc.nctu.edu.tw

‡Dr W. L. Pearn is a Professor of Operations Research and Quality Management at the Department of Industrial Engineering and Management, National Chiao Tung University, Taiwan, R.O.C. He received his Ph.D. degree from the University of Maryland at College Park, MD, U.S.A. Dr Pearn worked for AT&T Bell Laboratories at Switch Network Control Center and Process Quality Center.

§Mr R. C. Chien received his B.S. degree in Management Information System from Chung Yuan Christian University, and M.S. degree in System and Decision Sciences from the Department of Industrial Engineering and Management, National Chiao Tung University, Taiwan, R.O.C.

engineer has to determine which machines, and in what orders the jobs should be processed on these machines so that the total cost to complete the jobs is minimized.

The TPP was originally proposed by Ramesh [6]. The problem may be stated as follows. We are given a network with a set of markets $V = \{v_1, v_2, \ldots, v_n\}$ with a special market v_1 called the domicile, and a set of items $P = \{1, 2, \ldots, m\}$ to be purchased. We are also given a travel cost matrix $D = [d(v_i, v_j)]$, $1 \le i, j \le n$, between any two markets; and $C = [c(v_i, k)]$, $2 \le i \le n$, $1 \le k \le m$, the purchase costs of the items at the various markets. Then, the objective of the TPP is to find a tour, which starts at the domicile v_1, goes through a subset of the n markets and returns to v_1 once all m items are purchased at chosen markets, and the total travel and purchase cost is minimized. It is assumed that: (1) each item is available in at least one market; (2) the traveling purchaser may pass through a market or the domicile v_1 any number of times without purchasing an item there; (3) the traveling purchaser may purchase as many items as there are available at each market; and (4) no items are available at the domicile v_1.

The TPP has been shown to be computationally intractable, and many heuristic solution procedures have been proposed to solve the TPP approximately, including the Search algorithm [6], the Generalized-Savings algorithm [2], the Tour-Reduction algorithm [3] and the Commodity-Adding algorithm [4]. In this paper, we consider several variations of the four existing algorithms to improve the solutions. The proposed variations are compared with the existing solution procedures. The results indicate that the proposed variations significantly improve the original versions of the four existing solution procedures. Extensive computational experiments and comparisons are provided.

2. THE SEARCH ALGORITHM

Ramesh [6] developed an exact algorithm based on a lexicographic search procedure to solve the TPP optimally. In his approach, an alphabet table of words representing all possible sequences of markets is first created. The algorithm then searches for each possible word (a sequence of markets) according to the alphabet table, and checks if the current sequence of markets constitute a feasible solution. Some lower bound techniques are also used to accelerate the search procedure. Unfortunately, this algorithm is computationally inefficient, only problems with small and moderate sizes can be solved optimally. Based on the search procedure, Ramesh [6] also developed a heuristic algorithm to solve the problem approximately. The algorithm executes the search procedure until a feasible solution is found. To improve the solution, Ramesh [6] also considered a modification which requires that all the markets be examined at least once, or the bound of any partial solution exceeds current best solution. In the following, we consider two variations of the Search algorithm, which we refer to as the Next-Bloc (A1), and the Next-Neighbor (A2) search algorithms.

(A1) Next-Bloc Search Algorithm

We first execute the lexicographic search procedure until a bloc of word is rejected. We then move to the next bloc of a word and continue to search for better solution. The search procedure is repeated until a certain number (is set to 5 in this paper) of blocs are rejected.

(A2) Next-Neighbor Search Algorithm

We start with the nearest neighbor of v_1, and execute the lexicographic search procedure until a bloc of word is rejected. We then move to the second nearest neighbor of v_1 and continue to search for better solution until a bloc of word is rejected. The search procedure is repeated until all the $M - 1$ neighbors are examined.

In attempting to improve the solutions, Ong [3] considered a procedure using available TSP heuristics to resequence the markets that have to be visited in the solution, to reduce the travel cost. In this paper, we use the TSP heuristic developed by Basart and Huguet [1] for the resequencing. We shall call the variations which apply the TSP resequencing procedure as A1′ and A2′. To further improve the solutions, we proceed with the following two procedures called the market-drop and the market-exchange procedures. Given a TPP solution, the market-drop procedure drops a market from the current solution if it yields a cost reduction. Repeat the market-drop procedure until no more cost reduction can be made. On the other hand, the market-exchange procedure exchanges a market in the solution with another market not in the

solution if it yields a cost reduction. Repeat the market-exchange procedure until no more cost reduction can be made. We call the variations, which apply both the market-drop and the market-exchange procedures after the TSP resequencing, as A1″ and A2″.

3. THE GENERALIZED-SAVINGS ALGORITHM

Golden *et al*. [2] proposed a heuristic solution procedure to solve the TPP approximately. Their solution procedure has been referred to as the Generalized-Savings algorithm. The Generalized-Savings algorithm starts with an initial solution containing the domicile and the market selling more items than any other market at the cheapest price. If there is a tie, choose the one with minimal total price. At each iteration, the algorithm selects a market based on the savings calculated from the travel and the purchase costs, and insert that market into the current tour. The algorithm terminates when no more savings can be made. In the following, we consider two variations of the Generalized-Savings algorithm, which we refer to as the Parameter-Selection (B1), and the Tie-Selection (B2) Generalized-Savings algorithms.

(B1) Parameter-Selection Generalized-Savings Algorithm

In the following definition of savings function, we consider seven values of $\lambda = \{0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6\}$. Each value of λ is used to generate a complete solution, and the best one among the seven is then chosen as the solution from this variation.

$s(i, j, p) = d(i, j) - d(i, p) - d(p, j) + \lambda\Sigma_{k = 1, 2, \ldots, m} g(\alpha, p, k)$, where $d(i, j)$ = the shortest distance between i and j in the current tour; $g(\alpha, p, k) = \max\{f(\alpha, k) - c(p, k), 0\}$, $f(\alpha, k) = \min\{c(i, k)\}$ for all market i in the current tour α, and $c(i, k)$ = price of item k at market i.

(B2) Tie-Selection Generalized-Savings Algorithm

In constructing the initial solution in variation B1, the market "selling more items than any other market at the cheapest price" is selected. If there is a tie, choose one which is nearest to the depot.

Similar to the Search Algorithm, we could improve the solutions using the TSP heuristic by Basart and Huguet [1] to resequence the markets which have to be visited in the solution. We shall call such improved variations which apply the TSP resequencing procedure as B1′ and B2′. To further improve the solutions, we can also apply the market-drop and the market-exchange procedures as described earlier. We shall call such improved variations which apply both the market-drop and market-exchange procedures after the TSP resequencing as B1″ and B2″.

4. THE TOUR-REDUCTION ALGORITHM

Ong [3] proposed a heuristic solution procedure, called the Tour-Reduction algorithm, to solve the TPP approximately. The algorithm starts with an initial tour consisting of some selected markets (which may be chosen by some heuristic procedure or specified by the decision maker) which collectively carry all the m items. In practice, we may select the markets selling at least one item at the lowest prices [5]. The algorithm then examines each of the markets in the tour and checks whether a cost reduction can be made by dropping a market from the current tour. The algorithm repeats the market drop procedure and terminates until no more cost reduction can be made. In the following, we consider two variations of generating initial tours for the Tour-Reduction algorithm, which we refer to as the Adjusted-Cheapest (C1), and the Nearest-Cheapest (C2) Tour-Reduction algorithms. Let S_{C_i} = the set of markets in the initial tour for variation C_i ($i = 1, 2$) of the Tour-Reduction algorithm.

(C1) Adjusted-Cheapest Tour-Reduction Algorithm

$S1 = \{v_i | c(i, k)$ is minimal for some item k, $1 \le k \le m\}$,
$S2 = \{v_i | c(i, k) + d(v_1, v_i)$ is minimal for some item k, $1 \le k \le m\}$.
$S_{C_1} = S1 \cup S2$. Connect the markets in S_{C_1} using any TSP algorithm.

(C2) Nearest-Cheapest Tour-Reduction Algorithm

$S1 = \{v_i | c(i, k)$ is minimal for some item k, $1 \le k \le m\}$,
$S3 = \{$a proportion, p, of the markets nearest to $v_1\}$.

Table 1. A comparison between the variations and the originals for the Search and Generalized-Savings algorithms on the thirty test problems

	A	A1′	A2′	A1″	A2″	B	B1′	B2′	B1″	B2″
Average % above the optimal solution	18.7	13.7	5.4	6.8	2.0	15.7	5.6	4.5	1.5	2.0
Worst % above the optimal solution	70.0	70.0	35.0	40.0	20.0	46.3	32.3	35.7	19.0	23.2
Number of optimal solutions obtained	4	7	12	14	24	3	11	14	20	20
Number of the best solutions obtained	10	14	29	18	29	19	26	28	29	29
No. of solutions worse than the original	–	0	0	0	0	–	0	1	0	1
No. of solutions tied with the original	–	14	8	7	4	–	8	6	3	3
No. of solutions better than the original	–	16	22	23	26	–	22	23	27	26
Ave. % of improvement over the original	–	7.2	13.4	11.7	14.3	–	11.2	12.5	12.8	13.0

$S_{C_2} = S1 \cup S3$. Connect the markets in S_{C_2} using any TSP algorithm.

We consider $p = 0.1$, 0.2, 0.3, 0.4, and 0.5. Each value of p is used to generate a complete solution, and the best one is chosen to be the solution from this variation.

We could improve the solutions using the TSP algorithm by Basart and Huguet [1] to resequence the order of the markets that have to be visited in the solution. We shall call the improved variations which apply the TSP resequencing procedure as C1′ and C2′. To further improve the solutions, we can apply the market-exchange procedure to examine whether any cost reduction can be made by exchanging a market in the solution with another market not in the solution. We can also apply the market-drop procedure which drops a market from the current solution (after the market-exchange procedure) if it yields a cost reduction. We shall call such improved variations which apply both the market-exchange and the market-drop procedures as C1″ and C2″.

5. THE COMMODITY-ADDING ALGORITHM

Pearn [4] introduced a solution procedure to solve the TPP approximately. The solution procedure has been referred to as the Commodity-Adding algorithm. The Commodity-Adding algorithm starts with an initial solution containing the domicile and the market which minimizes the total cost (travel cost + purchase cost) for purchasing the first item. At each iteration, the algorithm considers the next commodity on the list and checks if any cost reduction can be made by adding another market to the solution. The algorithm terminates when all items have been considered. In the following, we consider two variations of the Commodity-Adding algorithm, which we refer to as the Random-Order (D1), and the Sequence-Order (D2) Commodity-Adding algorithms.

(D1) Random-Order Commodity-Adding Algorithm

In the Commodity-Adding algorithm, a list of items with given order 1, 2, ..., m, is given. We randomly generate k (is set to 10 in this paper) lists with different orders. Each order is used to generate a complete solution, and the best one among the k solutions is chosen as the solution from this variation.

(D2) Sequence-Order Commodity-Adding Algorithm

In the Commodity-Adding algorithm, a list of items with given order 1, 2, ..., m, is given. We consider the following lists with orders {(1, 2, ..., m), (2, 3, ..., m, 1), (3, 4, ..., m, 1, 2),

Table 2. A comparison between the variations and the originals for the Tour-Reduction and Commodity-Adding algorithms on the thirty test problems

	C	C1′	C2′	C1″	C2″	D	D1′	D2′	D1″	D2″
Average % above the optimal solution	9.3	7.1	4.6	5.7	4.1	8.5	1.5	2.2	0.9	1.4
Worst % above the optimal solution	61.2	50.0	50.0	50.0	50.0	38.7	6.5	10.1	6.5	9.7
Number of optimal solutions obtained	8	11	15	16	19	8	16	17	21	21
Number of the best solutions obtained	20	22	28	25	28	13	28	25	28	26
No. of solutions worse than the original	–	2	1	2	1	–	0	0	0	0
No. of solutions tied with the original	–	23	18	17	16	–	13	16	13	14
No. of solutions better than the original	–	5	11	11	13	–	17	14	17	16
Ave. % of improvement over the original	–	9.8	9.8	7.9	9.4	–	10.4	11.3	11.2	11.1

Table 3. Performance of the four algorithms and the variations with the best performance on the two sets of test problems

	A	A2″	B	B1″	C	C2″	D	D1″
Average % above the optimal (30)	18.7	2.0	15.7	1.5	9.3	4.1	8.5	0.9
Worst % above the optimal (30)	70.0	20.0	46.3	19.0	61.2	50.0	38.7	6.5
Number of optimal solutions (30)	4	24	3	20	8	19	8	21
Number of worse solutions (90)	–	0	–	0	–	6	–	0
Number of tied solutions (90)	–	9	–	11	–	22	–	23
Number of better solutions (90)	–	81	–	79	–	62	–	67
Average % of improvement (90)	–	14.5	–	12.6	–	9.8	–	11.8

$\ldots, (m, 1, 2, \ldots, m-1)\}$. Each order is used to generate a complete solution, and the best solution is chosen as the solution from this variation. In this paper, we will consider only min $\{10, m\}$ sequences of orders.

We could improve the solutions using the TSP algorithm by Basart and Huguet [1] to resequence the markets that have to be visited in the solution. We shall call the improved variations which apply the TSP resequencing procedure as D1′ and D2′. To further improve the solutions, we can also apply the market-drop and the market-exchange procedures. We shall call the further-improved variations as D1″ and D2″.

6. PERFORMANCE COMPARISONS

We first experimented with the proposed variations on a set of thirty sample problems, and compared with the original algorithms. The thirty sample problems were randomly generated with the following characteristics: (1) the number of markets is from 10 to 50, (2) the number of commodity items is from 5 to 60, (3) the travel costs are from 1 to x, where $15 \leq x \leq 140$, and (4) the purchase costs are from 0 to y, where $5 \leq y \leq 75$. Some networks are dense, and others are sparse. For those TPP problems, we also implemented the lexicographic search algorithm proposed by Ramesh [6] to find the problem optimal solutions. The preliminary comparison, summarized in Tables 1 and 2, is based on the average percentage above the optimal solution, the worst percentage above the optimal solution, number of optimal solutions obtained, number of the best solutions obtained, number of solutions worse than those obtained by the original algorithm, number of solutions tied with those obtained by the original algorithm, number of solutions better than those obtained by the original algorithm, and the average percentage of improvement over the original algorithm.

The results showed that for the Search Algorithm, variation A2″ received 29 best solutions (out of 30), which outperformed the other three variations. Therefore, only variation A2″ (Near-Neighbor Search) will be considered. For the Generalized-Savings Algorithm, variation B1″ received 29 best solutions (out of 30), which outperformed the other three variations. Therefore, only variation B1″ (Parameter-Selection Generalized-Savings) will be considered. For the Tour-Reduction Algorithm, variation C2″ received 28 best solutions (out of 30), which outperformed the other three variations. Therefore, only variation C2″ (Nearest-Cheapest Tour-Reduction) will be considered. For the Commodity-Adding Algorithm, variation D1″ received 28 best solutions (out of 30), which outperformed the other three variations. Therefore, only variation D1″ (Random-Order Commodity-Adding) will be considered. To compare the four best

Table 4. A comparison of the four best variations

	A2″	B1″	C2″	D1″
Average % above the optimal solution (30)	2.0	1.5	4.1	0.9
Worst % above the optimal solution (30)	20.0	19.0	50.0	6.5
Number of optimal solutions obtained (30)	24	20	19	21
Number of the best solutions obtained (90)	45	47	36	63
Number of the worst solutions obtained (90)	20	22	32	10
Average run time CPU ($40 \leq n \leq 50$)	25.6	1.56	4.07	0.94

Table 5. Performance of D1″ for various values of R

	10	25	50	100
Average % above the optimal solution (30)	0.9	0.8	0.7	0.5
Worst % above the optimal solution (30)	6.5	6.5	6.5	6.5
Number of optimal solutions obtained (30)	21	23	25	26
Number of the best solutions obtained (90)	63	66	72	79
Number of the worst solutions obtained (90)	10	7	4	3
Average run time CPU ($40 \leq n \leq 50$)	0.94	2.33	4.62	9.28

variations A2″, B1″, C2″, and D1″ with the original algorithms, we randomly generated another sixty test problems with the same characteristics.

Table 3 summarizes the performance of the four original algorithms (the Search algorithm, the Generalized-Savings algorithm, the Tour-Reduction algorithm, and the Commodity-Adding algorithm), and their best variations A2″, B1″, C2″, and D1″. The results indicated that the improvements made by those variations over the four original algorithms were indeed significant. In fact, for the Search algorithm, the best variation A2″ reduced 16.7% (on the average) of the problem solutions, reduced 50% of the worst solution, received 24 (out of 30) optimal solutions, and improved 81 (out of 90) problem solutions. For the Generalized-Savings algorithm, the best variation B1″ reduced 14.2% (on the average) of the problem solutions, reduced 27.3% of the worst solution, received 20 (out of 30) optimal solutions, and improved 79 (out of 90) problem solutions. For the Tour-Reduction algorithm, the best variation C2″ reduced 5.2% (on the average) of the problem solutions, reduced 11.2% of the worst solution, received 19 (out of 30) optimal solutions, and improved 62 (out of 90) problem solutions. For the Commodity-Adding algorithm, the best variation D1″ reduced 7.6% (on the average) of the problem solutions, reduced 32.2% of the worst solution, received 21 (out of 30) optimal solutions, and improved 67 (out of 90) problem solutions.

Table 4 is a comparison among the four best variations A2″, B1″, C2″, and D1″ in terms of the average percentage above the optimal solutions, the worst percentage above the optimal solutions, number of optimal solutions obtained, number of the best solutions obtained, number of the worst solutions obtained, and the average run time in CPU seconds. We note that variation D1″ (Random-Order Commodity-Adding) significantly outperformed the other three best variations. In fact, variation D1″ receive an average of 0.9% above the optimal solutions. Furthermore, variation D1″ requires the least amount of computer time (less than 1 CPU second) among all. In our experiment, the number of random orders selected for the commodity items (R) in the variation D1″ was initially set to ten ($R = 10$). For problems with $40 \leq n \leq 50$, the average run time was 25.6 (CPU) seconds for variation A2″, 1.56 s for variation B1″, 4.07 s for variation C2″, and 0.94 s for variation D1″.

Table 5 displays the results of our experiment on variation D1″ (the Random-Order Commodity-Adding algorithm) with $R = 10$, 25, 50 and 100. For $R = 100$, the average percentage above the optimal solution is reduced to 0.5%, and 26 out of 30 problems (87%)

Table 6. Performance comparisons on problems with $n > 50$

	n	m	A	B	C	D	A2″	B1″	C2″	D1″
1	75	100	6023	7727	11311	5854	4823	4594	8626	*3597*
2	90	300	6580	6571	6341	6600	6361	6322	6314	*6282*
3	100	300	2550	3008	5005	2658	2333	2398	4341	*2330*
4	110	120	3228	3057	3425	2979	2966	*2697*	3096	2712
5	120	80	2977	3181	3543	2746	2837	2590	3241	*2271*
6	130	120	4881	4831	9797	4418	4219	4520	8733	*3955*
7	140	180	3158	3264	3497	3142	2985	2853	3346	*2849*
8	150	200	3909	4945	8445	4022	3569	3690	6139	*3442*
9	180	150	2319	2213	3122	2043	2154	2058	2903	*1936*
10	200	140	2239	2223	4370	2264	2097	2144	3527	*1955*

received optimal solutions. The average run time (in CPU seconds) for problems with $40 \leq n \leq 50$, however, is increased to 9.3 s.

Table 6 displays the performance of the four original algorithms and the four best variations on the ten problems with number of markets ranging from 75 to 200, and number of commodity items ranging from 80 to 300. For the ten problems, variation D1″ received 9 best solutions (out of 10), while variation B1″ received 1 best solution. Variation D1″ again significantly outperformed the other three best variations as well as the four original algorithms. All the four original algorithms, the four best variations, and the lexicographic search algorithm which generated the problem optimal solutions were coded in FORTRAN programming language (with FORTRAN IV Compiler) and run on PC-486 DX-100.

7. CONCLUSIONS

The traveling purchaser problem (TPP) is an interesting generalization of the well-known traveling salesman problem (TSP), which has many applications. The TPP has been shown to be computationally intractable. Therefore, many heuristic solution procedures have been proposed to solve the TPP approximately, including the Search algorithm, the Generalized-Savings algorithm, the Tour-Reduction algorithm, and the Commodity-Adding algorithm.

In this paper, we considered several variations of the four existing algorithms to improve the solutions. Those variations include the Next-Bloc and the Next-Neighbor search algorithms, the Parameter-Selection and the Tie-Selection Generalized-Savings algorithms, the Adjusted-Cheapest and the Nearest-Cheapest Tour-Reduction algorithms, and the Random-Order and the Sequence-Order Commodity-Adding algorithms. We experimented with the proposed variations on many problems which were randomly generated. The results indicated that the proposed variations significantly improved the original algorithms. In particular, the improvements made by the Next-Neighbor Search, the Parameter-Selection Generalized-Savings, the Nearest-Cheapest Tour-Reduction, and the Random-Order Commodity-Adding algorithms with TSP resequencing, the market-drop, and the market-exchange improvement procedures, were remarkable. We also compared the four best variations. The results indicated that the Random-Order Commodity-Adding algorithm (variation D1″) significantly outperformed the other three best variations as well as the four original algorithms.

Acknowledgements—The authors would like to thank the anonymous referees for their careful reading of the paper and several suggestions which improved the paper.

REFERENCES

1. Basart, J. M. and Huguet, L., An approximation algorithm for the TSP. *Information Processing Letters*, 1989, **31**(2), 77–81.
2. Golden, B. L., Levy, L. and Dahl, R., Two generalizations of the traveling salesman problem. *OMEGA*, 1981, **9**(4), 439–445.
3. Ong, H. L., Approximate algorithms for the traveling purchaser problem. *Operations Research Letters*, 1982, **1**(5), 201–205.
4. Pearn, W. L., On the traveling purchaser problem. Working Paper 91-01. Department of Industrial Engineering and Management, National Chiao Tung University, 1991.
5. Pearn, W. L. and Chien, R. C., A new algorithm for the traveling purchaser problem. Working Paper 96-01. Department of Industrial Engineering and Management, National Chiao Tung University, 1996.
6. Ramesh, T., Traveling purchaser problem. *OPSEARCH*, 1981, **18**(2), 78–91.