Journal of the Operations Research
Society of Japan
Vol. 33, No. 4, December 1990

HEURISTIC ALGORITHMS FOR THE SINGLE VEHICLE DIAL-A-RIDE PROBLEM

Mikio Kubo Hiroshi Kasugai
Waseda University

(Received January 31, 1990; Revised June 22, 1990)

Abstract A number of papers have been proposed for the approximate solution of the dial-a-ride routing problem. The object of this paper is to examine some of these heuristics, to introduce some new heuristics, and to compare these approximate algorithms.

1. Introduction

In this paper, we analyze heuristic algorithms for the single vehicle many-to-many dial-a-ride problem (DARP), i.e. designing a route for a vehicle carrying customers between specified pick-up and delivery points to minimize the total travel distance. More formally, the DARP can be described as follows.

(Dial-A-Ride Problem : DARP)

Given a complete directed graph $G(V, E)$ consisting of a set V of n nodes and an arc set E. Node set V can be decomposed into three subsets; depot v_0, pick-up points $v_\ell^p, \ell = 1, 2, \cdots, m$, delivery points $v_\ell^d, \ell = 1, 2, \cdots, m$. Node v_0 represents a depot that is the starting and ending point of the vehicle. There exist m customers whose pick-up and delivery point pairs are $(v_\ell^p, v_\ell^d), \ell = 1, 2, \cdots, m$, where v_ℓ^p is the pick-up point and v_ℓ^d is the delivery point of customer ℓ. We denote the number of nodes by $n = \mid V \mid = 2m+1$. Associated with each arc $(i, j) \in E$, there is a weight D_{ij} that represents the distance between node i and node j. Our objective is to find a permutation of nodes $\delta : V \to \{1, 2, \cdots, n\}$ that minimizes the total travel distance

$$\sum_{i=1}^{n-1} D_{\delta^{-1}(i)\delta^{-1}(i+1)} + D_{\delta^{-1}(n)\delta^{-1}(1)} \tag{1}$$

and satisfies the following conditions:

1. $\delta_{(v_0)} = 1$,
2. $\delta_{(v_\ell^p)} < \delta_{(v_\ell^d)}$ for all $\ell = 1, 2, \cdots, m$.

Condition 2 states the pick-up point v_ℓ^p must be visited before its corresponding delivery point v_ℓ^d. Note that permutation $\delta_{(i)} = j$ means node i is the j-th visiting point of the vehicle.

Since the DARP is NP-complete [6], a polynomial time algorithm most unlikely exists for solving the DARP exactly. Consequently, for practical problems with several hundreds of nodes we must use an approximate algorithm that runs in low polynomial time order. The objective of this paper is to develop several approximate algorithms, and to compare these approximate algorithms with the known heuristics.

This paper is organized as follows. In section 2 we review several heuristic algorithms in literature. In section 3 we describe several new heuristic algorithms. In section 4 we compare these approximate algorithms on randomly generated Euclidean problems. Final section contains conclusions.

We should remark that our numerical experiments are carried out for a limited number of test problems on Euclidean distance metric. We feel that out results illustrate the degree of accuracy in many practical situations though there exist many problems of different geometry. Furthermore, we should remark that the asymptotically optimal heuristic for the DARP introduced by Stein [19] is not tested in this paper, since Psaraftis compared it in [16] with his heuristic (minimum spanning tree heuristic) and concluded that minimum spanning tree heuristic performs better than Stein's heuristic on medium size problems.

2 Previous Works

The previous works are summarized as follows.

Kalantari et al. [9] provided a branch and bound method for the precedence constrained traveling salesman problem (PCTSP) based on the branch and bound algorithm for the TSP by Little et al. [12], and could solve the 30-node problems with the asymmetric distance matrix. Suzuki and Nomura [22] also developed a branch and bound algorithm for the PCTSP using bounds derived from the arborescence problem or the shortest path problem, and could solve 30-node problems with the sparse distance matrix. In their model, nodes are permitted to be visited more than once and the starting and ending points are distinct, but this problem can be reduced to the ordinary PCTSP using a simple transformation. Psaraftis [15] developed an exact solution method using a dynamic programming algorithm for the DARP with additional constraints whose time complexity is $O(n^2 3^n)$, and could solve 20-node problems.

Several heuristic algorithms have been proposed for the DARP (Psaraftis [16], [17], Jaw et al. [8], Stein [19]). Psaraftis proposed the minimum spanning tree (MST) heuristic and local search (k-opt) procedures specialized to the DARP. We briefly summarized them below since we compare them with our algorithms in section 4.

(Minimum Spanning Tree (MST) Heuristic [16])

Step 1 Find the minimum spanning tree on $G(V, E)$.

Step 2 Construct a traveling salesman tour by duplicating the minimum spanning tree.

Step 3 Start with any pick-up point, move on the traveling salesman tour clockwise (or counterclockwise) until all nodes are visited not visiting any node that has been previously visited, or any delivery point whose pick-up point has not been previously visited.

Step 4 Repeat Step 3, each time choosing a different customer pick-up point, and choose the tour with minimum length.

(k-opt procedures [17])

Step 1 Obtain an initial feasible dial-a-ride routing tour.

Step 2 Substitute k arcs with other arcs. If the total travel distance is improved and all pick-up points are visited before their delivery points, then we get a new dial-a-ride routing tour. Continue until no improved tour can be found.

The MST heuristic requires $O(n^2)$ computations, while the k-opt procedure requires $O(n^k)$ computations under the appropriate implementation described in [17]. Since the computational complexity of k-opt procedure grows in exponential order of k, Psaraftis recommended $k = 2, 3$ (2-opt, 3-opt) in practice.

Stein [19], [20] provided an algorithm for the Euclidean DARP based on the Karp's algorithm [10] for the TSP, and proved the algorithm produces asymptotically optimal solutions when ζ approaches infinite.

Daganzo [5] developed analytical models to evaluate the performance of the DARP. Jaw et al. [8] provided an insertion heuristic for the multiple vehicle DARP with several additional constraints such as the time window and service quality constraints. Their algorithm builds tours of multiple vehicles through sequential insertion of customers in a dynamically changing environment. Some algorithms proposed in subsection 3.1 of this paper are based on the same concept as theirs, but the node selection rule and the insertion criteria are different.

3 Heuristic Algorithms

In this section we propose several new heuristic algorithms for the DARP. The first class of algorithms are based on the concept of sequential insertion of nodes that has been introduced in [8]. Our algorithms contain several new features designed to provide the efficiency and the performance. The second algorithm is an extended version of the classical nearest neighbor method, but it contains the randomized routine to overcome the difficulty of the DARP. The third algorithm uses the spacefilling curve heuristic [2]. This algorithm can be seen as a simplified version of the Stein's asymptotically optimal heuristic algorithm [19]. The fourth class of algorithms are local improvement procedures that are generalizations of the Or-opt procedure [11] for the TSP.

3.1 Insertion Methods

We describe a class of construction procedures based on the concept of sequential insertion of nodes. Though all construction procedures begin with a self loop as an initial subtour, they differ at the following two points:

1. the nodes are inserted one by one, or the pick-up and corresponding delivery points are inserted simultaneously;
2. the selection rule of the node (or the pair of nodes).

We first describe two variants of the farthest and arbitrary insertion methods [7],[11] for the TSP. These algorithms insert nodes one by one, but they differ in the selection rules of nodes. The first algorithm that uses the farthest insertion rule is as follows.

(FI : Farthest Insertion Method)

Step 1 (Initialization)
Start with a subtour consisting of node v_0 only, i.e. a self loop $(v_0, (v_0, v_0), v_0)$.

Step 2 (Node Selection)
Given a subtour, find node k that is not in the subtour and farthest to any node in the subtour.

Step 3 (Insertion)
Find the arc (i, j) in the subtour that minimizes $D_{ik}+D_{kj}-D_{ij}$ to satisfy the precedence constraints. Insert k between i and j.

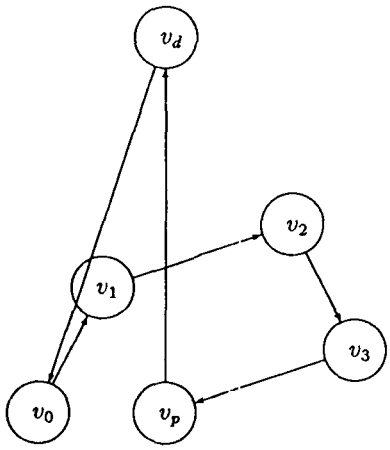

Figure 1: An example that FI and AI do not work well. Nodes v^d and v^p are pick-up and delivery points of the specified customer. If v^d is inserted to the subtour $(v_0, (v_0, v_1), v_1, (v_1, v_2), v_2, (v_2, v_p), v_p, (v_p, v_0), v_0)$, the precedence constraint forces v^d to be inserted between v_p and v_0.

Step 4 Go to Step 3, unless we have a tour.

The second algorithm that uses the randomized insertion rule is as follows.

(AI : Arbitrary Insertion Method)

Step 1 (Initialization)
Start with a subtour consisting of node v_0 only, i.e. a self loop $(v_0, (v_0, v_0), v_0)$.

Step 2 (Node Selection)
Given a subtour, find node k arbitrary that is not in the subtour.

Step 3 (Insertion)
Find the arc (i, j) in the subtour that minimizes $D_{ik}+D_{kj}-D_{ij}$ to satisfy the precedence constraints. Insert k between i and j.

Step 4 Go to Step 3, unless we have a tour.

The above insertion methods generate bad solutions since insertion points are restricted by the precedence constraints. For example see Figure 1. In the Figure v^d and v^p are pick-up and delivery points of the specified customer. If v^d is inserted to the subtour $(v_0, (v_0, v_1), v_1, (v_1, v_2), v_2, (v_2, v_p), v_p, (v_p, v_0), v_0)$, the precedence constraint forces v^d to be inserted between v_p and v_0. This phenomenon is the inherent deficit of the FI and AI methods both of which are natural generalization of the heuristic procedures for the TSP.

To overcome this difficulty, we design the following algorithm, which inserts a pair of pick-up and delivery points simultaneously.

(PI : Pairing Insertion Method)

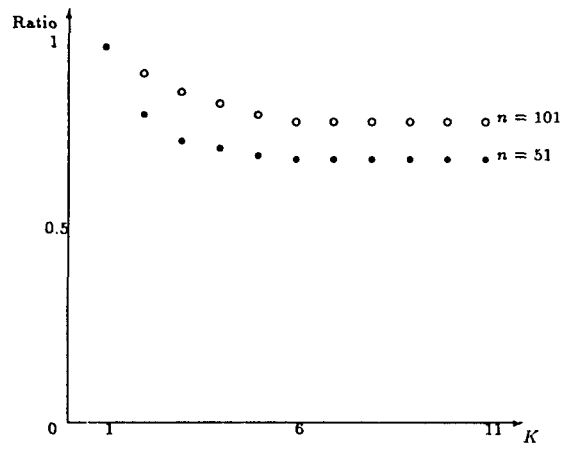

Figure 2: Solution values of MPI method with various K. The values are scaled so that the solution value with $K = 1$ is equal to 1.

Step 1 (Initialization)
Start with a subtour consisting of node v_0 only, i.e. $(v_0, (v_0, v_0), v_0)$.

Step 2 (Pair Selection)
Given a subtour, select a pair of pick-up and delivery points (v^p, v^d) that are not in the subtour.

Step 3 (Insertion)
Find the arcs (i, j) and (i', j') that minimize $D_{iv^p} + D_{v^pj} - D_{ij} + D_{i'v^d} + D_{v^dj'} - D_{i'j'}$. Insert v^p between i and j, and v^d between i' and j'.

Step 4 Go to Step 3, unless we have a tour.

The straightfoward implementation of the pairing insertion method requires $O(n^3)$ computations. We modify it to run in $O(n^2)$. Step 3 of the pairing insertion method is modified in the following way:

Step 3 (Insertion)
Find the K smallest $D_{iv^p}+D_{v^pj}-D_{ij}$ and denote them $(i_1, j_1), (i_2, j_2), \cdots, (i_k, j_k), \cdots, (i_K, j_K$
Then find the arcs (i_k, j_k) and (i', j') that minimize

$$D_{i_kv^p} + D_{v^pj_k} - D_{i_kj_k} + D_{i'v^d} + D_{v^dj'} - D_{i'j'}. \tag{2}$$

Insert v^p between i_k and j_k and v^d between i' and j' that attains minimum in (2).

We refer to this modified algorithm as modified pairing insertion (MPI) method. The MPI method runs in $O(Kn^2)$ where K is a prespecified constant that is independent of N, i.e. $K = O(1)$. The results of the numerical experiments shown in Figure 2 suggest that the solution values obtained by different K are about the same when K is more than 6; we set $K = 6$ in our numerical experiments in section 4.

Table 1: Comparison of the pair selection rules with various problem sizes. The number in the table represents the average percentage above Rule 1 (%).

n	Rule 2	Rule 3	Rule4
11	1.96	0.08	−0.15
21	4.46	1.39	0.62
31	7.36	2.07	1.67
41	7.72	2.54	0.37
51	8.27	2.87	1.99
61	8.20	2.13	1.52
71	7.91	2.99	2.73
81	10.74	1.68	1.91
91	9.32	2.24	1.21
101	9.96	2.19	2.21
111	12.29	3.49	2.60

The effectiveness of the (modified) pairing insertion method depends on the selection criteria of the pair of pick-up and delivery points. We tested four pair selection rules. Let us denote the set of nodes in the subtour by Π, the set of pick-up points in $V - \Pi$ by N_p, the set of delivery points in $V - \Pi$ by N_d. Then, the rules are:

Rule 1 : Farthest Pair

Select a pair (i', j') of pick-up and delivery points such that

$$D_{i'j'} = \max_{i \in N_p, j \in N_d} D_{ij}. \tag{3}$$

Rule 2 : Nearest Pair

Select a pair (i', j') of pick-up and delivery points such that

$$D_{i'j'} = \min_{i \in N_p, j \in N_d} D_{ij}. \tag{4}$$

Rule 3 : MAX-SUM

Select a pair (i', j') of pick-up and delivery points such that

$$D_{k'i'} + D_{\ell'j'} = \max_{k \in \Pi, i \in N_p, \ell \in \Pi, j \in N_d} D_{ki} + D_{\ell j}. \tag{5}$$

Rule 4 : MAX-MAX

Select a pair that contains node j' (pick-up point or delivery point) such that

$$D_{i'j'} = \max_{i \in \Pi, j \in N_p \cup N_d} D_{ij}. \tag{6}$$

We give the results of the numerical experiments to compare the rules in Table 1. As can be seen, the solution value of Rule 1 dominates other rules for all n except $n = 11$. Thus, we recommend Rule 1 (farthest pair rule), and use it in our numerical experiments in section 4.

3.2 A Randomized Nearest Neighbor Method

We also develop a modified version of the nearest neighbor heuristic that incorporates randomization routine. The algorithm called the randomized nearest neighbor (RNN) method can be described as follows.

(Algorithm : RNN method)

Step 1 (Initialization)
Let the set of nodes in the current path be $\Pi = \{v_0\}$ and the last node of the current path be $L = 1$. Determine the node selection parameter $\wp (0 < \wp \leq 1)$.

Step 2 Let the set of pick-up points in $V - \Pi$ be N_p, and the set of delivery points in $V - \Pi$, whose corresponding pick-up point is contained in Π, be N_d.

Step 3 Determine the next visiting point i as follows: go to the nearest pick-up point in N_p with probability

$$\frac{\wp \mid N_p \mid}{\mid N_p \mid + \mid N_d \mid}. \tag{7}$$

Otherwise go to the nearest delivery point in N_d.

Step 4 Let $L := i$ and $\Pi := \Pi \cup \{i\}$. If $\Pi = V$ then stop, otherwise goto Step 2.

3.3 A Spacefilling Curve Heuristic

We use a spacefilling curve instead of using the doubly minimum spanning tree in Psaraftis' MST heuristic. In this case, the heuristic can be seen as a simplified version of the Stein's asymptotically optimal heuristic in which the regions are sequenced according to the spacefilling curve. The computational complexity of this heuristic becomes $O(n \log n)$. If we repeat Step 3 of Psaraftis' MST heuristic, each time choosing a different customer pick-up point, and choose the tour with minimum length, then the complexity of the algorithm becomes the same as the MST heuristic, i.e. $O(n^2)$. We use this version in the numerical experiment in section 4.

3.4 Local Search Methods

In this subsection, we describe two local search algorithms that are extensions of the Or-opt procedure for the TSP [11]. Or-opt procedure temporary deletes a node (or a connected string of nodes) and then inserts it to the subtour. Since we developed several insertion procedures in subsection 3.1, we can easily construct the modified Or-opt procedures for the DARP. The following is a naive generalization of Or-opt procedure.

(Algorithm : Modified Or-opt Method)

Step 1 Get an initial feasible route.

Step 2 Let $s = 3$.

Step 3 Delete a connected string of s nodes from the current tour, and insert the string into the subtour to satisfy the precedence constraints and to minimize the increase of the distance. If no improved solution cannot be found, then go to Step 4.

Step 4 Set $s := s - 1$. If $s = 0$ then stop, otherwise go to Step 3.

If we apply the paring insertion instead of the ordinary insertion, we can get the following algorithm.

(Algorithm : Pairing Or-opt Method)

Step 1 Get an initial feasible route.

Step 2 Delete a pair of pick-up and destination points from the current tour, and insert them into the subtour to satisfy the precedence constraints and to minimize the increase of the distance. Continue Step 2 until no improved solution cannot be found.

Both of the modified Or-opt and pairing Or-opt methods run in $O(n^2)$ time per improvement under the appropriate implementation.

The modified Or-opt method considers only a small percentage of the exchanges that are considered by a 3-opt method, while the pairing Or-opt method considers a subset of those by a 4-opt method.

Of course, we may use two local search procedures described above sequentially. Two variations of the composite local search procedure are possible. The first one is to apply the pairing Or-opt method to begin with, and the modified Or-opt method next. We refer to this version as the composite Or-opt method 1. The second one is to apply the modified Or-opt method to begin with, and the pairing Or-opt method next. We refer to this version as the composite Or-opt method 2.

4 Numerical Experiments

To compare the performance of the approximate algorithms described in the previous section and the known algorithms, all the algorithms are implemented by BASIC and run on microcomputer PC9801VM (NEC) whose speed is almost 1/3600 of VS-FORTRAN on IBM3081.

Distance matrix $[D_{ij}]$ is generated according to two dimensional Euclidean distances between nodes that are distributed on 100 × 100 grids. The precedence relation is generated in the following way; Set the precedence constraints as $i \prec i + \lfloor n/2 \rfloor$ for $i = 2, 3, \cdots, \lfloor n/2 \rfloor$ where n is an arbitrary odd number that represents the number of nodes. We generate 100 instances for each class. Each table gives the average values above the MPI method, i.e. (solution value of each heuristic)/(solution value computed by solving MPI method). We use the following abbreviations in the Tables.

FI : farthest insertion method.

AI : arbitrary insertion method, best of 3 runs.

MPI : modified pairing insertion method.

RNN : randomized nearest neighbor method. Parameter φ is tested from [0, 1], step size is equal to 0.1, and the best solution is selected.

SFC : spacefilling curve method.

MST : Psaraftis' MST heuristic.

In the Tables, CPU time is measured by PC9801VM (10 MHz) seconds per problem. Results of experiments are shown in Table 2 and Table 3. Then, we test several local search types of heuristics. We generate 10 instances with $n = 101$ and take their average values. Initial feasible dial-a-ride tours are obtained randomly, using MST heuristic or using MPI method. We use the following abbreviations in the Tables.

Table 2: Comparison of the construction methods with various problem sizes. The number in the table represents the average percentage of 10 problems above MPI method (%).

n	FI	AI	RNN	SFC	MST
11	11.39	6.14	19.91	27.35	19.81
21	21.09	9.28	23.11	32.52	26.13
31	23.81	10.05	23.79	31.95	27.57
41	28.35	10.79	24.06	31.12	28.29
51	29.51	12.70	25.55	33.30	30.23
61	32.95	11.76	24.45	32.44	28.89
71	32.77	13.24	25.10	34.10	28.13
81	33.75	13.64	26.26	32.71	28.54
91	34.39	12.36	24.73	32.05	27.67
101	33.85	10.53	23.15	31.92	28.11
111	36.18	11.60	23.50	30.81	28.23

Table 3: Comparison of the computational time of the construction methods. The number in the table represents the average CPU time of 10 problems (seconds).

n	MPI	FI	AI	RNN	SFC	MST
11	0	0	1	4	2	3
21	2	1	3	10	6	7
31	4	3	6	17	12	16
41	7	5	10	26	19	22
51	11	8	15	38	29	46
61	15	11	21	52	40	48
71	20	15	28	66	52	77
81	25	19	37	83	66	100
91	33	24	46	103	82	128
101	40	29	56	123	100	144
111	51	35	67	146	120	176

Table 4: Comparison of the local search heuristics with $n = 101$. The number in the table represents the average solution values of 10 problems.

Initial tour	Initial Tour	2-opt	3-opt	Or-opt	Pair-opt	Com-opt1	Com-opt2
Random	534	135	108	114	118	117	109
MST	138	114	103	119	104	103	104
MPI	105	104	100	103	100	100	100

Table 5: Comparison of the computational time of the local search heuristics. The number in the table represents the average CPU times of 10 problems (second).

Initial tour	2-opt	3-opt	Or-opt	Pair-opt	Com-opt1	Com-opt2
Random	1949	30200	2224	563	935	2438
MST	111	12424	480	342	471	755
MPI	442	4226	2384	772	1398	2769

Random : The initial solution is determined randomly.

2-opt : Psaraftis' 2-opt method.

3-opt : Psaraftis' 3-opt method.

Or-opt : modified Or-opt method.

Pair-opt : pairing Or-opt method.

Com-opt1 : composite Or-opt method 1, in which the pairing Or-opt method is applied to begin with, and the modified Or-opt method next.

Com-opt2 : composite Or-opt method 2, in which the modified Or-opt method is applied to begin with, and the pairing Or-opt method next.

Results of experiments are shown in Table 4 and Table 5.

5 Conclusions

We proposed some heuristic algorithms for the many-to-many dial-a-ride routing problem and compared them with the algorithms in literature.

Our conclusions are summarized as follows.

1. Among the construction heuristics, one of the proposed heuristics named the modified pairing insertion (MPI) method performs best. If we compare with the asymptotically optimal value (when $n = 101$, this value is about 100) derived by Stein [19],[20], then the modified pairing insertion method could find a dial-a-ride tour within about 6% optimality. This ratio is identical to that of the best construction method (the farthest insertion, or the convex hull insertion method) for the TSP [7].

2. The local search procedures with random starting solutions produce worse solutions, and require much more computational time than the MPI method. Therefore we should use the construction method when a reasonable effective solutions are desired.

3. .When we start with random solutions (or solutions obtained using MST heuristic), Psaraftis' 3-opt procedure performs best. Whereas when we start with solutions obtained using the MPI method, the composite Or-opt procedure 1 performs best. We conjecture that the 3-opt (or 2-opt) method works better when the starting solution is not near-optimal, while the composite Or-opt (or modified, paring) method works better when the starting solution is not so bad.

4. Both 3-opt and composite Or-opt procedures give a dial-a-ride tour within about 1% of optimality with high regularity if we start with the solutions of the MPI method.

5. The composite Or-opt 2 is slower than and operates with approximately the same effectiveness as the the composite Or-opt 1 (the the composite Or-opt 1 is somewhat better). Thus we recommend the composite Or-opt 1.

To confirm the above conclusions we should compare the upper bounds obtained using several heuristic algorithms with the optimal solutions (or lower bounds). The most important future direction in this area is to develop an efficient exact algorithm or the lower bounding procedure.

Acknowledgments

We wish to thank the anonymous referees for their valuable comments. This work was partially supported by the Waseda University Grant for Special Research Projects: 63A-42 and 89A-74.

References

[1] A. V. Aho, J. E. Hopcroft and J. D. Ulman, "The Design and Analysis of Computer Algorithms," Addison-Wesley (1974).

[2] J. J. Bartholdi and L. K. Platzman,"An O(n log n) Planar Traveling Salesman Heuristic Based on Spacefilling Curves," Operations Research Letters, Vol. 5 (1986) pp.165-169.

[3] L. Bodin and B. Golden, "Classification in Vehicle Routing and Scheduling," Networks, Vol. 11 (1981) pp. 97-108.

[4] P. L. Cassidy and H. S. Bennett , "TRAMP - A Multi-Depot Vehicle Scheduling System," Operations Research Quarterly, Vol. 23 (1972) pp. 151-163.

[5] C. F. Daganzo,"Am Approximate Analytic Model of Many-to-Many Demand Responsive Transportation Systems," Transportation Research, Vol. 12 (1978) pp. 325-333.

[6] M. R. Garey and D. S. Johnson ,"Computers and Intractability : A Guide to the Theory of NP-Completeness," Bell Telephone Labo. (1979).

[7] B. Golden, L. Bodin, T. Doyle and W. Stewart JR. ,"Approximate Traveling Salesman Algorithm," Operations Research, Vol. 28 (1980) pp. 694-711.

[8] J. Jaw, A. R. Odoni, H. N. Psaraftis and N. H. M. Wilson ,"A Heuristic algorithm for the Multi-Vehicle Advance Request Dial-A-Ride Problem with Time Windows," Transportation Research, Vol. 20B (1986) pp. 243-257.

[9] B. Kalantari, A. V. Hill and S. R. Arora , " An Algorithm for the Traveling Salesman Problem with Pickup and Delivery Customer," European Journal of Operations Research, Vol. 22 (1985) pp. 377-386.

[10] R. Karp,"Probabilistic Analysis of Partitioning Algorithms for the Traveling Salesman in the Plane," Mathematics of Operations Research, Vol.2 (1977) pp.209-224.

[11] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys,"The Traveling Salesman Problem," John Wiley and Sons (1985).

[12] J. D. C. Little, K. G. Murty, D. M. Sweeney and C. Karel ,"An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11 (1963) pp. 972-989.

[13] S. Lin and W. Kernighan ,"An Effective Heuristic Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 21 (1973) pp. 498-516.

[14] M. W. Padberg and S. Hong," On the Symmetric Traveling Salesman Problem: A Computational Study," Mathematical Programming Vol. 7 (1980) pp. 78-107.

[15] H. Psaraftis ,"A Dynamic Programming Solution to the Single Vehicle Many-to-Many Intermediate Request Dial-A-Ride Problem," Transportation Science, Vol. 2 (1980) pp. 130-154.

[16]] H. Psaraftis ,"Analysis of an $O(N^2)$ Heuristic for the Single Vehicle Many-to-Many Euclidean Dial-A-Ride Problem," Transportation Science, Vol. 117B (1983) pp. 133-145.

[17] H. Psaraftis ,"K-interchange Procedures for Local Search in a Precedence Constrained Routing Problem," European Journal of Operations Research, Vol. 13 (1983) pp. 391-402.

[18] T. R. Sexton," The Single Vehicle Many to Many Routing and Scheduling Problem," Ph. Doc. Thesis, State University of New York (1979).

[19] D. Stein ,"Scheduling Dial-A-Ride Transportation System," Transportation Research, Vol. 12 (1978) pp. 232-249.

[20] D. Stein ,"An Asymptotic, Probabilistic Analysis of a Routing Problem," Mathematics of Operations Research, Vol. 3 (1978) pp. 89-101

[21] K. J. Supowit, E. M. Reingold and D. A. Plisted, "The Traveling Salesman Problem," SIAM Journal of Comput., Vol. 12 (1983) pp. 144-156.

[22] H. Suzuki and S. Nomura ,"A Shortest Path Problem with Visiting Order Constraints (In Japanese)," Proceedings of the 7th Mathematical Programming Symposium Japan (1986) pp. 127-142.

[23] R. E. Tarjan ,"Data Structures and Network Algorithms," Bell Labo. (1983).

[24] A. Wren and A. Holiday ,"Computer Scheduling of Vehicle from One or More Depot to a Number of Delivery Point," Operations Research Quarterly, Vol. 23 (1972) pp. 333-344.

Mikio KUBO : Department of
Industrial Engineering and
Management, Waseda
University, 3-4-1, Okubo
Shinjuku, Tokyo 169, Japan