アルゴリズム 63－3
（1998．7．22）

木構造ネットワーク上の車両配送計画問題

濱口真也[1]，加藤直樹[2]

[1] 大阪府 総務部 情報システム推進課
[2] 京都大学大学院工学研究科建築学専攻

本論文では単一デポを有する木構造ネットワーク上の配送計画問題を考察する。顧客は，木の頂点上に位置し，正の需要を持っている。同一の積載量制限のある車両群がデポを出発し，顧客の需要を満たしながらデポに戻って来る。また，一人の顧客の需要は分割可能であるとする（二台以上の車両によってその顧客の総需要を満たすことが許される）。このとき，全顧客の需要を満たす車両配送計画のなかで総走行距離を最小する配送計画を求める問題である。本論文はこの問題がNP-完全であることを示すとともに、近似比1.5の近似アルゴリズムを提案する。

A Capacitated Vehicle Routing Problem on a Tree

Shin-ya Hamaguchi[1] and Naoki Katoh[2]

[1] Information System Promotion Division, Osaka Prefectural Government
[2] Department of Architecture and Architectural Systems, Kyoto University

In this paper we deal with a vehicle routing problem on a tree-shaped network with a single depot. Customers are located on vertices of the tree, and each customer has a positive demand. Demands of customers are served by a fleet of identical vehicles with limited capacity. It is assumed that the demand of a customer is splittable, i.e., it can be served by more than one vehicle. The problem we are concerned with in this paper asks to find a set of tours of the vehicles with minimum total lengths. Each tour begins at the depot, visits a subset of the customers and returns to the depot without violating the capacity constraint. We show that the problem is NP-complete and propose a 1.5-approximation algorithm for the problem. We also give some computational results.

1 Introduction

In this paper we consider a capacitated vehicle routing problem on a tree-shaped network with a single depot. Let $T = (V, E)$ be a tree, where V is a set of n vertices and E is a set of edges, and $r \in V$ be a designated vertex called *depot*. Nonnegative weight $w(e)$ is associated with each edge $e \in E$, which represents the length of e. Customers are located at vertices of the tree, and a customer at $v \in V$ has a positive demand $D(v)$. Thus, when there is no customer at v, $D(v) = 0$ is assumed. Demands of customers are served by a set of identical vehicles with limited capacity. We assume throughout this paper that the capacity of every vehicle is equal to one, and that the demand of a customer is splittable, i.e., it can be served by more than one vehicle. Each vehicle starts at the depot, visits a subset of customers to (partially) serve their demands and returns to the depot without violating the capacity constraint. The problem we deal with in this paper asks to find a set of tours of vehicles with minimum total lengths to satisfy all the demands of customers. We call this problem TREE-CVRP.

Vehicle routing problems have long been studied by many researchers (see [5, 6, 7, 11, 12] for a survey), and are found in various applications such as scheduling of truck routes to deliver goods from a warehouse to retailers, material handling systems and computer communication networks. Recently, AGVs (automated guided vehicle) and material handling robots are often used in manufacturing systems, but also in offices and hospitals, in order to reduce the material handling efforts. The tree-shaped network can be typically found in buildings with simple structures of corridors and in simple production lines of factories.

Vehicle scheduling problems on tree-shaped networks have recently been studied by several authors [3, 8, 9, 13, 16]. All of them treated in the literature deal with a single-vehicle scheduling that seeks to find an optimal tour under certain constraints other than those treated in this paper. To the authors' knowledge, TREE-CVRP has not been studied in the literature. For general undirected networks, the problem contains TSP (traveling salesman problem) as a special case, and thus it is not only NP-hard but APX-hard (except the result by [2] for Euclidean-CVRP in the plane). However, when restricted to tree-shaped networks, the complexity of TREE-CVRP is not clear. We shall show that TREE-CVRP is strongly NP-complete by a reduction from *bin-packing problem* (the proof is omitted).

Thus, we turn our attention on developing approximate algorithms for the problem. Since TREE-CVRP is a special class of general CVRP, approximation algorithms originally developed for CVRP on general undirected networks can be used to approximately solve TREE-CVRP. In particular, the iterated tour partitioning (ITP) heuristic proposed by Haimovich and Rinooy Kan [14] and Altinkemer and Gavish [1] provides $1 + (1 - \frac{1}{k})\alpha$ approximation for such general CVRP when α-approximation algorithm for TSP is available, where the capacity of every vehicle is assumed to be equal to k and the demand of every customer is a positive integer. For instance, if the famous 1.5-approximate algorithm for TSP by Christofides [4] is used, the approximation factor becomes $2.5 - 1.5/k$. For tree-shaped networks, TSP can be optimally solved in a straightforward manner, and thus the direct consequence of [1, 14] results in a $(2 - \frac{1}{k})$-approximation algorithm.

In this paper, we shall present an improved 1.5-approximation algorithm for TREE-CVRP by exploiting the tree structure of the network. The basic idea behind the algorithm is to (i) first find an appropriate subtree, to (ii) generate two routings to partially serve the demands of customers in the subtree by applying two different heuristics, and to (iii) choose the better one. We repeat this process until all the demands are served. We have implemented our algorithm and carried out computational experiments to see how effective our algorithm is. Compared with lower bounds, the results demonstrate that the solutions obtained by our algorithm are very close to optimal.

The rest of the paper is organized as follows. In Section 2, we give some necessary definitions and basic facts. Section 3 presents an algorithm and proves that its approximation ratio is 1.5. Section 4 presents an example for which the approximation factor becomes 1.25. Section 5 reports computational results. Section 5 concludes the paper.

2 Preliminaries

Since T is a tree, there exists a unique path between two vertices. For vertices $u, v \in V$, let $path(u, v)$ be the unique path between u and v. The length of $path(u, v)$ is denoted by $w(path(u, v))$. We often view T as a directed tree rooted at r. For a vertex $v \in V - \{r\}$, let $parent(v)$ denote the parent of v, and $C(v)$ the set of children of v. We assume throughout this paper that when we write an edge $e = (u, v)$, u is a parent of v unless otherwise stated. For any $v \in V$, let T_v denote the subtree rooted at v, and $w(T_v)$ and $D(T_v)$ denote the sum of weights of edges in T_v, and the sum of demands of customers in T_v, respectively. Since customers are located on vertices, customers are often identified with vertices.

Suppose that we are given a set $S \subset V - \{r\}$ with $\sum_{v \in S} D(v) \leq 1$. Then one vehicle is enough to serve all the demands of customers in S, and an optimal tour for S can be trivially obtained by first computing a minimal subtree T' that spans $S \cup \{r\}$ and by performing a depth-first search with r as the starting vertex. Thus, when we speak of a tour in this paper, we do not need explicitly give a sequence of vertices that a vehicle visits, but it is enough to specify a set of customers that the vehicle visits. Since the demand of a customer is splittable, in order to define a tour of a vehicle, we need to specify the amount of demand of each customer served by the vehicle.

A solution of TREE-CVRP consists of a set of tours. From the above discussion, we represent the tour of the j-th vehicle by

$$\{D_j(v) \mid v \in S_j\}, \tag{1}$$

where S_j is the set of customers for which some positive demands are served in the j-th tour, and $D_j(v)(> 0)$ for $v \in S_j$ is the amount of demand that the j-th vehicle serves at v. often referred to as the *optimal cost.*

For an edge $e = (u, v)$, let

$$LB(e) = 2w(e) \cdot \lceil D(T_v) \rceil. \tag{2}$$

$LB(e)$ represents a lower bound of the cost required for traversing edge e in an optimal solution because, due to the unit capacity of a vehicle, the number of vehicles required for any solution to serve the demands in T_v is at least $\lceil D(T_v) \rceil$ and each such vehicle passes e at least twice (one is in a forward direction and the other is in a backward direction). Thus, we have the following lemma.

Lemma 2.1 $\sum_{e \in E} LB(e)$ *gives a lower bound of the optimal cost of TREE-CVRP.*

3 Algorithm

A vertex $v \in V$ is called *D-feasible* if $D(T_v) \geq 1$ and is called *D-minimal* if it is *D-feasible* but any of its children is not. The proposed algorithm first finds a *D-minimal* vertex, and determines a routing of one or two vehicles that partially serve demands of vertices in T_v by applying Strategy 1 or 2 depending the cases as will be described below. We then appropriately update the remaining demands of vertices visited by the routing currently determined. In addition, if the remaining demand of subtree T_v becomes zero, T_v as well as the edge $(parent(v), v)$ is deleted. In this section, we abuse the notations $D(v)$ and $D(T_v)$ to denote the remaining demands of vertex v or subtree T_v, respectively unless confusion occurs. We repeat this process until all the demands of T are served. Notice that, if there is no D-feasible vertex (i.e., $D(T_r) < 1$) any more, we can optimally serve the remaining demands by visiting the relevant vertices in a depth-first manner.

The algorithm consists of a number of applications of Strategy 1 or 2. One application of Strategy 1 or 2 is called a *round.*

When a D-minimal vertex v is found, in order to determine the routing of one or two vehicles to partially serve the demands in T_v, we apply the following two strategies and choose the one with cheaper cost. Let $C(v) = \{v_1, v_2, \ldots, v_p\}$. We assume $D(T_{v_i}) > 0$ for every $v_i \in C(v)$ since otherwise such subtree can be eliminated from T. Let $S \subseteq C(v)$ satisfying $1 \leq \sum_{v_i \in S} D(T_{v_i}) < 2$. Since $D(T_{v_i}) < 1$ for all $v_i \in C(v)$ from D-minimality of v, such S always exists and can be easily computed as $\cup_{i=1}^{k} T_{v_i}$ satisfying

$$\sum_{i=1}^{k-1} D(T_{v_i}) < 1 \quad \text{and} \quad \sum_{i=1}^{k} D(T_{v_i}) \geq 1. \tag{3}$$

If $\sum_{i=1}^{k} D(T_{v_i}) = 1$, we simply allocate one vehicle to serve all the demands in $\cup_{i=1}^{k} T_{v_i}$. Thus, we assume otherwise. For the ease of the exposition of the algorithm, we assume $k = 2$ without loss of generality because otherwise we can equivalently modify T_v by creating a fictitious vertex v' of $D(v') = 0$ as well as edge (v, v') of zero weight and replacing edges (v, v_i) for i with $1 \leq i \leq k-1$ by (v', v_i) with $w(v', v_i) = w(v, v_i)$ and an edge (v, v') with zero weight.

With this assumption, the algorithm considers subtrees T_{v_1} and T_{v_2} satisfying

$$D(T_{v_1}) < 1, D(T_{v_2}) < 1 \quad \text{and} \quad D(T_{v_1}) + D(T_{v_2}) > 1. \tag{4}$$

The first strategy (Strategy 1) prepares two vehicles to serve all the demands in $T_{v_1} \cup T_{v_2}$, while the second strategy (Strategy 2) prepare one vehicle to partially serve the demands in $T_{v_1} \cup T_{v_2}$ by using its full capacity (thus, the demand of some vertex may possibly be split).

Strategy 1: We prepare one vehicle for T_{v_1} and another vehicle for T_{v_2} to separately serve demands of T_{v_1} and T_{v_2}. The cost to serve these demands is

$$4w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2}) \tag{5}$$

because two vehicles run on the path $path(r,v)$ but each of T_{v_1} and T_{v_2} is traversed by only one vehicle.

Strategy 2: We assume $w(T_{v_1}) \geq w(T_{v_2})$ without loss of generality. We split the demand $D(u)$ of every $u \in T_{v_2}$ into $D'(u)$ and $D''(u)$ so that

$$\sum_{u \in T_{v_1}} D(u) + \sum_{u \in T_{v_2}} D'(u) = 1. \tag{6}$$

We allocate one vehicle to serve the set of demands

$$\{D(u) \mid u \in T_{v_1}\} \text{ and } \{D'(u) \mid u \in T_{v_2}\}.$$

The computation of such $D'(u)$ satisfying (6) is straightforward, i.e., it is done by performing a depth-first search on T_{v_2} with v_2 as a starting vertex in such a way that

(1) Initially set $sum = \sum_{u \in T_{v_1}} D(u)$.

(2) Every time a new vertex u is visited, if $sum + D(u) \leq 1$ holds, we set $D'(u) = D(u)$, $D''(u) = 0$ and increment sum by $D(u)$, otherwise we set $D'(u) = 1 - sum$, $D''(u) = D(u) - D'(u)$ and increment sum by $D'(u)$.

Notice that the demand of at most one vertex is split by this procedure. The cost required for Strategy 2 is at most

$$2w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2}). \tag{7}$$

Demands of a subset of vertices in T_{v_2} remain unserved, and thus T_{v_2} (or its subgraph) will be visited later by other vehicles. Thus, in total the cost to visit T_{v_2} (or its subgraph) will be counted twice or more as (7). For the ease of the analysis of approximation ratio of the proposed algorithm, we amortize the the cost to visit T_{v_2} in the current round so that it is charged to T_{v_1}. Since $w(T_{v_1}) \geq w(T_{v_2})$, the cost of (7) is bounded from above by

$$2w(path(r,v)) + 4w(T_{v_1}). \tag{8}$$

We consider (8) as the cost for Strategy 2. An alternative interpretation of (8) is that a set of demands defined by $\{D'(u) \mid u \in T_{v_2}\}$ are served without visiting T_{v_2} (resp. T_{v_1}) at the expense of visiting T_{v_1} twice. Notice that the subtree T_{v_1} will never be visited in future rounds because all the demands therein are served in the current round.

It should be remarked that our algorithm chooses Strategy 1 or 2 not by directly comparing the costs of (5) and (8), but by the following rule.

Selection rule of Strategy 1 or 2:

We apply Strategy 1 if

$$\frac{4w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2})}{2w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2})} \leq \frac{2w(path(r,v)) + 4w(T_{v_1})}{2w(path(r,v)) + 2w(T_{v_1})}, \tag{9}$$

and apply Strategy 2 otherwise.

The rationale behind this selection rule is as follows: Since the amounts of demands as well as the sets of customers served by Strategies 1 and 2 are different in general, it may not be fair to directly compare (5) and (8), but it is reasonable to compare the costs of (5) and (8) divided by the lower bounds of the costs to optimally execute their corresponding tasks. In fact, the denominators of the left-hand and right-hand sides of (9) stand for such lower bounds as will be seen below.

Now we shall prove the following main theorem.

Theorem 3.1 *The approximation of our algorithm for TREE-CVRP is* 1.5.

PROOF. The proof is done by induction on the number of rounds. When $D(T_r) \leq 1$, our algorithm trivially finds an optimal solution. This proves the base case of the induction.

Assuming that our algorithm computes a 1.5-approximate solution for problems that require at most k rounds, we prove that the theorem also holds for the case which requires $k+1$ rounds. Let P denote the problem instance of TREE-CVRP for which our algorithm requires $k+1$ rounds. Let us consider the first round for P in which a D-minimal subtree T_v is found. As we remarked earlier, let us assume without loss of generality that the algorithm considers two T_{v_1} and T_{v_2} satisfying (4) and that $w(T_{v_1}) \geq w(T_{v_2})$. Depending whether (9) holds or not, Strategy 1 or 2 is applied by which one or two tours are determined to serve the demands of customers $\in T_{v_1} \cup T_{v_2}$. Let P' be the problem instance of TREE-CVRP obtained from P after the first round by decreasing demands served in this round from original $D(\cdot)$. Let d_{sum} denote the total amount of demands served in this round. Thus, for each edge $e = (x, y) \in path(r, v)$, the remaining demand $D_{remain}(T_y)$ for subtree T_y satisfies

$$D_{remain}(T_y) = D(T_y) - d_{sum}. \tag{10}$$

Notice that $d_{sum} \geq 1$ holds. Let $d(u)$ denote the demand of $u \in T_{v_1} \cup T_{v_2}$ served by this round, and let $D_{remain}(u) = D(u) - d(u)$ for $u \in T_{v_1} \cup T_{v_2}$. Let $LB(P')$ be the lower bound for the problem P'. From Lemma 2.1, we then have

$$LB(P') = 2\Big(\sum_{\substack{e \in path(r,v) \text{ or} \\ e \in T_{v_1} \cup T_{v_2}}} \lceil D_{remain}(e) \rceil w(e) + \sum_{\substack{e \notin path(r,v) \text{ and} \\ e \notin T_{v_1} \cup T_{v_2}}} \lceil D(e) \rceil w(e) \Big). \tag{11}$$

Let $cost(P)$, $cost_1$ and $cost(P')$ denote the total cost required for the original problem P by our algorithm, the cost required by the first round and the cost for the remaining problem P' to be required by our algorithm, respectively, (i.e., $cost(P) = cost_1 + cost(P')$). Let $LB(P)$ denote the lower bound of the optimal cost for P given by Lemma 2.1.

(Case 1): (9) holds. Thus, the algorithm applies Strategy 1. Since $\lceil x \rceil \geq 1 + \lceil x - a \rceil$ holds in general for any positive x, a with $x \geq a$ and $a \geq 1$, it follows from (10) and $d_{sum} \geq 1$ that $LB(e) \geq 1 + LB_{remain}(e)$ holds for each $e \in path(r, v)$, where $LB_{remain}(e)$ is a lower bound of the optimal cost of P' involving edge e defined by (2). Also, $LB(e) = 1$ holds for all $e \in T_{v_1} \cup T_{v_2}$ from $D(T_{v_1}) < 1$ and $D(T_{v_2}) < 1$, and $D_{remain}(u) = 0$ holds for all $u \in T_{v_1} \cup T_{v_2}$ from the way of Strategy 1. Thus, we have

$$LB(P) \geq 2w(path(r, v)) + 2w(T_{v_1}) + 2w(T_{v_2}) + LB(P'). \tag{12}$$

From (5), we have

$$cost_1 = 4w(path(r, v)) + 2w(T_{v_1}) + 2w(T_{v_2}).$$

Thus,

$$\begin{aligned} \frac{cost(P)}{LB(P)} &\leq \frac{cost_1 + cost(P')}{2w(path(r, v)) + 2w(T_{v_1}) + 2w(T_{v_2}) + LB(P')} \\ &= \frac{4w(path(r, v)) + 2w(T_{v_1}) + 2w(T_{v_2}) + cost(P')}{2w(path(r, v)) + 2w(T_{v_1}) + 2w(T_{v_2}) + LB(P')}. \end{aligned} \tag{13}$$

Since $cost(P')/LB(P') \leq 1.5$ holds from the induction hypothesis, it suffices to prove

$$\frac{4w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2})}{2w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2})} \leq 1.5. \tag{14}$$

Since

$$\min\{\frac{b}{a}, \frac{d}{c}\} \leq \frac{b+d}{a+b} \tag{15}$$

holds for any positive a, b, c, d, it follows from (9) that

$$\frac{4w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2})}{2w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2})} \leq \frac{6w(path(r,v)) + 6w(T_{v_1}) + 2w(T_{v_2})}{4w(path(r,v)) + 4w(T_{v_1}) + 2w(T_{v_2})} \leq 1.5.$$

Thus, (14) holds, and hence $cost(P)/LB(P) \leq 1.5$ follows from (13).

(Case 2): (9) does not hold, i.e,

$$\frac{4w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2})}{2w(path(r,v)) + 2w(T_{v_1}) + 2w(T_{v_2})} > \frac{2w(path(r,v)) + 4w(T_{v_1})}{2w(path(r,v)) + 2w(T_{v_1})}, \tag{16}$$

i.e., Strategy 2 is applied. In this case, $d_{sum} = 1$ holds since one vehicle is scheduled to serve demands in $T_{v_1} \cup T_{v_2}$ with its full capacity. Thus, from (10), $LB(e) = 1 + LB_{remain}(e)$ holds for each $e \in path(r,v)$. Since all the demands of T_{v_1} are served in this round from the way of Strategy 2, $D_{remain}(u) = 0$ holds for all $u \in T_{v_1}$. Thus, from (2) we have

$$LB(P) \geq 2w(path(r,v)) + 2w(T_{v_1}) + LB(P'). \tag{17}$$

From (8), we have

$$cost_1 = 2w(path(r,v)) + 4w(T_{v_1}).$$

Thus,

$$\begin{aligned} \frac{cost(P)}{LB(P)} &\leq \frac{cost_1 + cost(P')}{2w(path(r,v)) + 2w(T_{v_1}) + LB(P')} \\ &= \frac{2w(path(r,v)) + 4w(T_{v_1}) + cost(P')}{2w(path(r,v)) + 2w(T_{v_1}) + LB(P')}. \end{aligned} \tag{18}$$

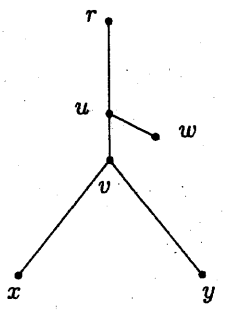
Since $cost(P')/LB(P') \leq 1.5$ holds from the induction hypothesis, it suffices to prove

$$\frac{2w(path(r,v)) + 4w(T_{v_1})}{2w(path(r,v)) + 2w(T_{v_1})} \leq 1.5. \tag{19}$$

Using (15) again, we can prove (19) in a manner similar to Case 1. Thus, $cost(P)/LB(P) \leq 1.5$ follows.

□

Although the details are omitted, it is easy to see that the algorithm can be implemented in such a way that each round runs in linear time. Thus, the total running time of our approximate algorithm is $O(\sum_{v \in V} D(v) \cdot n)$ time since the number of rounds required by the algorithm is clearly $O(\sum_{v \in V} D(v))$. This is polynomial if $\sum_{v \in V} D(v)$ is polynomial in n.



$$w(r,u) = 1 - \epsilon$$
$$w(u,v) = w(u,w) = \epsilon$$
$$w(v,y) = w(v,y) = 1$$

$$D(r) = D(u) = D(v) = 0$$
$$D(w) = D(x) = D(y) = 0.6$$

Figure 1: Illustration of the worst case example

4 Lower Bound

We shall show that the approximation ratio of the proposed algorithm is at least 1.25 by giving such an example which is illustrated in Figure 1. In the figure, ϵ is a sufficiently small positive constant.

First of all, the algorithm finds a D-minimal vertex v and chooses Strategy 1 since (9) holds, and the cost of the first round required by the algorithm is 8. There still remains a demand of 0.6 at w, which requires cost of 2. Thus, the total cost is 10. On the other hand, the optimal schedule is that the first vehicle serves the demand of 0.6 at x and on the way back to the depot, visits vertex w to serve the demand of 0.3, which requires cost $4 + 2\epsilon$ and that the second serves the remaining demands with the same cost $4 + 2\epsilon$. Thus, the optimal cost is $8 + 4\epsilon$. Therefore, the approximation ratio for this example is 1.25.

5 Computational Results

We have implemented our algorithm and the ITP heuristic developed by [1, 14] in order to see the practical performance of our algorithm by comparing with ITP heuristic. For this, we have randomly generated 10 problem instances for each of $n = 50, 100, 150, 200, 250, 300$ (we have omitted the description of the generation scheme we sdopted).

The performance of the algorithms are evaluated by comparing the ratios of costs of solutions produced by the algorithms to the lower bound of (??). The overall average of such ratio of our algorithm is 1.016, while that of ITP is 1.113. The worst-case ratio among 60 instances is 1.072 for our algorithm while that for ITP heuristic is 1.238. For 37 cases out of 60 instances generated, our algorithm produced solutions that match the lower bound, while there was only one such case for ITP. From this experiments, we can observe that our algorithm is superior to ITP heuristic, and that it practically produces solutions very close to optimal. More extensive computational experiments are reported in [15].

References

[1] K. Altinkemer and B. Gavish, Heuristics for delivery problems with constant error guarantees, *Transportation Science*, 24 (1990), 294-297.

[2] T.Asano, N.Katoh, H.Tamaki and T.Tokuyama, Covering points in the plane by k-tours.: towards a polynomial time approximation scheme for general k, *Proc. of 29th Annual ACM Symposium on Theory of Computing*, pp.275-283, 1997.

[3] I. Averbakh and O. Berman, Sales-delivery man problems on treelike networks, *Networks*, 25 (1995), 45-58.

[4] N. Christofides, Worst-case analysis of a new heuristic for the traveling salesman problem, Report 388, Graduate School of Industrial Administration, 1976.

[5] N. Christofides, A. Mingozzi and P. Toth. The vehicle routing problem. in: N. Christofides, A. Mingozzi, P. Toth and C. Sandi, editors. *Combinatorial Optimization.* John Wiley & Sons Ltd, London,1979.

[6] M. Desrochers, J. K. Lenstra and M. W. P. Savelsbergh. A classification scheme for vehicle routing and scheduling problems. *Eur. J. Oper. Res.* 46, 322–332, 1990.

[7] M.L. Fischer. Vehicle Routing. in *Network Routing,* Handbooks in Operations Research and Management Science, **8**, Ball, M. O., T. L. Magnanti, C. L. Monma and G. L. Nemhauser (Eds.), Elsevier Science, Amsterdam, 1-33, 1995.

[8] G. Frederickson, Notes on the complexity of a simple transportation problem, *SIAM J. Computing*, 22-1 (1993), 57-61.

[9] G. Frederickson and D. Guan, Preemptive ensemble motion planning on a tree, *SIAM J. Computing*, 21-6 (1992), 1130-1152.

[10] M.R.Garey , D.S.Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[11] B.L. Golden. Introduction to and Recent Advances in Vehicle Routing Methods. in *Transportation Planning Models,* M. Florian (Ed), Elsevier Sci. Publ. B. V. (North Holland), 383-418, 1984.

[12] B.L. Golden and A. A. Assad (Eds.). *Vehicle Routing: Methods and Studies,* Studies in Manag. Science and Systems 16, North-Holland Publ., Amsterdam, 1988.

[13] Y. Karuno, H. Nagamochi, T. Ibaraki, Vehicle Scheduling on a tree with Release and Handling Times, Proc. of ISAAC'93, *Lecture Notes in Computer Science 762, Springer-Verlag* 486-495, 1993.

[14] M. Haimovich and A.H.G. Rinnooy Kan. Bounds and Heuristics for capacitated routing problems *Mathematics of Operations Research*, **10**(4), 527-542, 1985.

[15] S. Hamaguchi, A study on Vehicle Routing Problems on Tree-shaped Networks, Master Thesis, Graduate School of Business Administration, Kobe Univ. of Commerce, 1998.

[16] Y. Karuno, H. Nagamochi, T. Ibaraki, Vehicle Scheduling on a tree to Minimize Maximum Lateness, *Journal of the Operations Research Society of Japan*, Vol. 39, No.3 (1996) 345-355.

[17] G.Laporte. The Vehicle Routing Problem : An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59 (1992) 345-358.

[18] J.K.Lenstra and A.H.G.Rinooy Kan. Complexity of Vehicle Routing and Scheduling Problem. *Networks*, 11 (1981) 221-227.

[19] K. Lund VRP References - Part I/II. WWW home page, http : //www.imm.dtu.dk/documents/users/kl/vrp_1.html