アルゴリズム 68−2
(1999. 5. 10)

木構造ネットワーク上の車両配送計画問題の新しい近似解法

浅野哲夫 [1], 加藤直樹 [2], 川島一浩 [1]

[1] 北陸先端科学技術大学院大学情報科学研究科
[2] 京都大学大学院工学研究科建築学専攻

あらまし 本文では，単一の出発点をもった木状のネットワーク上での車両配送問題に対する新たな近似アルゴリズムを提案する．顧客は木の頂点上に配置されており，各顧客は正の需要を持っているものとする．顧客の需要は，容量に制限のある同一の車両を用いて満たさなければならないが，需要を分割することは許されていると仮定する．すなわち，ある顧客の需要を複数台数の車両によって満たしてもよい．このとき，配送に要する総距離を最小にするにはどうすればよいかを求めるのが問題である．どの車両も配送センターから出発して，積載制限に違反しないように荷物を積んで再びセンターに戻る．従来，この問題に対して 1.5-近似アルゴリズムが知られているが，本文では近似率を 1.35 に改善することに成功した．

A New Approximation Algorithm for the Capacitated Vehicle Routing Problem on a Tree

Tetsuo Asano[1], Naoki Katoh[2], and Kazuhiro Kawashima[1]

[1]School of Information Science, JAIST,
Asahidai, Tatsunokuchi, 923-1292, Japan
{t-asano, kawasima}@jaist.ac.jp.
[2]Department of Architecture and Architectural Systems, Kyoto University,
Yoshida-Honmachi, Sakyou-ku, Kyoto, 606-8501 Japan
naoki@is-mj.archi.kyoto-u.ac.jp

Abstract This paper presents a new approximation algorithm for 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 propose a 1.35-approximation algorithm for the problem, which is an improvement over the existing 1.5-approximation.

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 [2, 3, 4, 7, 8] 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 [1, 5, 6, 10, 11, 12]. Most of them dealt with a single-vehicle scheduling that seeks to find an optimal tour under certain additional constraints.

However, TREE-CVRP has not been studied in the literature until very recently. Last year Hamaguchi and Katoh [9] proved its NP-hardness and proposed a 1.5-approximation algorithm ([12] considered the variant of TREE-CVRP where demand of each customer is not splittable and gave 2-approximation algorithm.)

In this paper, we shall present an improved 1.35-approximation algorithm for TREE-CVRP by exploiting the tree structure of the network. This is an improvement of the existing 1.5-approximation algorithm by Hamaguchi and Katoh [9]. A basic idea behind the improvement is the use of reforming operations preserving the lower bound on the cost, which simplifies the analysis.

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. The total tour length of an optimal solution for TREE-CVRP is 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 1 $\sum_{e \in E} LB(e)$ *gives a lower bound of the optimal cost of TREE-CVRP.*

3 Reforming Operations

Our approximation algorithm repeats the following two steps until all the demands are served. In the reforming step we reshape a given tree following six different operations all of which are "safe" in the sense that they do not increase the lower bound on the cost. The second step is a selection step in which we choose an appropriate subtree and choose among a few possible strategies depending on the cases the best one to serve the demands in the subtree.

The first reforming operation R_1 is applicable when some nodes have demands greater than or equal to 1. Suppose that a node v has a demand $D(v) \geq 1$. Then, we allocate $k = \lfloor D(v) \rfloor$ vehicles to v to serve k units of its demand (integral part of the demand). This operation results in demand at v less than one. Note that this operation is apparently safe by the argument based on the lower bound on the tour cost. Thus, it is reasonable to assume that each demand is less than one.

The second operation R_2 is to remove positive demand from each internal node. If there is any internal node v with positive demand, we create a new node connected with v by an edge of weight zero and descend the weight of v to the new node. It is easy to see that any tour on an original tree can be transformed into another tour on the tree reshaped as above with just the same cost. It implies that this reform is safe, in other words it does not affect the lower bound. Therefore, we can assume that positive demand is placed only at leaves, that is, demand at any internal node is zero.

The third reform R_3 is applied to a pair of nodes (u, v) such that a leaf v is a unique child of u. The node u has no other children. In this case we contract the edge (u, v), i.e., delete (u, v) and the node v, after replacing the demand $D(u)$ at u by $D(u) + D(v)$ and then increasing the cost of the edge to u by the cost $w(u, v)$ of the edge between u and v. If the resulting node u has demand ≥ 1, then we can apply the operation R_1 to u to reduce the integral part of its demand.

To define more essential reformation for approximation algorithm, we need some more assumptions and definitions.

A node v of a tree T is called a **p-node** if
(i) v is an internal node, and
(ii) all of the children of v are leaves, and
(iii) the sum of the demands at those children are less than 2.

A node u is called a **q-node** if
(i) the sum of the demands in the subtree T_u rooted at u, denoted by $D(T_u)$, is at least 2, and
(ii) no child of u has the property (i).

The fourth reforming operation R_4 is to merge leaves of p-nodes and q-nodes. Let u be a p-node or q-node and $\{v_1, v_2, \ldots, v_k\}$ be a set of its children (leaves by definition). By w_i we denote the weight of the edge between u and v_i. We examine every pair of leaves. For the pair (v_i, v_j) we check whether the sum of their weights exceeds 1. If $D(v_i) + D(v_j) \leq 1$, then we

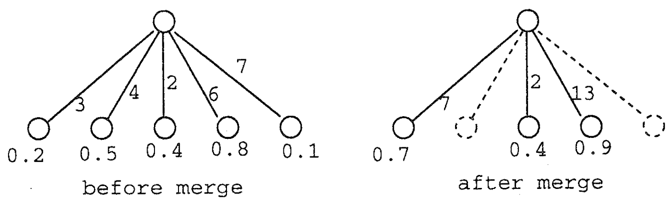
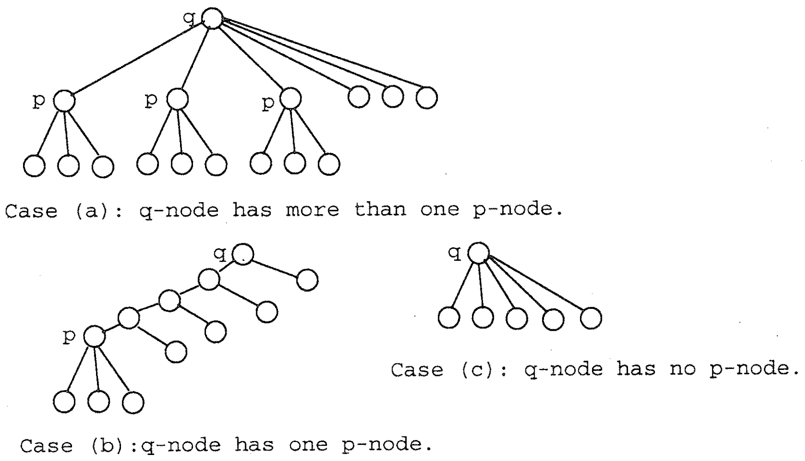


Figure 1: The merging operation.



q
p
p
p
Case (a): q-node has more than one p-node.
q
p
Case (b):q-node has one p-node.
q
Case (c): q-node has no p-node.

Figure 2: The three cases for a q-node.

merge them. Exactly speaking, we remove the leaf v_j together with its associated edge (u, v_j) after replacing the demand of v_i with $D(v_i) + D(v_j)$ and the weight w_i with $w_i + w_j$. Then, we proceed to the next unexamined pair of leaves. We repeat this process while there is any mergeable pair of leaves. Figure 1 illustrates how this merging process proceed.

An important property of the resulting tree is that any p-node has at most three children (leaves) after the reforming operation since otherwise the sum of the demands of those children exceeds 2, which causes a contradiction to the definition of a p-node. Thus, we can assume that any p-node has at most three children. It may happen that all the leaves of a p-node are merged into one leaf. In this case, we apply the contraction operation R_3.

Now, after applying the reform R_4 to each p-node, a subtree rooted at a q-node may include more than one p-node. But, in that case those p-nodes must be children of the q-node. Otherwise, an internal node having more than one p-node in its descendants must have demand exceeding 2 in its descendants, which contradicts to the fact that the internal node is not a q-node.

Now we have a limited number of situations for q-nodes listed as follows:

Case (a): A q-node u has more than one p-nodes in its descendants.

In this case, children of the q-node are either p-nodes or leaves (see Figure 2-Case (a)).

Case (b): A q-node u has exactly one p-node v in its descendants.

In this case we may have arbitrary number of internal nodes on the unique path from u to v each of which has only one edge connected to a leaf (see Figure 2-Case (b)).

Case (c): A q-node u has no p-node v in its descendants.

In this case, all of the children of the q-node are leaves (see Figure 2-Case (c)). This is just like a p-node except that the sum of weights exceeds 2.

There are only two types of p-nodes since they have two or three children (leaves). The fifth reforming operation R_5 removes p-nodes having only two children. This is done by connecting those children directly by the parent of the p-node by edges of weights increased by the weight between the p-node and its parent. Formally, suppose a p-node v is connected to its parent u by an edge of weight a and to two children v_1 and v_2 by edges of weights w_1 and w_2, respectively. Then, v_i is connected directly to u by an edge of weight $w_i + a$. Note that the reforming operation does not change the lower bound since the demand of $D(T_v)$ is greater than 1 by definition of a p-node.

Yet another reforming operation R_6 is required in Case (b) above. In this case a q-node u has a single p-node v and on the way to v there are some branches to leaves. Then, those leaves are placed as the children of the p-node by edges of weights equal to those for the branches. This operation also preserves the lower bound.

After all, we can assume that if a q-node has any p-node as its child then the p-node must have three children. Again, it is obvious that the reformations described above do not increase the lower bound although they certainly increase the upper bound since we restrict possible tours.

4 Approximation Algorithm

The approximation algorithm to be presented in this paper is based on the reformations preserving the lower bound and combining two or more different strategies. The main difference from the previous approximation algorithm given by Hamaguchi and Katoh [9] is the definition of a minimal subtree to which algorithmic strategies are applied. They introduced the notion of *D-minimality* and *D-feasibility*. That is, 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 none of its children is. Their 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 one of the two strategies depending on their merits.

Our algorithm pays attention to subtrees of demands exceeding 2 instead of 1. Usually this causes explosion of the possible cases, but the point here is the reforming operations described above that extremely simplify the possible cases. This is the main contribution of this paper.

Now, let us describe our algorithm. It first applies the safe reforming operations R_1 and R_2 to an input tree. Then, the four reforming operations R_3, R_4, R_5 and R_6 are repeatedly applied until reforming operations cannot be applied any more. Then, as stated in the previous section, if there are any q-nodes then all of those q-nodes are directly connected to leaves.

We first describe the algorithm for treating those q-nodes and then consider the case where no q-node is contained in the tree, in other words, the total sum of the demands in the tree is less than 2.

So, suppose that we have a q-node u. We prepare four different procedures to serve demands in the subtree rooted at the q-node. The first procedure is applied whenever u has more than one p-node as its children. In this case we choose any two such p-nodes and serve their demands. Note that the sum of demands of two p-nodes exceeds 2. Repeating the above procedure, we come up with the two cases, depending on whether one p-node is left. If no p-node is left, the q-node u may have leaves directly connected to u. Here notice that we have already applied the merging operations and so the sum of demands of any two leaves exceeds 1.

The second procedure is applied when there are three leaves with total demands exceeding 2. When there are no such three leaves we need four leaves for their total demand to exceed 2. The third procedure is to treat this case.

The last procedure is prepared to treat the last case where one p-node is left in addition to leaves after the applications of the first procedures.

The basic idea is to prepare two or more strategies to serve the demands for each case described above to choose one of them giving the best ratio between the cost of strategy and the decreased lower bound by the application of strategy.

Due to the space limit we are concentrated on the analysis of the ratio for the procedure to treat a q-node u having four leaves. The analysis for the other procedures will be included in a full version.

Let v_1, v_2, v_3 and $v_4$be those four leaves. We denote the demand at v_i by D_i, and the weigh of the edge (u, v_i) by w_i as before. Now, by assumption, $0 \leq D_i \leq 1$, $i = 1,2,3,4$, $1 < D_1+D_2 < 2$, and $1 < D_1+D_2+D_3 < 2$, and further $1 < D_1+D_2+D_3+D_4 > 2$. Moreover, we assume that $1 < D_1+D_2+D_4 < 2$. Otherwise, we can apply the strategy for Case 1 against the three leaves v_1, v_2, and v_4. Here, without loss of generality we assume that $w_1 \geq w_2 \geq w_3 \geq w_4$.

For this case we prepare two strategies and choose one giving a better ratio. The first strategy allocates two vehicles. The first one serves the full demand at v_1 and partial demand at v_3 to fill the capacity, and the second one serves the full at v_2 and the remaining demand at v_3 and moreover partial demand at v_4. The remaining demand at v_4 is left for the next round. Then, the ratio r_1 is defined by

$$r_1 = \frac{4a + 2w_1 + 2w_2 + 4w_3 + 2w_4}{4a + 2w_1 + 2w_2 + 2w_3} < \frac{4a + 8w_3 + 2w_4}{4a + 6w_3} = \frac{2a + 4w_3 + w_4}{2a + 3w_3}. \tag{3}$$

The second one uses three vehicles, (1) to serve the full demand at v_1 and partial demand at v_4 to fill the capacity, (2) to serve the full demand at v_4 and the remaining demand at v_4, and (3) to serve the full demand at v_3. Then, the ratio r_2 is defined by

$$r_2 = \frac{6a + 2w_1 + 2w_2 + 2w_3 + 4w_4}{4a + 2w_1 + 2w_2 + 2w_3 + 2w_4} < \frac{6a + 6w_3 + 4w_4}{4a + 6w_3 + 2w_4} = \frac{3a + 3w_3 + 2w_4}{2a + 3w_3 + w_4}. \tag{4}$$

When $w_4 = 0$, the ratio r_1 is bounded by 4/3 since

$$r_1 < \frac{2a + 4w_3}{2a + 3w_3} \leq \min\{\frac{2a}{2a}, \frac{4w_3}{3w_4}\} = \frac{4}{3}. \tag{5}$$

When $w_4 > 0$, we set $x = a/w_4$ and $y = a/w_4$. Then, $x \geq 0$ and $y \geq 1$ since $w_3 \geq w_4$. Now, the ratios above are expressed as follows:

$$\begin{aligned} r_1' &= \frac{2x + 4y + 1}{2x + 3y}, \\ r_2' &= \frac{3x + 3y + 2}{2x + 3y + 1}. \end{aligned}$$

For a constant $\lambda > 1$, the inequalities $r_1'\lambda$ and $r_2' < \lambda$ both correspond to half planes. What we are looking for is the largest possible value of λ for which the union of those half planes completely covers the region defined by the intersection of $x \geq 0$ and $y \geq 1$. These half planes are depicted in Figure3, from which it is easily observed that the union covers the region if the y-coordinate y_c of the intersection of the two lines is less than or equal to 1, which is given by

$$\begin{aligned} y_c &= \left(\frac{2\lambda - 2}{4 - 3\lambda}\frac{\lambda - 2}{3 - 3\lambda} + \frac{1}{4 - 3\lambda}\frac{2\lambda - 3}{3 - 3\lambda}\right)/\left(\frac{2\lambda - 2}{4 - 3\lambda} - \frac{2\lambda - 3}{3 - 3\lambda}\right) \\ &\Longleftrightarrow 2\lambda^2 + \lambda - 5 \leq 0 \Longleftrightarrow \lambda \leq \frac{-1 + \sqrt{41}}{4} \simeq 1.35078. \end{aligned}$$

The x-coordinate x_c of the intersection is given by λ

That is, the first strategy is better for $x \geq \lambda \simeq 1.35078$ while the second one is better otherwise. In any case, the minimum of $r_1$and r_2 is bounded by $\lambda \simeq 1.35078$.

We can summarize the results as the next theorem.

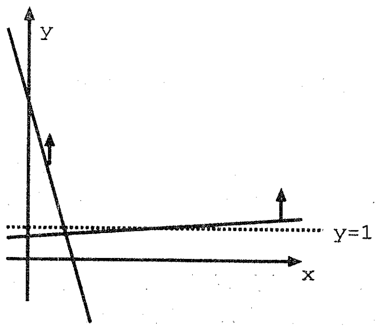


Figure 3: The region covered by the half planes associated with the two inequalities.

Theorem 1 *The approximation of our algorithm for TREE-CVRP is* 1.35.

The theorem can be proved by induction on the number of rounds. Whenever the sum of the demands in the tree exceeds two, we perform the reforming operations to have q-nodes and design how to serve the demands in the subtree rooted at each q-node. Then, we apply the reforming operations again to the resulting tree and repeat this process until there is no q-node. This is the base case.

Let P denote the problem instance of TREE-CVRP for which our algorithm requires $k+1$ rounds. Each time we find a q-node and apply an appropriate strategy based on the ratios defined above. 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 $LB(P')$ be the lower bound for the problem P' and LB_1 be the decreased lower bound at this round. 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., $cost(P) = cost_1 + cost(P')$). Then, we have

$$\frac{cost(P)}{LB(P)} \leq \frac{cost_1 + cost(P')}{LB_1 + LB(P')}. \tag{6}$$

Since $cost(P')/LB(P') \leq 1.35$ holds from the induction hypothesis, it suffices to prove

$$\frac{cost_1}{LB_1} \leq 1.35. \tag{7}$$

As we already saw, the above inequalities hold in every case. Thus, we have the theorem.

5 Conclusions

We have presented a new approximation algorithm for finding an optimal tours to serve demands located at nodes of a tree-shaped network. Our new algorithm establishes the approximation ration 1.35 (exactly, $(\sqrt{41}-1)/4$). This ratio seems to be almost best possible since there is an instance of TREE-CVRP for which the cost of an optimal solution is asymptotically 4/3 times larger than the lower bound of the cost. To have better ratio we have to improve the lower bound.

References

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

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.

[9] S. Hamaguchi and N. Katoh. A Capacitate Vehicle Routing Problem on a Tree, Proc. of ISAAC'98, *Lecture Notes in Computer Science 1533, Springer-Verlag* 397-406, 1998.

[10] 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.

[11] 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.

[12] M. Labbé, G. Laporte and H. Mercure. Capacitated Vehicle Routing Problems on Trees, *Operations Research*, Vol. 39 No. 4 (1991) 616-622.

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

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

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