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

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

[*]School of Information Science, JAIST,
Asahidai, Tatsunokuchi, 923-1292, Japan
{t-asano, kawasima}@jaist.ac.jp.

[**]Department of Architecture and Architectural Systems, Kyoto University,
Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501 Japan
naoki@archi.kyoto-u.ac.jp

Corresponding address:

Naoki Katoh
Department of Architecture and Architectural Systems
Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501 Japan
FAX: +81-75-753-4906
email: naoki@archi.kyoto-u.ac.jp

[1] Currently he is with Fujisoft ABC, Inc., Kamakura, Japan.

**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 1.35078-approximation algorithm for the problem (exactly, $(\sqrt{41}-1)/4$), 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 [5, 8, 9, 12, 13] 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, 10, 11, 16, 17, 18]. 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. In 1998, Hamaguchi and Katoh [15] proved its NP-hardness and proposed a 1.5-approximation algorithm ([18] considered the variant of TREE-CVRP where demand of each customer is not splittable and gave 2-approximation algorithm.) 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 was not clear. It is shown that TREE-CVRP is strongly NP-complete by a reduction from *bin-packing problem* [15].

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 $\alpha$-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 1.35078-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 [15]. 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

We assume that tree $T = (V, E)$ is weighted, i.e., a nonnegative weight $w(e)$ is associated with each edge $e \in E$, which represents the length of $e$. 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 $v$. 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). Let

$$LB(T) = \sum_{e \in E} LB(e). \tag{3}$$

Thus, we have the following lemma.

**Lemma 1** *$LB(T) = \sum_{e \in E} LB(e)$ 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. The first step is a reforming step in which we reshape a given tree following seven different operations all of which are "safe" in the sense that they do not increase the lower bound on the cost given in Lemma 1. The second step is to 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 (this step will be explained in details in the next section).

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; 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 operation $R_3$ 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. If the combined demand $D(u) + D(v)$ does not exceed 1, then we just 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 $v$. On the other hand, if $D(u) + D(v)$ exceeds 1, we send one vehicle to serve the full demand at $v$ and partial demand at $u$ to fullfil the capacity of the vehicle, and then reduce the demand at $u$ accordingly. The edge to $v$ is then removed together with $v$. Note that the combined demand $D(u) + D(v)$ never exceeds 2 after the reforming operation $R_1$.

The fourth operation $R_4$ is to merge a subtree whose demand is less than or equal to 1 into a single edge. Namely, for an internal node $v$ with $D(T_v) \leq 1$, $T_v$ is replaced by a single edge $(v, v')$ with edge weight equal to $w(T_v)$ and $D(v') = D(T_v)$. Since $D(T_v) \leq 1$ holds, this operation is also safe.

To define more essential reform operations 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 is between 1 and 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 fifth reforming operation $R_5$ is to merge leaves of nodes. For a node $u$, let $\{v_1, v_2, \ldots, v_k\}$ be a subset of its children that are leaves. 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 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 proceeds. This operation is useful particularly for p-nodes and q-nodes to reduce the number of possible cases to be considered.

An important property of the resulting tree after performing $R_5$ to p-nodes is that any p-node has at most three children (leaves) 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.

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

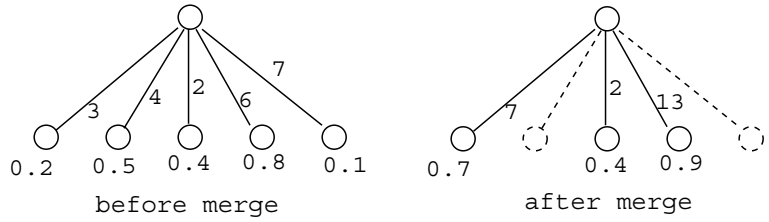


Figure 1: The merging operation.

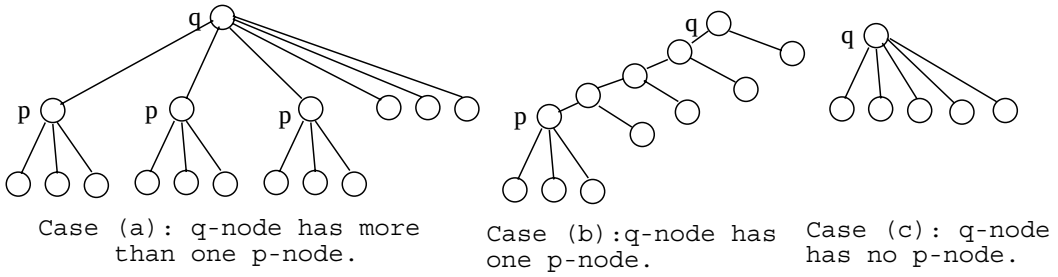


Figure 2: The three cases for a q-node.

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-node 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 sixth reforming operation $R_6$ 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 $R_7$ is required in Cases (a) and (b) above. In these cases, for a p-node $v$, there may have some branches to leaves on the way from $u$ to $v$. Then, those leaves are placed as the children of the p-node with edges of weights equal to those for the branches. In addtion, internal nodes on the path from $u$ to $v$ are erased so that the path is replaced by a single edge with the weight equal to that for the path. This operation also

preserves the lower bound. Notice that when $u$ has children which are leaves directly connected to $u$, such a leaf $x$ can be a candidate for this operation if $D(T_v) + D(x) < 2$ holds.

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 [15] 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 seven reforming operations $R_1$ through $R_7$ to an input tree so that any of these operations cannot be applied any more. The algorithm consists of a number of rounds. In each round, it focuses on a particular $q$-node $u$ and prepare a few strategies each of which allocates two or three vehicles to (partially) serve the demands in the subtree $T_u$. Among strategies prepared, we choose the best one and apply it to $T_u$. As in the same manner as the one used by [15], the choice of the best strategy is made as follows. For each strategy, we compute the cost of tours required by the strategy. We also compute the decrease of the lower bound. Namely, let $P$ denote the problem before applying the strategy and let $LB(P)$ denote the lower bound of optimal cost of $P$ given in Lemma 1. After applying the strategy, demands of nodes in the subtree $T_u$ are decreased and we obtain another problem instance $P'$ to which the algorithm will be further applied. The decrease of the lower bound is defined as $LB(P) - LB(P')$. The best strategy is the one giving the smallest ratio of the cost of tours to the decrease of the lower bound. As we shall show later, the smallest ratio is always at most 1.35078.

When there is not any $q$-node any more, it is the final round and is called *base case*. In this case, similarly to the other cases, we prepare a few strategies and apply the one with better ratio. It will be shown that this ratio is also at most 1.35078.

**Theorem 1** *The approximation of our algorithm for TREE-CVRP is* 1.35078.

Proof. The proof technique is similar to the one by Hamaguchi and Katoh [15]. In fact, 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, depending on the three cases explained. 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.

Assuming that the theorem holds for problem instances that require at most $k$ rounds, we consider the problem instance $P$ 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

$$\frac{cost(P)}{LB(P)} \leq \frac{cost_1 + cost(P')}{LB_1 + LB(P')}. \tag{4}$$

Since $cost(P')/LB(P') \leq 1.35078$ holds from the induction hypothesis, it suffices to prove

$$\frac{cost_1}{LB_1} \leq 1.35078.$$

As we shall prove below (Lemmas 2 through 5), the above inequality holds in every case (the base case will be proved in Lemma 6). Thus, we have the theorem. □

Suppose there is at least one $q$-node. Strategies we prepare depend on the cases explained below. As stated in the previous section, the following three cases of $q$-nodes are possible.

**Case 1:** A q-node has more than one p-node as its children.

**Case 2:** A q-node has only one child of p-node.

**Case 3:** A q-node has no child of p-node.

In Case 1, we focus on arbitrary two p-nodes. Remaining p-nodes, if any, will be considered in later rounds of the algorithm. In Case 2, let $u$ and $v$ denote the q-node and the p-node respectively. ¿From the definition of a q-node, $u$ has at least one child other than the p-node. We choose arbitrary one child $v'$ other than the p-node. The algorithm then focuses on the subgraph consisting of edges $(u, v)$, $(u, v')$ and the subtree $T_v$. We notice here that $D(T_v)+D(v')$ exceeds 2 since otherwise $v'$ can be shifted down to become a child of $v$ by reform operation $R_7$.

In Case 3, we can assume that the q-node has at least three leaves from definition of the q-node. If it has exactly three leaves, it follows that the sum of demands of these leaves exceeds 2 due to the definition of a q-node. If it has at least four leaves, we arrange those leaves in any order and find where the sum of the demands exceeds 2. Recall that the merging operation $R_5$ is already applied, and thus the sum of demands of the first four leaves certainly exceeds 2. Since the sum of the demands of the first two leaves is less than 2 (each demand is less than 1), we can conclude that there are only two possibilities, that is, either that of the first three leaves exceeds 2 (Subcase 3A) or that of the first four does (Subcase 3B). In either case, the algorithm focuses on such first three or four leaves depending on the cases. (The remaining leaves will be treated in later rounds.)

Let us describe the algorithm for treating q-nodes depending on the above cases and then consider the base case, i.e., the total sum of the demands in the tree is less than 2. For each case, we shall explain how to schedule vehicles to serve demands of nodes that the algorithm focuses on, and prove that the ratio of the cost to the lower bound is at most 1.35078. In the proofs for all cases, we shall implicitly use the following simple facts:

**Fact 1:**

$$0 < x < y \text{ and } a > 0 \Longrightarrow \frac{y+a}{x+a} < \frac{y}{x}.$$

**Fact 2:** For $p, q, r, s > 0$,

$$0 \leq a < b \text{ and } \frac{r}{p} > \frac{s}{q} \Longrightarrow \frac{ra+s}{pa+q} < \frac{rb+s}{pb+q},$$

and

$$0 \le b < a \text{ and } \frac{r}{p} < \frac{s}{q} \Longrightarrow \frac{ra+s}{pa+q} < \frac{rb+s}{pb+q}.$$

**Case 1:** Let $u$ denote the q-node and let $x$ and $x'$ denote the two p-nodes the algorithm focuses on. Let $v_1, v_2$ and $v_3$ denote three leaves of the subtree $T_x$. We denote by $D_1, D_2$ and $D_3$ their demands, and by $w_1, w_2$, and $w_3$ weights of edges connecting leaves to their parent. The weight of the unique path from the tree root $r$ to $u$ is denoted by $a$ (if $u$ coincides with the tree node $a = 0$), and that from $u$ to $x$ by $b$. Now, by assumption, $0 < D_i < 1, i = 1, 2, 3,\ 1 < D_1 + D_2 < 2$, and $1.5 < D_1 + D_2 + D_3 < 2$ ($D_1 + D_2 + D_3 > 1.5$ follows since otherwise the sum of some two demands among $D_1$, $D_2$ and $D_3$ is less than or equal to 1, and these two demands are mergeable, a contradiction. Here, without loss of generality we assume $w_1 \ge w_2 \ge w_3$. Similarly, let $v'_1, v'_2$ and $v'_3$ denote three leaves of the subtree $T_{x'}$. We denote by $D'_1, D'_2$ and $D'_3$ their demands, and by $w'_1, w'_2$, and $w'_3$ weights of edges connecting leaves to their parent. The weight of the path from $u$ to $x'$ is denoted by $b'$. Similarly, by assumption, $0 < D'_i < 1, i = 1, 2, 3,\ 1 < D'_1 + D'_2 < 2$, and $1.5 < D'_1 + D'_2 + D'_3 < 2$ hold, and $w'_1 \ge w'_2 \ge w'_3$ is assumed.

Here we prepare only one strategy. The strategy is to allocate two vehicles for each of subtrees $T_x$ and $T_{x'}$ to serve the demands of each of these subtrees; For $T_x$, the first vehicle serves the demand at $v_1$ and the partial demand at $v_3$ so that the sum of demands is equal to 1, and the second vehicle serves the demand at $v_2$ and the remaining demand at $v_3$. Similarly for $T_{x'}$, the first vehicle serves the demand at $v'_1$ and the partial demand at $v'_3$ so that the sum of demands is equal to 1, and the second vehicle serves the demand at $v'_2$ and the remaining demand at $v'_3$. Let $v'_1, v'_2$ and $v'_3$ denote three leaves of the subtree $T_{x'}$. We denote by $D'_1, D'_2$ and $D'_3$ their demands, and by $w'_1, w'_2$, and $w'_3$ weights of edges connecting leaves to their parent. The weight of the path from $u$ to $x$ is denoted by $b'$.

The cost required by these four vehicles is given by

$$8a + 4b + 4b' + 2w_1 + 2w_2 + 4w_3 + 2w'_1 + 2w'_2 + 4w'_3.$$

The decrease of the lower bound is given by

$$6a + 4b + 4b' + 2w_1 + 2w_2 + 2w_3 + 2w'_1 + 2w'_2 + 2w'_3.$$

The first term $6a$ comes from the fact that for every edge $e$ on the path from the root $r$ to $u$, the decrease of the lower bound $LB(e)$ is at least $6w(e)$ and at most $8w(e)$ from (2) because the decrease of $D(T_u)$ is between 3 and 4. Thus, the ratio of the cost of the tours to the decreased lower bound is given by

$$r_1 \quad = \quad \frac{8a + 4b + 4b' + 2w_1 + 2w_2 + 4w_3 + 2w'_1 + 2w'_2 + 4w'_3}{6a + 4b + 4b' + 2w_1 + 2w_2 + 2w_3 + 2w'_1 + 2w'_2 + 2w'_3}.$$

¿From $w_1 \ge w_2 \ge w_3$ and $w'_1 \ge w'_2 \ge w'_3$, we can show that $r_1$ is bounded by $4/3$ as follows:

$$r_1 \le \frac{8a + 2w_1 + 2w_2 + 4w_3 + 2w'_1 + 2w'_2 + 4w'_3}{6a + 2w_1 + 2w_2 + 2w_3 + 2w'_1 + 2w'_2 + 2w'_3} \le \frac{8a + 8w_3 + 8w'_3}{6a + 6w_3 + 6w'_3} = \frac{4}{3}.$$

The first inequality holds from Fact 1, and the second from Fact 2.

**Lemma 2** *In Case 1, the approximation ratio is at most* $4/3$.

Before explaining how to treat Case 2, we shall describe the algorithm for Case 3 because Case 2 is most complicated.

**Subcase 3A** A q-node $u$ has three leaves whose total demands exceed 2. Let $v_1, v_2$, and $v_3$ be those three leaves. We denote by $D_1, D_2$ and $D_3$ their demands, and by $w_1, w_2$, and $w_3$ the

edge weights. The weight of the unique path from the tree root $r$ to $u$ is denoted by $a$. Now, by assumption, $0 < D_i < 1, i = 1, 2, 3,\ 1 < D_1 + D_2 < 2$, and $D_1 + D_2 + D_3 > 2$. Here, without loss of generality we assume $w_1 \geq w_2 \geq w_3$.

We consider two strategies and choose the better one. The first strategy is to allocate three vehicles to serve all the demands $D_1$, $D_2$ and $D_3$. In this case, the cost is given by $6a + 2(w_1 + w_2 + w_3)$ because the $i$-th vehicle starts at the root and serves the demand $D_i$ and returns to the root. After applying the strategy, the three leaves are removed. The decrease of the lower bound is given by $4a + 2(w_1 + w_2 + w_3)$.

Thus the ratio between the cost of the tours and the decreased lower bound is given by

$$r_1 \quad = \quad \frac{6a + 2(w_1 + w_2 + w_3)}{4a + 2(w_1 + w_2 + w_3)}.$$

The second strategy splits the demand $D_3$. We allocate two vehicles to serve the demand at $v_1$ and serve the demand at $v_3$ partially so that the sum of the demand is equal to 1, the capacity of the vehicle. The second vehicle visits $v_2$ and $v_3$ in a similar fashion. Since the sum of the three demands exceeds 2, some portion of the demand $D_3$ is still remaining although the demands at $v_1$ and $v_2$ are exhausted. The remaining demand at $v_3$ is left for the next round. Now, the associated cost and lower bound are given by $4a + 2w_1 + 2w_2 + 4w_3$, and $4a + 2w_1 + 2w_2$, respectively. Thus, the ratio is

$$r_2 \quad = \quad \frac{4a + 2w_1 + 2w_2 + 4w_3}{4a + 2w_1 + 2w_2}.$$

¿From $w_1 \geq w_2 \geq w_3$, we then have

$$\begin{aligned} r_1 \quad &\leq \quad \frac{6a + 6w_3}{4a + 6w_3}, \quad \text{and} \\ r_2 \quad &\leq \quad \frac{4a + 8w_3}{4a + 4w_3}. \end{aligned}$$

Therefore, if $a \leq 2w_3$, we have

$$r_1 \quad \leq \quad \frac{12w_3 + 6w_3}{8w_3 + 6w_3} = \frac{9}{7},$$

and otherwise we have

$$r_2 \quad < \quad \frac{8w_3 + 8w_3}{8w_3 + 4w_3} = \frac{4}{3}.$$

**Lemma 3** *In Subcase 3A, the approximation ratio is at most* 1.35078.

**Subcase 3B:** A q-node $u$ has four leaves. This case can be viewed as a special case of Case 2 in which the length of the path from $u$ to $v$ is equal to 0. Thus, it will be treated in Case 2.

**Case 2:** Suppose that a q-node $q$ has only one p-node $p$ having three leaves $v_1, v_2$ and $v_3$ together with a single leaf $v_4$. As before, we denote the demand and edge weight associated with $v_i$ by $D_i$ and $w_i$, respectively, and $w_1 \geq w_2 \geq w_3$ is assumed. We denote the path length from the root to the q-node by $a$, and the weight of the edge between the p-node and q-node by $b$. The algorithm prepares different strategies depending on whether

$$D_1 + D_2 + D_4 \leq 2$$

holds or not. The case where this inequalty holds is called Subcase 2A, and the other is called Subcase 2B.

**Subcase 2A:** We prepare the following four strategies.
**Strategy 1:** This strategy allocates three vehicles. The first vehicle serves the full demand at $v_1$ and partial demand at $v_3$ to the full capacity. The second one serves the full demand at $v_2$ and the remaining demand at $v_3$ (which is less than 1). The third one serves the demand of $v_4$. Then, the ratio is given by

$$r_1 = \frac{6a+4b+2w_1+2w_2+4w_3+2w_4}{4a+4b+2w_1+2w_2+2w_3+2w_4}.$$

**Strategy 2:** This strategy is the same as Strategy 1 except that the roles of $v_3$ and $v_4$ are exchanged. The ratio is given by

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

**Strategy 3:** This strategy allocates two vehicles, (1) to serve the full demand at $v_1$ and partial demand at $v_3$ to fill the capacity, and (2) to serve the full demand at $v_2$ and the remaining demand at $v_3$ and moreover partial demand at $v_4$. This is possible since $D_1 + D_2 + D_3 < 2$. The remaining demand at $v_4$ is left for the next round. Then, the ratio $r_3$ is defined by

$$r_3 = \frac{4a+4b+2W+2w_3}{4a+4b+2W-2w_4},$$

where $W = w_1 + w_2 + w_3 + w_4$.
**Strategy 4:** This strategy allocates two vehicles, (1) to serve the full demand at $v_1$ and partial demand at $v_2$ to fill the capacity, and (2) to serve the full demand at $v_4$ and the remaining demand at $v_2$ and moreover partial demand at $v_3$. This is possible since $D_1 + D_2 + D_4 \leq 2$. The remaining demand at $v_3$ is left for the next round. Then, the ratio $r_4$ is defined by

$$r_4 = \frac{4a+4b+2W+2w_2}{4a+2b+2W-2w_3}.$$

The smallest value among the above four values is evaluated by the following case analysis.

(i) $w_3 \geq w_4$. We choose the better one between Strategies 2 and 3. From $w_3 \geq w_4$, we have

$$\begin{aligned} r_2 &\leq \frac{6a+4b+10w_4}{4a+4b+8w_4} \leq \frac{6a+10w_4}{4a+8w_4}, \\ r_3 &\leq \frac{4a+10w_4}{4a+6w_4}. \end{aligned}$$

By letting $x = a/w_4$, we compute

$$\max_x \min\{\frac{6x+10}{4x+8}, \frac{4x+10}{4x+6}\}. \tag{5}$$

The maximum is attained when $\frac{6x+10}{4x+8} = \frac{4x+10}{4x+6}$ holds, i.e., $x = \frac{-1+\sqrt{41}}{4} \simeq 1.35078$. The maximum value of (5) is also $\frac{-1+\sqrt{41}}{4} \simeq 1.35078$.

(ii) $w_2 \geq w_4 \geq w_3$.

We choose the better one between Strategies 1 and 3. The analysis is done in the same way as (i).

(iii) $w_4 \geq w_2$. We choose the better one between Strategies 1 and 4.

$$\begin{aligned} r_1 &= \frac{6a+4b+2w_1+2w_2+4w_3+2w_4}{4a+4b+2w_1+2w_2+2w_3+2w_4} \leq \frac{6a+4b+2w_1+6w_2+2w_4}{4a+4b+2w_1+4w_2+2w_4} (\text{ from } w_2 \geq w_3) \\ &\leq \frac{6a+4b+10w_2}{4a+4b+8w_2}. \text{ (from } w_1, w_4 \geq w_2). \end{aligned} \tag{6}$$

$$r_4 = \frac{4a+4b+2w_1+4w_2+2w_3+2w_4}{4a+2b+2w_1+2w_2+2w_4} \leq \frac{4a+4b+2w_1+6w_2+2w_4}{4a+2b+2w_1+2w_2+2w_4} \quad (\text{from } w_2 \geq w_3)$$
$$\leq \frac{4a+4b+10w_2}{4a+2b+6w_2}. \quad (\text{from } w_1, w_4 \geq w_2).$$

When $w_2 = 0$, we have

$$\min\{r_1, r_4\} \leq \min\{\frac{6a+4b}{4a+4b}, \frac{4a+4b}{4a+2b}\}.$$

By easy calculation, we get $\min\{r_1, r_4\} \leq 1.3$. Now consider the case of $w_2 > 0$. By letting $x = a/w_2, y = b/w_2$, we have

$$\min\{r_1, r_4\} \leq \min\{\frac{3x+2y+5}{2x+2y+4}, \frac{2x+2y+5}{2x+y+3}\}.$$

Let

$$f(x,y) = \min\{\frac{3x+2y+5}{2x+2y+4}, \frac{2x+2y+5}{2x+y+3}\}. \tag{7}$$

We compute $\max_{x,y\geq 0} f(x,y)$ using the known theorem for generalized fractional program [6, 7] (see Appendix A). ¿From Theorem 3 in Appendix A, in order to maximize (7), we consider the following parametric problem:

$$z(\lambda) = \max_{x,y} \min\{3x+2y+5-\lambda(2x+2y+4), 2x+2y+5-\lambda(2x+y+3)\}. \tag{8}$$

Let

$$f_1(x,y) = 3x+2y+5-\lambda(2x+2y+4), \quad f_2(x,y) = 2x+2y+5-\lambda(2x+y+3).$$

Since $f_1(x,y) - f_2(x,y) = x - \lambda y - \lambda$, the minimum of $f_1(x,y)$ and $f_2(x,y)$ is attained when $f_1(x,y) = f_2(x,y)$, i.e.,

$$x = \lambda y + \lambda.$$

Then, substituting $x = \lambda(y+1)$ into $f_1(x,y)$ we have

$$f_1(x,y) = (-2\lambda^2 + \lambda + 2)y + 5 - \lambda - 2\lambda^2. \tag{9}$$

When $-2\lambda^2+\lambda+2 > 0$, i.e., $\lambda < (1+\sqrt{17})/4 \simeq 1.28$, $5-\lambda-2\lambda^2 > 0$ always holds for any $y \geq 0$. Thus, $f_1(x,y) = 0$ does not occur when $-2\lambda^2+\lambda+2 > 0$. When $-2\lambda^2+\lambda+2 < 0$, maximum of (9) is attained at $y = 0$, and the value of $f_1(x,y)$ at $y = 0$ is $5-\lambda-2\lambda^2$. Thus, when $5-\lambda-2\lambda^2 = 0$, i.e., $\lambda = 1.35078$, $z(\lambda) = \max_{x,y} \min\{f_1(x,y), f_2(x,y)\} = 0$ holds. Therefore, the approximation ratio is 1.35078 in this case. Summarizing the analysis made in (i), (ii) and (iii), we have the following lemma.

**Lemma 4** *In Subcase 2A, the approximation ratio is at most* 1.35078.

**Lemma 5** *In Subcase 2B, the approximation ratio is at most* 1.35078.

The proof for the lemma proceeds in a similar manne although slightly more sophisticated analysis is required. For readability the proof was put in Appendix B.

Next, we shall turn our attention to the base case where there is no q-node in the input tree, that is, the total sum of the demands is less than 2. Applying the reforming operations to the tree results in a simple tree of the three forms depending on the number of leaves, shown in Figure 3.

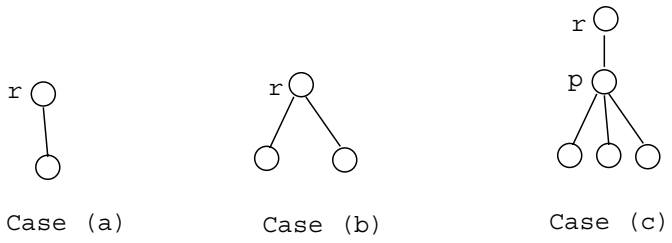


Figure 3: The base case.

In case (a), the remaining demand is less than one, and thus is served optimally by a single vehicle. In case (b), we allocate two vehicles, one for each leaf. It is also easy to see that this strategy is also optimal. Now let us consider case (c) in which a p-node $u$ has three leaves $v_1$, $v_2$ and $v_3$. Symbols $D_i$, $w_i$, and $a$ are used as before. We also assume that $w_1 \geq w_2 \geq w_3$.

We allocate two vehicles; one to serve the demands $D_1$ and a fraction of $D_3$ so as to fill the full capacity of the vehicle, and another to serve $D_2$ and the remaining demand of $D_3$. Then, the ratio is given by

$$r_1 = \frac{4a + 2w_1 + 2w_2 + 4w_3}{4a + 2w_1 + 2w_2 + 2w_3} \leq \frac{4a + 8w_3}{4a + 6w_3} \leq 4/3 < 1.35078.$$

**Lemma 6** *In the base case, the approximation ratio is at most* 1.35078.

# 5 Lower Bound

So far we have been concerned with the approximation algorithm for TREE-CVRP which has some improved performance ratio, 1.35078 (exactly $(\sqrt{41} - 1)/4$). The following theorem suggests that the ration seems to be nearly best possible to achieve under the lower bound model in this paper.

**Theorem 2** *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.*

*Proof:* Let $T$ be a tree consisting of the root $r$, one internal node $q$ connected with $r$ by an edge of cost $a$, and $2n + 1$ leaves all connected with the node $q$ by edges of costs 1. The demand at each leaf is $0.5 + \epsilon$. The value of $\epsilon$ is small enough so that $(2n + 1)\epsilon + 0.5 < 1$, i.e., $\epsilon < 1/(4n + 2)$).

By the definition, the total demand $D_q$ at $q$ is calculated as follows:

$$D_q = (2n + 1) \times (0.5 + \epsilon) = n + (2n + 1)\epsilon + 0.5 < n + 1.$$

Thus, the lower bound $LB$ is given by

$$LB = 2(2n + 1) + 2a(n + 1) = 4n + 2 + (2n + 2)a.$$

What is an optimal solution? How many vehicles should we use? Since the total demand is greater than $n$, we need at least $n + 1$ vehicles. $2n + 1$ vehicles are also sufficient since we can serve all of the $2n + 1$ leaves by them. Thus, the number of vehicles to be used is between $n + 1$ and $2n + 1$. For each $k \in [n + 1, 2n + 1]$, we denote by $OPT(k)$ the cost of an optimal solution when $k$ vehicles are used.

It is easy to calculate $OPT(2n + 1)$, the cost for $2n + 1$ vehicles. Just use one vehicle for each leaf. Hence, we have

$$OPT(2n + 1) = (2n + 1) \times (2a + 2) = 4n + 2 + (4n + 2)a.$$

Next, consider $OPT(2n+1-k)$, where $k \in [1,n]$. In this case, it is obvious that at least $k$ leaves must be split, i.e., must be visited at least twice. Thus, if we can design a feasible vehicle schedule so that exactly $k$ leaves are visited twice, then it is optimal. In fact, it is possible. Let $v_1, v_2, \ldots, v_{2n+1}$ be the leaves. The first $k$ leaves are split, while the remaining ones remain unsplit. Let $V_1, V_2, \ldots, V_{2n+1-k}$ be the vehicles to be used. The demand $0.5+\epsilon$ at the first leaf $v_1$ is split into $0.5-\epsilon$ and $2\epsilon$, which are assigned to $V_{k+1}$ and $V_{k+2}$, respectively. The demand at $v_2$ is split into $0.5-3\epsilon$ and $4\epsilon$, which are assigned to $V_{k+2}$ and $V_{k+3}$, respectively, that at $v_3$ into $0.5-5\epsilon$ and $6\epsilon$, and so on. The demand at $v_k$ is split into $0.5-(2k-1)\epsilon$ and $2k\epsilon$, which are assigned to $V_{2k}$ and $V_{2k+1}$. Here note that $2k+1 \leq 2n+1$ since $k \leq n$. Thus, these vehicles assignments are always possible. Then, each of the vehicles $V_{k+1}, \ldots, V_{2k}$ serves exactly one demand, and $V_{2k+1}$ serves $0.5+\epsilon+2k\epsilon = 0.5+(2k+1)\epsilon < 1$. The remaining vehicles if any visits only one leaf and thus serves $0.5+\epsilon$. Thus, the cost of the solution is given by

$$\begin{aligned} OPT(2n+1-k) &= 2(2n+1-k)+4k+2a(2n+1-k) \\ &= 4n+2+2k+(4n+2-2k)a. \end{aligned}$$

The worst case occurs when we have
$OPT(2n+1) = OPT(2n) = \cdots = OPT(n+1)$,
which leads to

$$4n+2+(4n+2)a = 4n+2+2k+(4n+2-2k)a,$$

and thus

$$2k-2ka=0, \quad \text{that is,} \quad a=1.$$

This implies that the cost $a$ of the edge $(r,q)$ must be 1 to achieve this extreme case. Substituting $a=1$, we have

$$\frac{OPT(2n+1)}{LB} = \cdots = \frac{OPT(n+1)}{LB} = \frac{4n+2+(4n+2)\times 1}{4n+2+(2n+2)\times 1} = \frac{8n+4}{6n+4}.$$

The ratio above approaches $4/3$ as $n$ goes infinity.

# 6 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 1.35078 (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, which is left for future research.

# Acknowledgments

This work was partially supported by Grant in Aid for Scientific Research of the Ministry of Education, Science and Cultures of Japan.

# 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] J.P. Crouzeix, J.A. Ferland and S. Schaible, An algorithm for generalized fractional programs, *J. of Optimization Theory and Applications*, 47 (1985), 35-49.

[7] J.P. Crouzeix, J.A. Ferland and S. Schaible, A note on an algorithm for generalized fractional programs, *J. of Optimization Theory and Applications*, 50 (1986), 183-187.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[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 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.

[16] 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.

[17] 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.

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

## Appendix A: Generalized Fractional Program

Let us consider the following problem (called generalized fractional program):

$$P: \ \text{maximize}_{(x,y)\in S} f(x,y) = \min_{1\le j\le m} \frac{g_j(x,y)}{h_j(x,y)},$$

where $x, y$ are real variables, $S$ denotes a feasible domain of $(x, y)$, $g_j(x, y) > 0$ for any $(x, y) \in S$, and $m$ is a positive integer. Let us define the following parametric problem associated with $P$.

$$P(\lambda): \ z(\lambda) \equiv \max_{(x,y)\in S} f_\lambda(x,y) = \min_{1\le j\le m} g_j(x,y) - \lambda h_j(x,y)$$

**Theorem 3** *Let $(x^\lambda, y^\lambda)$ be the maximizer of $P(\lambda)$. If $\lambda^*$ satisfies*

$$z(\lambda^*) = 0,$$

*then $(x^{\lambda^*}, y^{\lambda^*})$ is an optimal solution of $P$.*

## Appendix B: Proof of Lemma 5

**Subcase 2B:** $D_1 + D_2 + D_4 > 2$ holds.
(i) $w_2 \ge w_4$. This case can be treated exactly in the same manner as the cases (i) and (ii) of the proof of Subcase 2A.

(ii) $w_2 < w_4$. We prepare three strategies. The first strategy is the same as Strategy 1 of Subcase 2A. The second strategy allocates three vehicles which fill the demand of $v_1$, $v_2$ and $v_4$, respectively. The whole demand of $v_3$ will be served in the succeeding rounds. Thus, the ratio is given by

$$r_2 = \frac{6a + 4b + 2w_1 + 2w_2 + 2w_4}{4a + 2b + 2w_1 + 2w_2 + 2w_4}.$$

The third strategy allocates two vehicles. The first vehicle is to serve the full demand at $v_1$ and partial demand at $v_2$ to fill the capacity, and the second vehicle is to serve the remaining demand at $v_4$ and the partial demand at $v_2$ to fill the capacity. Since the demand at $v_2$ is still remaining from $D_1 + D_2 + D_4 > 2$ even after this strategy is applied, we have

$$r_3 = \frac{4a + 4b + 2w + 2w' + 4w_2}{4a + 2b + 2w + 2w'} \quad (\text{from } w_1, w_4 > w_2)$$

Thus,

$$r_3 \le \frac{4a + 4b + 8w_2}{4a + 2b + 4w_2}.$$

For the ratio $r_1$ of the first strategy, we have

$$r_1 \le \frac{6a + 4b + 10w_2}{4a + 4b + 8w_2}$$

from (6). For the ratio $r_2$ of the second strategy, we have

$$r_2 \le \frac{6a + 4b + 6w_2}{4a + 2b + 6w_2}.$$

We will compute

$$\min_{a,b,w_2} \{r_1, r_2, r_3\}.$$

If $w_2 = 0$, $\min\{r_1, r_2, r_3\} \leq 1.35078$ is clear (minimum of $r_1$ and $r_2$ is at most 1.26). Thus, let us assume $w_2 > 0$. Letting $x = a/w_2$ and $y = b/w_2$, we have

$$r_1 = \frac{3x + 2y + 5}{2x + 2y + 4},\ r_2 = \frac{3x + 2y + 3}{2x + y + 3},\ r_3 = \frac{2x + 2y + 4}{2x + y + 2}.$$

For a positive parameter $\lambda$, let us define

$$f_1 = 3x+2y+5-\lambda(2x+2y+4),\ f_2 = 3x+2y+3-\lambda(2x+y+3),\ f_3 = 2x+2y+4-\lambda(2x+y+2).$$

¿From Theorem 3, we consider the problem of finding $\lambda$ such that

$$z(\lambda) = \max_{x,y \geq 0} \min\{f_1, f_2, f_3\}$$

becomes 0. This happens only when $f_1 = f_2 = f_3 = 0$ holds. $f_1 = f_2$ implies

$$\lambda(y + 1) = 2,$$

and $f_1 = f_3$ implies

$$x = \lambda(y + 2) - 1.$$

It follows from these two equations that

$$x = \lambda + 1, y = 2/\lambda - 1.$$

Substituting this into $f_1 = 0$, we have

$$2\lambda^3 + \lambda^2 - 2\lambda - 4 = 0.$$

The positive solution of this equation is $\lambda = 1.3462718\ldots$. In fact, for $\lambda = 1.3462718$, $\max_{x,y \geq 0}$ $\min\{f_1, f_2, f_3\} \simeq 0$ holds.