An Approximation Algorithm for the Pickup and Delivery Vehicle Routing Problem on Trees

Naoki Katoh[1] and Taihei Yano[2]

Abstract

This paper presents an approximation algorithm for a vehicle routing problem on a tree-shaped network with a single depot where there are two types of demands, pickup demand and delivery demand. Customers are located on nodes of the tree, and each customer has a positive demand of pickup and/or delivery.

Demands of customers are served by a fleet of identical vehicles with unit capacity. Each vehicle can serve pickup and delivery demands. 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. In each tour, a vehicle begins at the depot with certain amount of goods for delivery, visits a subset of the customers in order to deliver and pick up goods and returns to the depot. At any time during the tour, a vehicle must always satisfy the capacity constraint, i.e., at any time the sum of goods to be delivered and that of goods that have been picked up is not allowed to exceed the vehicle capacity. We propose a 2-approximation algorithm for the problem.

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 nodes and E is a set of edges, and $r \in V$ be a designated node called *depot*. Nonnegative weight $w(e)$ is associated with each edge $e \in E$, which represents the length of e. Customers are located at nodes of the tree. There are two types of demand, i.e., pickup and delivery demands. Nonnegative values $D^p(v)$ and $D^d(v)$ are associated with each customer at $v \in V$ which represent pickup and delivery demands, respectively. Pickup demand (p-demand for short) is a request to bring goods located at the customer to the depot and delivery demand (d-demand) is the one to bring goods located at the depot to the customer.

Thus, when there is no customer at v, $D^p(v) = 0$ and $D^d(v) = 0$ are 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 p- and d-demands of a customer are splittable, i.e., the p-demand (or d-demand) 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. It is assumed that at any time during the tour, a vehicle is not allowed to violate the capacity constraint, i.e., $Z(t) \leq 1$ holds at any time t, where $Z(t)$ is defined as $Z(t) =$ (the total amount of goods to be delivered in a tour) $-$ (the amount of goods that have been delivered before time t in the tour)+(the amount of goods that have been picked up before time t in the tour). exceed 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-CVRPPD.

Vehicle routing problems have long been studied by many researchers (see [4, 5, 6, 8, 12] for a survey), and are found in various applications such as scheduling of truck routes to deliver

[1]Dept. of Architecture and Architectural Engineering, Kyoto University, Kyoto, 615-8540 Japan, email: naoki@archi.kyoto-u.ac.jp.

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

[2]National Astronomical Observatory of Japan, Mitaka, Tokyo, 181-8588, Japan, email: yano.t@nao.ac.jp

goods from a warehouse to retailers, material handling systems and computer communication networks. Recently, AGVs (automated guided vehicles) and material handling robots are often used in manufacturing systems, but also in offices and hospitals, in order to reduce the material handling efforts. Vehicle routing problems with p- and d-demands (VRPPD for short) for general networks have also been studied [13, 12]. They have recently received a considerable attention because it is more profitable if we can combine demands of delivery and pickup than carrying goods only for the delivery and returning to the depot without any goods. However, to the authors' knowledge, the capacitated vehicle routing problems with pickup and delivery demands on trees have not been studied yet. It should be noted that general pickup and delivery vehicle routing problems assume that the origin and destination of a demand are at arbitrary positions. In this respect, our problem treats a restricted case. However, our case where either the origin or the destination of transportation request is a depot has certain applications in scheduling stacker cranes in automated store and retrieval systems and in beer and soft drinks delivery where full bottles are delivered and empty bottles are collected.

The problem related to but simpler than VRPPD is the traveling salesman problem with pickup and delivery (TSPPD). This is a special case in which both of the total pickup and delivery demands in the network do not exceed the vehicle capacity. In the literature [1, 7, 11], general network has been considered as the underlying network, and pickup (resp. delivery) demands are requests to bring goods from the customer to the depot (resp. from the depot to the customer) as in our model. Mosheiov [11] proposed a $(1+\alpha)$-approximation algorithm for the problem, where α is the approximation ratio for the standard traveling salesman problem. Anily and Mosheiov [1] proposed the better approximation algorithm with approximation ratio 2 based on the minimum spanning tree. After constructing the minimum spanning tree, they used a linear time exact algorithm for the special case of TSPPD where the graph is a tree rooted at the depot. Gendreau et al. [7] proposed a heuristic with approximation ratio 3 and carried out an extensive comparison among several heuristics. They showed that their heursitic exhibited better performance than others.

The tree-shaped network can be found in buildings with simple structures of corridors and in simple production lines of factories. Capacitated vehicle scheduling problems on tree-shaped networks (TREE-CVRP for short) have recently been studied by several authors [2, 9, 10]. [10] considered the variant of TREE-CVRP where demand of each customer is not splittable and gave a 2-approximation algorithm. Hamaguchi and Katoh [9] and Asano et al. [2] studied the case where demand is allowed to be split. Hamaguchi and Katoh [9] proved the NP-hardness (in a strong sense) of TREE-CVRP and proposed a 1.5-approximation algorithm. Asano et al. [2] improved the approximation ratio of 1.5 by [9] to 1.35078.

This paper extends the work by [2, 9] in such a way that two types of transportation requests (i.e., pickup and delivery requests) are taken into consideration. Since TREE-CVRP is strongly NP-complete, our problem TREE-CVRPPD is also strongly NP-complete. Thus, we turn our attention to developing approximate algorithms for the problem. In this paper, we shall present a 2-approximation algorithm for the problem. If the 1.35078-approximation algorithm by [2] is applied to process pickup demands and delivery demands separately, this results in a trivial approximation ratio of 2.70156. In this paper, we shall present an improved 2-approximation algorithm for TREE-CVRPPD.

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 nodes. For nodes $u, v \in V$, 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. We assume throughout this paper that when we write an edge $e = (u, v)$, u is a

parent of v unless otherwise stated. For a node $v \in V - \{r\}$, let *parent*(v) denote the parent of v. Let $C(v)$ denote the set of children of v. If $w \in C(v)$ is a leaf, it is called a *leaf child* of v. For $u \in V$ and $v \in C(u)$, an edge (u, v) is called a *child edge* of u. An edge (u, v) is called a *leaf edge* or a *pendant edge* if v is a leaf. An edge which is not a leaf edge is called an *internal edge*. For any $v \in V$, let T_v denote the subtree rooted at v. For a connected subgraph H of T, let $w(H)$, $D^p(H)$ and $D^d(H)$ denote the sum of weights of edges in H, the sum of pickup and delivery demands of customers in H, respectively. Since customers are located on nodes, customers are often identified with nodes.

Suppose that we are given a set $S \subset V - \{r\}$ with $\sum_{v \in S} D^p(v) \leq 1$ and $\sum_{v \in S} D^d(v) \leq 1$. Let T' denote a minimal subtree that spans $S \cup \{r\}$. It is known [1] that a simple depth-first search generates a routing of a single vehicle that serves all the demands of customers in S with optimal tour length of $2 \sum_{e \in T'} w(e)$. For the completeness, we shall review the algorithm by Anily and Mosheiov [1]. Since the amount of goods loaded on a vehicle cannot exceed one (i.e., the vehicle capacity), we should be careful about the choice of the node to be visited next from among the children of the current vertex. Suppose $D^p(T_v)$ and $D^d(T_v)$ for all v are computed in advance. When we arrive at node v, we choose the child $w \in C(v)$ to be visited next such that

$$D^d(T_w) - D^p(T_w) = \max\{D^d(T_{w'}) - D^p(T_{w'}) \mid w' \in C(v) \text{ and is not visited yet}\}. \quad (1)$$

After satisfying all p- and d-demands in T_w, we backtrack to v. We continue this process until all demands are satisfied. It has been proved [1] that the routing schedule for T' obtained by the depth-first search based on the above rule does not get stuck, i.e., it does not violate the capacity constraint at any moment.

Thus, when we speak of a tour in this paper, we do not need to explicitly give a sequence of nodes 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 also need to specify the amounts of pickup and delivery demands of each customer served by the vehicle. A solution of TREE-CVRPPD consists of a set of tours. From the above discussion, we represent the tour of the j-th vehicle by

$$\{(D_j^p(v), D_j^d(v)) \mid v \in S_j\}, \quad (2)$$

where S_j is the set of customers for which some positive demands are served in the j-th tour, and $D_j^p(v)(\geq 0)$ and $D_j^d(v)(\geq 0)$ for $v \in S_j$ are the amounts of pickup and delivery demands that the j-th vehicle serves at v.

The total tour length of an optimal solution for TREE-CVRPPD is often referred to as the *optimal cost*. For an edge $e = (u, v)$, let

$$lb\_\#vehicles(e) = \max\{\lceil D^p(T_v) \rceil, \lceil D^d(T_v) \rceil\}. \quad (3)$$

$lb\_\#vehicles(e)$ represents a lower bound of the number of vehicles that must traverse edge e in a forward direction 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 $\max\{\lceil D^p(T_v) \rceil, \lceil D^d(T_v) \rceil\}$. Let $LB(e)$ denote the lower bound of the cost required to traverse edge e in an optimal solution. Then, since each vehicle must traverse an edge e at least twice in a forward direction and in a backward direction, we have

$$LB(e) = 2w(e) \cdot lb\_\#vehicles(e). \quad (4)$$

From this, we have the following lemma.

Lemma 1 *$LB^* = \sum_{e \in E} LB(e)$ gives a lower bound of the optimal cost of TREE-CVRPPD.*

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 by the following seven operations which are "safe" in the sense that they do not either increase the lower bound given in Lemma 1 or decrease the ratio of the optimal cost to the lower bound. The second step is to choose an appropriate subgraph and choose a strategy to serve (possibly partial) demands thereof.

The first operation R_1 is applied to nodes both of whose p- and d-demands are greater than or equal to 1. For such a node v, we allocate $k = \lfloor \min\{D^p(v), D^d(v)\} \rfloor$ vehicles to v (i.e., each vehicle delivers one unit of goods and returns to the depot by picking up one unit of goods). As a result of this operation, at least one of p- or d-demand at any leaf is decreased to less than one. Let $a = w(path(r, v))$. Then the cost required for scheduling such k vehicles is $2a$. Notice that for every edge e on $path(r, v)$, $LB(e)$ is decreased by k. Thus, after satisfying k units of p- and d-demands at v by k vehicles, the lower bound given by Lemma 1 decreases by $2ka$. Let P denote the original problem instance and P' the one resulted after satisfying k units of p- and d-demands at v by k vehicles. Let $cost(P)$, $cost_1$, $cost(P')$ denote the total cost required for the original problem P by our algorithm, the cost required by serving k units of p- and d-demands at v, and the cost for the remaining problem P' to be required by our algorithm, respectively, (i.e., $cost(P) = cost_1 + cost(P')$). Let $LB(P')$ be the lower bound for the problem P' and LB_1 be the decrease of lower bound by scheduling such k vehicles. From the above discussion, $cost_1 = LB_1$ holds. Thus, $\frac{cost(P)}{LB(P)} \leq \frac{cost(P')}{LB(P')}$ follows since

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

Note that this operation is apparently safe because this does not decrease the ratio of the optimal cost to the lower bound.

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. This reform operation is clearly safe. 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 two consecutive edges (u, u') and (u', u'') such that u'' is a unique child of u'. We just replace the edges (u, u') and (u', u'') by a single edge (u, u'') with $w(u, u'') = w(u, u') + w(u', u'')$

The fourth operation R_4 is to merge a subtree such that both p- and d-demands are less than or equal to 1 into a single edge. Namely, for an internal node v with $D^p(T_v) \leq 1$ and $D^d(T_v) \leq 1$, T_v is replaced by a single edge (v, v') with edge weight equal to $w(T_v)$, $D^p(v') = D^p(T_v)$ and $D^d(v') = D^d(T_v)$. Since $D^p(T_v) \leq 1$ and $D^d(T_v) \leq 1$ hold, and from the remark given in Section 2, this operation is also safe.

The fifth operation R_5 is applied to a leaf whose p- or d-demand is greater than or equal to one. We create a sufficient number of nodes connected to v with zero edge weight so that both p- and d-demands of every leaf are less than or equal to one. This operation is clearly safe.

The sixth operation R_6 is applied to a node v such that there are at least two leaves the sum of whose p- or d-demand exceeds one. Let v_1 and v_2 be such leaves. We then introduce a new node v' and a new edge (v, v') of zero weight and reconnect v_1 and v_2 to v' so that v' becomes a new parent of v_1 and v_2. This operation is clearly safe. We apply this operation only to the node having a non-leaf child because otherwise it will be infinitely repeated. We need this operation only for the technical purpose.

The seventh reforming operation R_7 is to merge leaves. For a node v, let $\{v_1, v_2, \ldots, v_k\}$ be a set of its leaf children. By w_i we denote the weight of the edge between v and v_i. We examine

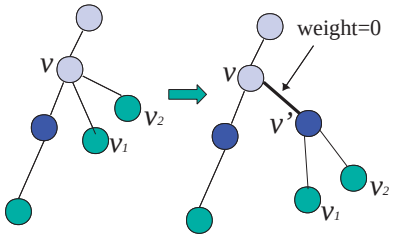

Figure 1: The operation R_6.

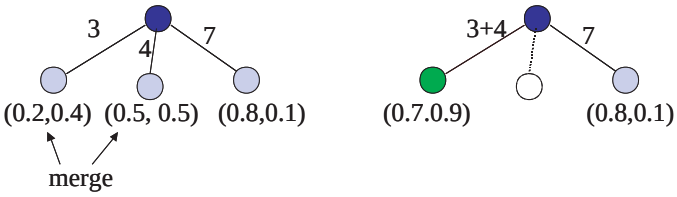

Figure 2: Merge operation R_7.

every pair of leaves. For the pair (v_i, v_j) we check whether the sum of their p-demands exceeds 1. If $D^p(v_i) + D^p(v_j) \leq 1$ and $D^d(v_i) + D^d(v_j) \leq 1$, then we merge them. Exactly speaking, we remove the leaf v_j together with its associated edge (u, v_j) after replacing the p- and d-demands of v_i with $D^p(v_i) + D^p(v_j)$ and $D^d(v_i) + D^d(v_j)$, respectively, 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 2 illustrates how this merging process proceeds.

Notice that after the above seven operations are repeatedly applied until they cannot be applied anymore, both of p- and d-demands of every leaf is at most one and at least one of them is less than one.

4 Approximation Algorithm

The approximation algorithm to be presented in this paper repeats the following two steps:

Step 1: Apply seven reform operations described in the previous section until we cannot perform any operation.
Step 2: Focusing on a connected subgraph H of the resulting graph such that (i) H contains the root, (ii) both $D^p(H)$ and $D^d(H)$ are small enough, and (iii) $\lceil D^p(H) \rceil = \lceil D^d(H) \rceil$, we prepare a few vehicles to (possibly partially) serve the demands in H.

One iteration of Steps 1 and 2 is called a *round*.

Theorem 1 *The approximation of our algorithm for TREE-CVRPPD is* 2.

Proof. The proof technique is similar to the one by Hamaguchi and Katoh [9] and Asano et al. [2]. The theorem can be proved by induction on the number of rounds. Whenever the sum of p- or d-demands in the tree exceeds one, we perform the reforming operations to have a subgraph that falls into one of five cases shown below, and design how to serve the demands in the subgraph. Then, we apply the reforming operations again to the resulting tree and repeat this process until the remaining p-demands and d-demands are both less than or equal to one. This is the base case. For the base case, the algorithm mentioned in Section 2 finds an optimal tour by a single vehicle.

Assuming that the theorem holds for problem instances that require at most k rounds, we consider the problem instance P of TREE-CVRPPD for which our algorithm requires $k+1$ rounds. Each time we find a subgraph and apply an appropriate strategy. Let P' be the problem instance of TREE-CVRPPD 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

$$\frac{cost(P)}{LB(P)} \leq \frac{cost_1 + cost(P')}{LB_1 + LB(P')}. \tag{5}$$

Since $cost(P')/LB(P') \leq 2$ holds from the induction hypothesis, it suffices to prove $cost_1/LB_1 \leq 2$.

As we shall prove below (Lemmas 2 through 10), the above inequality holds in any of Cases 1 through 5 to be explained below. Thus, we have the theorem. □

In order to explain how to determine the subgraph which we focus on and how to design vehicle scheduling for the subgraph, we need the following definition. An internal edge $e = (u, v)$ is called *p-dominant*, *d-dominant*, and *pd-balanced* if $\lceil D^p(T_v) \rceil \geq \lceil D^d(T_v) \rceil + 1$, $\lceil D^d(T_v) \rceil \geq \lceil D^p(T_v) \rceil + 1$, and $\lceil D^p(T_v) \rceil = \lceil D^d(T_v) \rceil$, respectively.

4.1 Cases 1 through 4

Case 1: There exists a node v with two leaf children such that both the sum of p-demands and that of d-demands are at least one. From reform operations R_4 and R_7, we have $1 \leq D^p(v_1) + D^p(v_2) \leq 2$ and $1 \leq D^d(v_1) + D^d(v_2) \leq 2$. In this case, we consider the subgraph consisting of $path(r, v)$ and edges connecting v and its two or three leaf children (see Fig. 3). In Figures 3 through 5, a leaf is represented as a black square, and a leaf edge as a grey line. Also, p-dominant and d-dominant edges are represented by thick and thin lines, respectively, and a double line represents a path. In this case, we prepare two vehicles, one for v_1 and the other for v_2 so that the first vehicle (resp. the second vehicle) serve the p- and d-demands for v_1 (resp. v_2). Let $e_1 = (v, v_1)$ and $e_2 = (v, v_2)$. Let a denote the weight of $path(r, v)$. Then the the cost of the tour required for the first (resp. the second) vehicle is $2a + 2w(e_1)$ (resp. $2a + 2w(e_2)$). Thus, the total cost for these two vehicles is $4a + 2w(e_1) + 2w(e_2)$. The decrease of the lower bound is at least $2a + 2w(e_1) + 2w(e_2)$ because both of p- and d-demands served by two vehicles are at least one, and hence $lb\_\#vehicles(e)$ for each edge on $path(r, v)$ is decreased at least by one. Thus, we have the following lemma.

Lemma 2 *In Case 1, the approximation ratio is at most* 2.

Case 2: There exists a node v such that (i) there are two leaf children v_1 and v_2 the sum of whose p-demands is at least one, and (ii) all edges along $path(r, v)$ are p-dominant. We prepare two vehicles, each serving the demands of v_1 and v_2, respectively. Let $e_1 = (v, v_1)$ and $e_2 = (v, v_2)$, and let a be the one defined in Case 1. Then the the cost of tours of these two vehicles is $4a + 2w(e_1) + 2w(e_2)$. The decrease of the lower bound is at least $2a + 2w(e_1) + 2w(e_2)$ since every edge on $path(r, v)$ is p-dominant and $LB(e)$ for each edge on $path(r, v)$ is decreased by at least one.

Lemma 3 *In Case 2, the approximation ratio is at most* 2.

Case 3: There exists a node v such that (i) there are two leaf children the sum of whose d-demands is at least one, and (ii) all edges along $path(r, v)$ are d-dominant. This case can be treated similarly to Case 2.

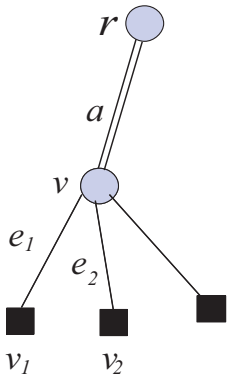

Figure 3: The illustration of Case 1.

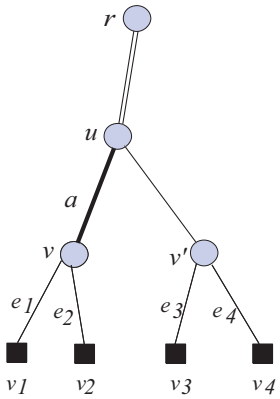

Figure 4: The illustration of Case 4.

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

Case 4: There exist nodes u, v and v' such that (i) both v and v' are descendants of u such that all edges on $path(u, v)$ are p-dominant and all edges on $path(u, v')$ are d-dominant, (ii) there are two leaf children v_1 and v_2 of v the sum of whose p-demands is at least one, and there are two leaf children v_3 and v_4 of v' the sum of whose d-demands is at least one.

Fig. 4 illustrates this case. If the sum of d-demands of v_1 and v_2 is larger than or equal to one, this reduces to Case 1. The same remark holds if the sum of p-demands of v_3 and v_4 is larger than or equal to one. Thus, we can assume that the sum of d-demands of v_1 and v_2 is less than one, and the sum of p-demands of v_3 and v_4 is also less than one. Let $e_1 = (v, v_1)$, $e_2 = (v, v_2)$, $e_3 = (v', v_3)$, $e_4 = (v', v_4)$, and assume without loss of generality that $w(e_1) \geq w(e_2)$ and $w(e_3) \geq w(e_4)$. We prepare two vehicles. The first vehicles visits v_1 and v_2 to serve the whole p-demand of v_1 and a partial p-demand of v_2 so that the total p-demand of v_1 and v_2 served by the vehicle is equal to one, and to serve the whole d-demands of v_1 and v_2. The remaining p-demand at v_2 is left for the succeeding rounds. Notice that the order of visits may be different from what is written here in order to preserve capacity constraint. The second vehicle visits v_3 and v_4 to serve the whole d-demand of v_3 and a partial d-demand of v_4 so that the total d-demand of v_3 and v_4 served by the vehicle is equal to one, and to serve the whole p-demands of v_3 and v_4. The remaining d-demand at v_4 is left for succeeding rounds. Let $a = w(path(r, u))$, $b_1 = w(path(u, v))$ and $b_2 = w(path(u, v'))$. The costs required for the first and second vehicles are $2a + 2b_1 + 2w(e_1) + 2w(e_2)$ and $2a + 2b_2 + 2w(e_3) + 2w(e_4)$, respectively. Thus, the cost required by the two vehicles is given by

$$cost = 4a + 2b_1 + 2b_2 + 2w(e_1) + 2w(e_2) + 2w(e_3) + 2w(e_4).$$

The decrease of the lower bound by the first and the second vehicles is given by

$$lb = 2a + 2b_1 + 2b_2 + 2w(e_1) + 2w(e_3).$$

Thus ratio of *cost* to *lb* is

$$\begin{aligned} \frac{cost}{lb} &= \frac{4a + 2b_1 + 2b_2 + 2w(e_1) + 2w(e_2) + 2w(e_3) + 2w(e_4)}{2a + 2b_1 + 2b_2 + 2w(e_1) + 2w(e_3)} \\ &\leq \frac{4a + 2b_1 + 2b_2 + 4w(e_1) + 4w(e_3)}{2a + 2b_1 + 2b_2 + 2w(e_1) + 2w(e_3)} \leq 2. \end{aligned} \tag{6}$$

The first inequality is derived from $w(e_1) \geq w(e_2)$ and $w(e_3) \geq w(e_4)$. Thus,

Lemma 5 *In Case 4, the approximation ratio is at most* 2.

4.2 Case 5

Case 5: This case is much more involved than the previous ones. Suppose any of Cases 1 through 4 does not hold. We focus on the node u such that edge $(parent(u), u)$ is pd-balanced and there is not any pd-balanced edge in T_u. Before explaining the subgraph we shall focus on, we need the following lemma.

Lemma 6 *Let $v(\neq u)$ be an internal node in T_u which has a non-leaf child, and let $v' = parent(v)$. (i) If (v', v) is p-dominant (resp. d-dominant) and v has only one non-leaf child w, (v, w) is also p-dominant (resp. d-dominant). (ii) If (v', v) is p-dominant (resp. d-dominant) and v has at least two non-leaf children, at least one non-leaf child edge of v is p-dominant (resp. d-dominant).*

Proof. (i) Suppose v has a leaf child l. Notice that v has a unique leaf child since otherwise the sum of p- or d-demands of leaves is greater than or equal to one due to merge operation R_4 and hence R_5 creates a new internal edge connecting these leaves which makes v be not allowed to have a leaf child. Thus, v has at most one leaf child. Then (v, w) cannot be d-dominant from $D^d(l) \leq 1$. From the assumption of T_u (v, w) cannot be pd-balanced either. Thus, (i) follows.

(ii) Since v has at most one leaf child, (ii) immediately follows. □

From this lemma, if v has two non-leaf children w and w' such that (v, w) is p-dominant and (v, w') is d-dominant, there exists a subgraph of Case 4. The reason is as follows: From Lemma 6, there is a path from v passing through (v, w) to a node x having only leaf children such that all edges on the path is p-dominant. Similarly there is a path from v passing through (v, w') to a node x' having only leaf children such that all edges on the path is d-dominant.

Thus, from this lemma and since it is assumed that any of Cases 1 through 4 does not hold, we can assume that there exists a node u such that (i) either all internal edges of T_u are p-dominant, or all of them are d-dominant, and (ii) edge $(parent(u), u)$ is pd-balanced. We assume that every internal edge of T_u is p-dominant without loss of generality (the other case can be similarly treated).

We focus on an internal node $v \in T_u$ such that all children of v are leaves. We choose arbitrary two leaf children v_1 and v_2 of v and then choose a set of leaves $\hat{L} = \{l'_1, l'_2, \ldots, l'_h\}$ whose parent is on the path $path(u, v)$ so that the following two equations are satisfied. Here, the subscript i of l'_i is given so that $parent(l'_i)$ is a descendant of $parent(l'_j)$ if $i < j$.

$$1 + \lceil \sum_{i=1}^{h} D^p(l'_i) \rceil = \lceil \sum_{i=1}^{2} D^d(v_i) + \sum_{i=1}^{h} D^d(l'_i) \rceil \tag{7}$$

and

$$1 + \lceil \sum_{i=1}^{j} D^p(l'_i) \rceil = 1 + \lceil \sum_{i=1}^{2} D^d(v_i) + \sum_{i=1}^{j} D^d(l'_i) \rceil \quad \text{for } j = 0, 1, \ldots, h-1. \tag{8}$$

We will leave the proof of the existence of such $\hat{L}$ and the explanation of how such $\hat{L}$ can be obtained to the next subsection. We shall explain how we can construct a vehicle scheduling for the leaves of $\{v_1, v_2\} \cup \hat{L}$, assuming that $\hat{L}$ is given. Let

$$n = 1 + \lceil \sum_{i=1}^{h} D^p(l'_i) \rceil. \tag{9}$$

We first prepare n vehicles to serve p- and d-demands of $\{v_1, v_2\} \cup \hat{L}$ so that every vehicle (possibly except the last one) serves a unit amount of p- and d-demands. Let n vehicles be numbered from 1 through n. We assume $w(v, v_1) \geq w(v, v_2)$ without loss of generality. The

first vehicle satisfies the entire p-demand at v_1 and a partial p-demand of v_2 so that the total amount is equal to one. The remaining demand of v_2 is left for the succeeding rounds. For p-demands at leaves of $\hat{L}$, the succeeding vehicles are assigned to the leaves of $\hat{L}$ in the order of $l'_1, l'_2, \ldots, l'_h$ so that the p-demand of a leaf with smaller subscript is assigned to the vehicle with smaller or the same number. Namely, we first place $l'_1, l'_2, \ldots, l'_h$ in a queue in this order, and prepare the vehicles $2, 3, \ldots n$. We iteratively select the leaf l'_i located at the top of the queue, and assign its p-demand to the vehicle (say, vehicle k. Initially k is set to 2.). If the entire p-demand of l'_i is assigned to vehicle k, delete l'_i from the queue and select the next leaf. If the vehicle capacity becomes full (i.e. a unit amount of p-demand is loaded on the vehicle), we select the next vehicle (vehicle $k+1$) and assign the remaining p-demand of l'_i to the vehicle. We repeat this process until all p-demands are assigned. We call this algorithm *Assignment*. Eventually, every vehicle is assigned a unit amount of p-demands possibly except the last one. Notice that for every leaf l'_j, at most two vehicles serve its p-demand. See Example 1 below for an illustration.

For the d-demands of $\{v_1, v_2\} \cup \hat{L}$, the assignment of d-demands to vehicles are made in essentially the same manner as for p-demands. Namely, preparing n vehicles numbered from 1 through n and placing leaves v_1, v_2, $l'_1, l'_2, \ldots, l'_h$ in the queue in this order, the assignment of d-demands to vehicles are made in such a way that (i) every vehicle serves a unit amount of d-demands possibly except the last one, and (ii) for any l'_i and l'_j with $i < j$, if d-demand of l'_i is served by the vehicle k, d-demand of l'_j is never served by a vehicle whose number is smaller than k. Notice that for every leaf l'_i, at most two vehicles are assigned to serve its d-demand.

The vehicle scheduling given by Algorithm Assignment is denoted by S. Therefore, S satisfies all p- and d-demands of $\{v_1, v_2\} \cup \hat{L}$ possibly except the p-demand of v_2 and that every vehicle (possibly except the last one) serves one unit of p- and d-demands.

Although for each leaf $l'_i \in \hat{L}$, the number of vehicles which visit l'_i to satisfy p-demand (resp. d-demand) is at most two, respectively, the total number of vehicles which visit l'_i may exceed two (see Example 1 below where three vehicles visit a leaf l_2). If so, the approximation ratio for this round may exceed two when the edge weight $w(parent(l'_i), l'_i)$ is very large compared with other edge weights. In order to avoid this, we adjust S by executing the following steps depending on the four cases (actually, we do not need do anything except Cases (iii) and (iv)). The resulting scheduling is denoted by S':

Adjustment of the scheduling

Case (i): There is only one vehicle (say, k) which visits l'_i to satisfy p-demand, and also there is only one vehicle (say, k') which visits l'_i to satisfy d-demand. In this case, the total number of vehicles which visit l'_i is two, and thus we do not need do anything.

Case (ii): There are two vehicles (say, k and $k+1$) which visit l'_i to satisfy p-demand while there is only one vehicle (say, k') which visits l'_i to satisfy d-demand. If $k' < k$ or $k' > k+1$, this contradicts that (8) holds for $j = i-1$ or $j = i$, respectively. Thus, $k' = k$ or $k+1$. Therefore, the total number of vehicles which visit l'_i is two, and we do not need do anything.

Case (iii): There is only one vehicle (say, k) which visits l'_i to satisfy p-demand while there are two vehicles (say, k' and $k'+1$) which visit l'_i to satisfy d-demand. Similarly to Case (ii), $k' = k-1$ or $k' = k-2$ holds from (7) and (8). $k' = k-1$ holds only if $i = h$ from (7). In this case, the total number of vehicles which visit l'_i is two, and we do not need an additional vehicle. $k' = k-2$ holds only if $i < h$ and $\lceil \sum_{j=1}^{i-1} D^p(l'_j) \rceil = \sum_{j=1}^{i-1} D^p(l'_j)$. In this case, there are three vehicles to visit l'_i and thus we prepare a new vehicle which serves all p- and d-demands at l'_i instead of such three vehicles.

Case (iv): There are two vehicles (say, k and $k+1$) which visit l'_i to satisfy p-demand, and also there are two vehicles (say, k' and $k'+1$) which visit l'_i to satisfy d-demand. As proved in Cases (ii) and (iii), $k = k'+1$ holds. In this case, there are three vehicles in total which visit l'_i. Thus, instead of these three vehicles, we assign one new vehicle to satisfy all p- and d-demands at l'_i. Example 1 illustrates this case.

Table 1: Schedule S and S' for Example 1. The numbers shown at lines 4, 6 and 8 represent those of vehicles assigned, and those at lines 5 and 7 shows the amount of demand assigned to the corresponding vehicle.

		v_1	v_2	l_1	l_2	l_3
input	p-demand	0.8	0.8	0.7	0.5	0.6
	d-demand	0.3	0.2	0.4	0.8	0.6
Schedule S for p-demand	vehicle No.	1	1	2	2, 3	3
	p-demand	0.8	0.2	0.7	0.3, 0.2	0.6
Schedule S for d-demand	vehicle No.	1	1	1	1, 2	2, 3
	d-demand	0.3	0.2	0.4	0.1, 0.7	0.3, 0.3
Schedule S'	vehicle No.	1	1	1, 2	4	2, 3

Example 1: We shall show an illustrative example. In Figure 5, $L = \{l_1, l_2, l_3\}$. Let p- and d-demands of v_1, v_2, l_1, l_2, l_3 be given as in Table 1. Notice that $\hat{L} = L$ holds because $\hat{L} = L$ satisfies (7) and (8). Vehicle assignment by schedule S is shown at lines from 4 to 7 in the table. Since three vehicles are assigned to l_2, a new vehicle (vehicle 4) is assigned to l_2 to obtain a final solution.

Lemma 7 *For any i with $1 \leq i \leq h$, the number of new vehicles we prepare according to Case (iii) or (iv), is at most $q-2$, where $q = 1 + \lceil \sum_{j=1}^{j} D^p(l'_j) \rceil$.*

Proof. We first claim that the number of leaves in $\{v_1, v_2\} \cup \{l'_1, l'_2, \ldots, l'_i\}$ such that schedule S assigns two vehicles to serve its p-demand is at most $q-2$. From the way of constructing the schedule S, if two vehicles visit the leaf to serve its p-demand, their vehicle numbers are consecutive. Also, at most two vehicles visit each leaf l'_j in order to serve p-demand because $D^p(l) \leq 1$ holds. Therefore, there are at most $q-1$ leaves for which schedule S assigns two vehicles to serve its p-demand. However, from the way of constructing the schedule S, there is no leaf which both vehicles 1 and 2 visit. Thus, the claim follows. Suppose a new vehicle is prepared for leaf l'_j according to Case (iii). Then the schedule S assigns a single vehicle (say, k) for l'_j to serve its p-demand and two vehicles $k-2$ and $k-1$ to serve its d-demand. The condition $\lceil \sum_{j=1}^{i-1} D^p(l'_j) \rceil = \sum_{j=1}^{i-1} D^p(l'_j)$ implies that vehicle $k-1$ is scheduled in S to serve p-demand of l'_{j-1}. Thus, there is no leaf for which both vehicles $k-1$ and k are scheduled in S to serve its p-demand. Therefore, if g vehicles are prepared according to Case (iii) by the schedule S', this observation implies that the number of leaves in $\hat{L}$ whose p-demand is split in S is at most $q-2-g$. Thus, at most $q-2-g$ vehicles are prepared according to Case (iv) in S'. This proves the lemma. □

In order to prove that the scheduling S of vehicles in Case 5 attains a 2-approximation, we shall show that for every edge e which some vehicle passes in this scheduling, the ratio of the cost to the decrease of the lower bound $LB(e)$ is at most 2.

(1) e is a leaf edge whose end vertex is v_1 or v_2. In this case, we consider such two edges together. The cost required to pass these two edges is $2w(v, v_1) + 2w(v, v_2)$, and the decrease of the lower bound concerning these two edges is $2w(v, v_1)$. From the assumption of $w(v, v_1) \geq w(v, v_2)$, we have $(2w(v, v_1) + 2w(v, v_2))/2w(v, v_1) \leq 2$.

(2) e is a leaf edge whose end vertex is a leaf of $\hat{L}$. In this case, at most two vehicles pass the edge to satisfy all p- and d-demands at the leaf. Thus, the ratio of the cost required to pass the edge e to the decrease of $LB(e)$ is clearly at most 2.

(3) e is on $path(u, v)$. Notice that e is p-dominant. Suppose that e is between the path connecting l'_j and l'_{j+1}. The amounts of p- and d-demands served by the vehicles passing e are $1 + \sum_{i=1}^{j} D^p(l'_i)$ and $\sum_{i=1}^{2} D^d(v_i) + \sum_{i=1}^{j} D^d(l'_i)$, respectively. From (8) and since e is p-dominant, the decrease of $lb\_\#vehicles(e)$ is at least $\lceil \sum_{i=1}^{j} D^p(l'_i) \rceil$. Let $q = 1 + \lceil \sum_{i=1}^{j} D^p(l'_i) \rceil$. The scheduling algorithm first prepares q vehicles and then additionally prepares at most $q-2$ vehicles from Lemma 7. Thus, the number of vehicles passing e is at most $2q-2$. Hence, the ratio of the cost required to traverse the edge e to the decrease of $LB(e)$ is clearly at most 2.

(4) e is on $path(r, u)$. Similarly to (3) above, it follows that the number of vehicles passing e is at most $2n-2$. The amounts of p- and d-demands satisfied by the vehicles passing e are $1 + \sum_{i=1}^{h} D^p(l'_i)$ and $\sum_{i=1}^{2} D^d(v_i) + \sum_{i=1}^{h} D^d(l'_i)$, respectively. From (7), the decrease of $lb_{\#}vehicles(e)$ is at least $n-1$. This implies that the ratio of the cost required to traverse the edge e to the decrease of $LB(e)$ is clearly at most 2.

Lemma 8 *In Case 5, the approximation ratio is at most* 2.

4.3 Existence of $\hat{L}$

We now prove the existence of $\hat{L}$. We first show the following lemma.

Lemma 9 *For an internal node $v \in T_u$ such that all children of v are leaves, $D^p(v') + D^p(v'') \geq 1$ and $D^d(v') + D^d(v'') < 1$ hold for any two children v' and v'' of v.*

Proof.

Let $C(v) = \{v_1, v_2, \ldots, v_g\}$ where leaves are rearranged in the nondecreasing order of $D^p(v_i) - D^d(v_i)$.

We first show that $\sum_{i=1}^{2} D^p(v_i) \geq 1$ and $\sum_{i=1}^{2} D^d(v_i) < 1$ hold. Suppose otherwise. From merging operation R_6, either $\sum_{i=1}^{2} D^p(v_i) \geq 1$ or $\sum_{i=1}^{2} D^d(v_i) \geq 1$ holds. We derive a contradiction by assuming $\sum_{i=1}^{2} D^d(v_i) \geq 1$. If $\sum_{i=1}^{2} D^p(v_i) \geq 1$, we have the subgraph satisfying Case 1. Thus, $\sum_{i=1}^{2} D^p(v_i) < 1$ holds. It then follows that v must have a leaf v' other than v_1 and v_2 such that $D^p(v') > D^d(v')$ because the edge $(parent(v), v)$ is p-dominant. If v has only three leaf children, $D^p(T_v) < 2$ holds and hence $(parent(v), v)$ cannot be p-dominant, a contradiction. Thus, v has at least four children. Then there exist leaves v_3 and v_4 of v which are different from v_1 and v_2 such that $D^p(v_3) > D^d(v_3)$ and $D^p(v_4) > D^d(v_4)$ because otherwise $(parent(v), v)$ cannot be p-dominant. If $\sum_{i=3}^{4} D^d(v_i) \geq 1$, the subgraph consisting of the leaves v_3 and v_4 together with $path(u, v)$ satisfies Case 1. Thus, $\sum_{i=3}^{4} D^d(v_i) < 1$ holds. Also, $\sum_{i=3}^{4} D^p(v_i) \geq 1$ since v_3 and v_4 can be merged otherwise. Hence, the subgraph consisting of v_1, v_2, v_3, v_4 and $path(u, v)$ satisfies the condition of Case 4 where u, v, v' of Case 4 are the same node. Therefore, a contradiction is again derived. Thus, $\sum_{i=1}^{2} D^p(v_i) \geq 1$ and $\sum_{i=1}^{2} D^d(v_i) < 1$ hold.

Now let us consider arbitrary two children v_j and v_k. From the choice of v_1 and v_2, it follows that $D^p(v_j) + D^p(v_k) > D^d(v_j) + D^d(v_k)$. Similarly to the case of v_1 and v_2, we have $D^p(v_j) + D^p(v_k) > 1$ and $D^d(v_j) + D^d(v_k) < 1$. This proves the lemma. □

We then consider two arbitrary leaves v_1 and v_2 of v, and let

$$L = \{l \in V \mid l \text{ is a leaf}, parent(l) \in path(u, parent(v))\}$$

When v has an odd number of children, let v_3 be a child of v other than v_1 and v_2 and let

$$L = \{v_3\} \cup \{l \in V \mid l \text{ is a leaf}, parent(l) \in path(u, parent(v))\}.$$

We will choose an appropriate subset $\hat{L}$ of L which satisfy (7) and (8). Let $L = \{l_1, l_2, \ldots, l_m\}$ (see Fig. 5). Here the subscript i of l_i is given so that if $parent(l_i)$ is a proper descendant of $parent(l_j)$, $i < j$ holds.

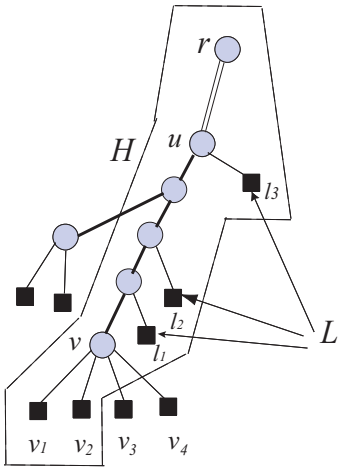

Figure 5: The illustration of the subgraph H considered in Case 5.

Lemma 10

$$\text{(i)} \quad \lceil \sum_{i=1}^{2} D^d(v_i) + \sum_{l \in L} D^d(l) \rceil \geq \lceil \sum_{i=1}^{2} D^p(v_i) + \sum_{l \in L} D^p(l) \rceil. \tag{10}$$

$$\text{(ii)} \quad \lceil \sum_{i=1}^{2} D^d(v_i) + \sum_{l \in L} D^d(l) \rceil \geq 1 + \lceil \sum_{l \in L} D^p(l) \rceil. \tag{11}$$

Proof. Since $\sum_{i=1}^{2} D^p(v_i) \geq 1$ holds from Lemma 9, (ii) is immediate from (i). Thus, we shall prove (i) only. Let F be the set of internal edges that branch from $path(u, v)$. Namely, F is the set of internal edges $e = (x, y)$ such that x is on $path(u, v)$ but y is not. Let V_F be the set of such vertices y. Let C' be the set of children of v other than v_1 and v_2 (when v has an odd number of children, v_3 is excluded from C'). Then we have

$$D^p(T_u) = \sum_{i=1}^{2} D^p(v_i) + \sum_{l \in L} D^p(l) + \sum_{y \in V_F} D^p(T_y) + \sum_{v' \in C'} D^p(v') \tag{12}$$

and

$$D^d(T_u) = \sum_{i=1}^{2} D^d(v_i) + \sum_{l \in L} D^d(l) + \sum_{y \in V_F} D^d(T_y) + \sum_{v' \in C'} D^d(v'). \tag{13}$$

From the assumption, all edges $e \in F$ are p-dominant, i.e. $\lceil D^p(T_y) \rceil > \lceil D^d(T_y) \rceil$ holds for any $y \in V_F$. Thus,

$$\sum_{y \in V_F} \lceil D^p(T_y) \rceil > \sum_{y \in V_F} \lceil D^d(T_y) \rceil. \tag{14}$$

Noting the following fact,

Fact 1: $\lceil a \rceil > \lceil b \rceil$ and $\lceil c \rceil > \lceil d \rceil$ imply $\lceil a + c \rceil > \lceil b + d \rceil$ for any positive numbers a, b, c, d,

(14) implies

$$\lceil \sum_{y \in V_F} D^p(T_y) \rceil > \lceil \sum_{y \in V_F} D^d(T_y) \rceil. \tag{15}$$

If $C' \neq \emptyset$, $D^p(v') + D^p(v'') \geq 1 > D^d(v') + D^d(v'')$ holds for every pair of $v', v'' \in C'$ from Lemma 9. Hence, using Fact 1, we have

$$\lceil \sum_{v' \in C'} D^p(v') \rceil > \lceil \sum_{v' \in C'} D^d(v') \rceil \tag{16}$$

because $|C'|$ is even. Thus, if (i) does not hold, this implies $\lceil D^p(T_u) \rceil > \lceil D^d(T_u) \rceil$ from (10) and (11). This contradicts that $\lceil D^p(T_u) \rceil = \lceil D^d(T_u) \rceil$ holds since $(parent(u), u)$ is assumed to be pd-equivalent. Therefore, the lemma follows. □

We shall explain how we compute $\hat{L} \subseteq L$ satisfying (7) and (8). The algorithm is rather greedy and is described as follows.

Algorithm **Find_$\hat{L}$**

Step 1: $\hat{L} = \emptyset, i = 0$.
Step 2: Let $i = i + 1$.
Step 3: If $1 + \lceil D^p(l_i) + \sum_{l \in \hat{L}} D^p(l) \rceil - (\lceil \sum_{i=1}^{2} D^d(v_i) + D^d(l_i) + \sum_{l \in \hat{L}} D^d(l) \rceil) = 1$, let $\hat{L} = \hat{L} \cup \{l_i\}$ and return to Step 2.
Step 4: If $1 + \lceil D^p(l_i) + \sum_{l \in \hat{L}} D^p(l) \rceil - (\lceil \sum_{i=1}^{2} D^d(v_i) + D^d(l_i) + \sum_{l \in \hat{L}} D^d(l) \rceil) = 2$,
let $\hat{L} = \emptyset$ and return to Step 2.
Step 5: If $1 + \lceil D^p(l_i) + \sum_{l \in \hat{L}} D^p(l) \rceil - (\lceil \sum_{i=1}^{2} D^d(v_i) + D^d(l_i) + \sum_{l \in \hat{L}} D^d(l) \rceil) = 0$, let $\hat{L} = \hat{L} \cup \{l_i\}$ and $u = parent(l_i)$. Halt after outputting $\hat{L}$.

Lemma 11 *Algorithm Find_$\hat{L}$ correctly computes $\hat{L}$ which satisfy (7) and (8).*

Proof. Suppose condition of Step 4 holds for some i. Then, $\lceil D^p(l_i) + \sum_{l \in \hat{L}} D^p(l) \rceil > \lceil D^d(l_i) + \sum_{l \in \hat{L}} D^d(l) \rceil$ holds. Let $L' = \{l_{i+1}, l_{i+2}, \ldots, l_m\}$ which is a set of leaves in L not yet examined by the algorithm. From Lemma 10(i) (inequality (10)), $L' \neq \emptyset$ must hold. Using Fact 1,

$$1 + \lceil \sum_{l \in L'} D^p(l) \rceil \leq \lceil \sum_{i=1}^{2} D^d(v_i) + \sum_{l \in L'} D^d(l) \rceil \tag{17}$$

holds which implies that the inequality of (11) again holds for $L = L'$. Thus, we can again apply the algorithm. Since L is a finite set, the condition of Step 5 eventually holds. It should be noticed that $i = m$ does not hold at Step 2 when returning from Step 3 or 4. □

Notice that if the scheduling algorithm in Case 5 is changed so that p-demand of v_2 is not left for future rounds, the above lemma does not hold any more because (17) does not always hold for this case.

4.4 Lower bound

We now show that the approximation ratio 2 proved in Theorem 1 is tight. Figure 6 illustrates such an example. The numbers (α, β) attached to a leaf represent its p- and d-demands, and the number attached to an edge represents the edge weight where ϵ is a sufficiently small positive number. It is easy to see that none of Cases 1 through 4 holds, but Case 5 holds for this example. Applying the method for treating Case 5, two vehicles are prepared to serve the demands for T_u. The first vehicle serves the whole p- and d-demands at v_1, and a partial p-demand of 0.2 and the whole d-demand of 0.2 at v_2. It also serves a partial d-demand of 0.6 at l_1. The second vehicle serves the remaining p-demand of 0.6 at v_2, and the whole p-demand of 0.2 and the remaining d-demand of 0.2 at l_1. The total cost required by two vehicles is $4a + 14 + 2\epsilon$.

On the other hand, the optimal vehicle scheduling is as follows: The first vehicle satisfies all demands at v_1, and the second one does all demands at v_2 and l_1. The total cost for this scheduling is $2a + 12 + \epsilon$. As a increases, the approximation ratio approaches 2.

Table 2: Schedules obtained by our algorithm and an optimal schedule for an example given in Section 4.4. The numbers shown at lines 4, 6, 8 and 10 represent those of vehicles assigned, and those at lines 5, 7, 9 and 11 shows the amount of demand assigned to the corresponding vehicle.

		v_1	v_2	l_1
input	p-demand	0.8	0.8	0.2
	d-demand	0.2	0.2	0.8
Schedule obtained by our algorithm for p-demand	vehicle No.	1	1,2	2
	p-demand	0.8	0.2, 0.6	0.2
Schedule obtained by our algorithm for d-demand	vehicle No.	1	1	1,2
	d-demand	0.2	0.2	0.6, 0.2
optimal schedule for p-demand	vehicle No.	1	2	2
	p-demand	0.8	0.8	0.2
optimal schedule for d-demand	vehicle No.	1	2	2
	d-demand	0.2	0.2	0.8

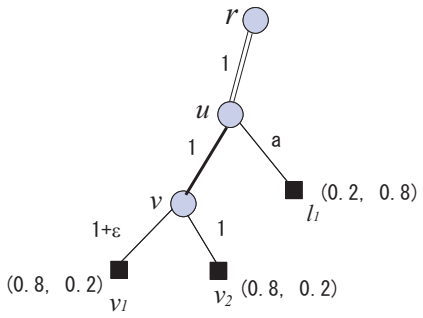

Figure 6: The illustration of the graph considered in Section 4.4.

Theorem 2 *The approximation ratio of the proposed algorithm is tight.*

4.5 Running time

Among seven reforming operations, R_1, R_2 and R_5 are applied only once at the first round of the algorithm. R_1 and R_5 require $O(|V| + \sum_{v \in V}(D^p(v) + D^d(v)))$ time while R_2 requires $O(|V|)$ time. Let V' denote the node set of the graph after applying R_1, R_2 and R_5 as much as possible in the first round. Notice that $|V'|$ may become $O(|V| + \sum_{v \in V}(D^p(v) + D^d(v)))$. Performing operations R_3, R_4 and R_6 in one round can be carried out by a depth-first search starting from the root which requires $O(|V'|)$ time in total. The total time to execute the operations R_7 in the first round is $O(|V'|^2)$ time since for each leaf v_i testing whether there is another leaf v_j that is mergeable with v_i can be done in $O(|V'|)$ time, and merging two leaves decreases the number of nodes by one. In the suceedings rounds, the execution of R_7 requires $O(|V'|)$ time. The reason is as follows. In one round, for some leaves all p- and d-demands of them are satisfied, and those leaves will be deleted. In addition, in Case 4 or 5 p- or d-demand

for at most two leaves are partially satisfied and a certain amount of demand is remaining. In this case, R_7 is applied to such leaves to check whether they are mergeable to other leaves. This clearly requires $O(|V'|)$ time.

Thus, the total time to execute reforming operations is $O(|V'|^2) = O(|V|^2(\sum_{v \in V}(D^p(v) + D^d(v)))^2)$ for the first round while it is $O(|V'|)$ for the succeeding rounds.

In each round, we need to determine which one of Cases 1 through 5 holds. For this, we need to compute $D^p(v), D^d(v)$ for every $v \in V$ and determine whether each edge is p-dominant, d-dominant or pd-balanced. This can be done in $O(|V'|)$ time by executing the depth-first search starting from r. From this information obtained, we can determine whether one of Cases 1, 2 and 3 holds. In order to determine whether Case 4 holds or not, we need to find a node u such that (i) u has non-leaf children u' and u'' such that (u, u') is p-dominant and (u, u'') is d-dominant and (ii) there is no such node other than u in $T(u)$ satisfying (ii). In fact, if such u, u' and u'' are found, as shown at the beginning of Subsection 4.2 (right after Lemma 6), there exists a subgraph of $T(u)$ for which Case 4 holds. The path from u passing through (u, u') to a node x having only leaf children such that all edges on the path is p-dominant can be found in linear time by following p-dominant edges starting from u. Similarly, we can find in linear time a path from u passing through (u, u'') to a node x' having only leaf children such that all edges on the path is d-dominant.

Finding a subgraph for which Case 5 holds can be done similarly. We first need to find an internal node u such that $(parent(u), u)$ is pd-balanced and there is not any pd-balanced edge in T_u. We then find a path from u to a node x having only leaf children such that all edges on the path is p-dominant. As in Case 4, this can be done in linear time. We then compute the set of leaves $\hat{L}$ by applying Algorithm **Find_**$\hat{L}$ given in Section 4.3. It is easy to see that Algorithm **Find_**$\hat{L}$ runs in linear time.

We now claim that the number of vehicles prepared by our algorithm is at most $2\lceil \sum_{v \in V}(D^p(v) + D^d(v)) \rceil$. For every edge $e = (r, u)$ incident to the root r, the number of vehicles passing e in a forward direction is at most $2lb\_\#vehicles(e)$ because the algorithm schedules vehicles which pass the edge e so that the cost required to pass the edge e is at most twice the decrease of the lower bound $LB(e)$ for any of Cases 1 through 5. This proves the above claim. Thus, the number of rounds required by the algorithm is $O(\sum_{v \in V}(D^p(v) + D^d(v)))$.

Theorem 3 *The total time required for the proposed algorithm is* $O(M(\sum_{v \in V}(D^p(v) + D^d(v)) + M^2)$, *where* $M = |V| + (\sum_{v \in V}(D^p(v) + D^d(v))$.

Note that the above running time is polynomial in the output (i.e. the number of vehicles scheduled) and in the input size.

5 Conclusions

We have presented a 2-approximation algorithm for finding optimal tours to serve pickup and delivery demands located at nodes of a tree-shaped network. This approximation ratio was shown to be tight. For future research, it is interesting to extend the research to the cases for general networks and for the plane.

References

[1] S. Anily and G. Mosheiov, The traveling salesman problem with delivery and backhauls, *Operations Research Letters*, 16 (1994) 11-18.

[2] T. Asano, N. Katoh and K. Kawashima, A New Approximation Algorithm for the Capacitated Vehicle Routing Problems on a Tree, *Journal of Combinatorial Optimization*. Vol.5 No.2 213-231, June 2000. (Preliminary version was published in *Proc. of ISAAC'99*, LNCS 1741, Springer-Verlag, 317-326, 1999.)

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

[4] 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.

[5] 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.

[6] 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.

[7] M. Gendreau, G. Laporte and D. Vigo, Heuristics for the traveling salesman problem with pickup and delivery, *Computers & Operations Research*, 26 (1999) 699-714.

[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 Capacitated Vehicle Routing Problem on a Tree, Proc. of ISAAC'98, *Lecture Notes in Computer Science 1533, Springer-Verlag* 397-406, 1998.

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

[11] G. Mosheiov, The traveling salesman problem with pickup and delivery, *European Journal of Operational Research*, 79 (1994) 299-310.

[12] P. Toth and D. Vigo, *The Vehicle Routing Problem*, *SIAM*, 2001.

[13] M.W.P. Savelsbergh, The general pickup and delivery problem, *Transportation Science*, Vol.29, No.1 (1995), 17-29.