Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 1, No. 3, pp. 527–543, September 2005

SCHEDULING VEHICLES ON TREES

Yoshiyuki Karuno and Hiroshi Nagamochi

Dedicated to Professor T. Ibaraki on the Occasion of His 65th Birthday.

Abstract: The vehicle scheduling problem consists of a set of n jobs which are located at different vertices in a given graph. Each job is characterized by a release time, a handling time and a due date (or a deadline). There are m ($1 \leq m \leq n$) identical vehicles on the graph to process the jobs. The problem asks to find a routing schedule of the m vehicles that minimizes a given objective function. Graphs are restricted to paths or trees in some applications, and thus the problem on these graphs has been studied extensively. In this paper, we give a brief review of approximation algorithms to the problem on paths or trees obtained by the previous work. As a closely related topic, we also discuss the subtree cover problem where a given edge-weighted tree with n weighted vertices is partitioned into a number of subtrees so as to minimize a given objective. In this paper, we propose an $O(n^3 \log n)$ time 3-approximation algorithm to the problem of minimizing the number of subtrees, where the weight of each subtree must not exceed a specified capacity.

Key words: *combinatorial optimization, approximation algorithm, vehicle scheduling problem, subtree cover problem*

Mathematics Subject Classification: *90B10, 90C27, 90C59*

1 Introduction

The *vehicle scheduling problem* (VSP for short) consists of a set of n jobs (such as items to be picked up or facilities to be inspected) which are located at different vertices in a given graph. Each job is characterized by a release time, a handling time and a due date (or a deadline). For a job, its *time window* means the time interval between its release time and deadline. The handling times are given as *vertex-weights* of the graph. There are m ($1 \leq m \leq n$) identical vehicles on the graph to process the jobs. When a vehicle traverses an edge, it takes a travel time associated with the edge. The travel times are given as *edge-weights* of the graph. A job at each vertex must be processed by exactly one vehicle; a vehicle can pass through a vertex without processing the job at the vertex any number of times. In this paper, we assume that no interruption of processing is allowed. The problem asks to find a routing schedule of the m vehicles that minimizes a given objective function such as the maximum tour time of vehicles, the makespan (i.e., the maximum completion time of jobs), the maximum lateness from due dates, and so on. The VSP is an important topic encountered in a variety of industrial and service sector applications (e.g., see Desrosiers, Dumas, Solomon and Soumis [7]).

Graphs are restricted to paths or trees in some applications such as the delivery scheduling by ships on a shoreline [24] and by robots with elevator boarding function in a building.

Thus, the problem on these graphs has been studied extensively. In this paper, we give a brief review of approximation algorithms to the problem on paths or trees obtained by the previous work. By noting that the approximability of the bin-packing problem has been well studied (e.g., see Ausiello et al. [3]), which can be regarded as a special case of the VSP on trees where the graph is a star, investigation of the VSP on trees is an important issue to categorize approximation classes of related NP-hard problems. In this paper, when given graphs are restricted to trees (resp., paths), the VSP is denoted by VSP-TREE (resp., VSP-PATH). The VSP with $m = 1$ is particularly called the single-vehicle scheduling problem. The VSP-TREE (resp., VSP-PATH) with $m = 1$ is referred to as the 1-VSP-TREE (resp., 1-VSP-PATH).

Approximation algorithms to the routing problems (with no time window constraint) on trees such as the p-traveling salesmen problem (p-TSP) and the capacitated vehicle routing problem (CVRP) have also been studied (e.g., see Asano, Katoh and Kawashima [1], Averbakh and Berman [4, 5, 6], Labbé, Laporte and Mercure [17], Nagamochi and Okada [21]). In this paper, we do not give a comprehensive review of such related problems, but we discuss the *subtree cover problem* (SCP for short) as a fundamental problem for scheduling or routing vehicles on trees, where the SCP asks to partition a given edge-weighted tree with n weighted vertices into a number of subtrees so as to minimize a given objective function. For a given integer p (≥ 2), the *minmax* SCP asks to find a partition that consists of p subtrees and minimizes the maximum weight of the p subtrees, where the weight of each subtree is the sum of edge-weights and vertex-weights in the subtree. On the other hand, the *capacitated* SCP asks to find a partition that minimizes the number of subtrees, where the weight of each subtree must not exceed a specified capacity d (> 0). We denote by MM-SCP and CAPA-SCP the minmax and capacitated subtree cover problems, respectively. Recently, Nagamochi and Okada [20] showed that the MM-SCP is $(2 - 2/(p + 1))$-approximable in $O(p^2 n)$ time. In this paper, we consider the CAPA-SCP and propose an $O(n^3 \log n)$ time 3-approximation algorithm to the problem, exploiting a few properties of the MM-SCP provided in [20]. A relation between the VSP-TREE and MM-SCP was showed by Nagamochi and Okada [21], who observed that there exists a constant factor approximation algorithm to the VSP-TREE of minimizing the makespan, based on a $(2 - 2/(p + 1))$-approximation algorithm to the MM-SCP.

The remainder of this paper is organized as follows. In Section 2, we formulate the 1-VSP-TREE, VSP-TREE and CAPA-SCP to be discussed here. We review some approximation algorithms to the 1-VSP-TREE in Section 3, and to the VSP-TREE in Section 4. In Section 5, we discuss the SCP. After reviewing the recent results of the MM-SCP, we propose an $O(n^3 \log n)$ time 3-approximation algorithm to the CAPA-SCP. Finally, in Section 6, we give some concluding remarks.

2 Preliminaries

Let $T = (V, E)$ be a tree, where V and E are the vertex set and edge set of T, respectively. Let $n = |\, V \,|$. A vertex with degree 1 is called a *leaf* in a tree T, but the root with degree 1 in a rooted tree is not called a leaf. The set of leaves in a tree T is denoted by L. Let $b = |\, L \,|$. A connected subgraph T' of T is called a *subtree* of T, and we denote this by $T' \subseteq T$. The vertex set and edge set of T' are denoted by $V(T')$ and $E(T')$, respectively.

A tree may be called a path if it consists of exactly two vertices with degree 1, called *end vertices*, and $n - 2$ *interior vertices* with degree 2. A connected subgraph of a path G is called a *subpath* of G. In this paper, a path is referred to as an *end-rooted* path if it is rooted at an end of the path. On the other hand, a path is called an *interior-rooted* path if

it is rooted at an interior vertex.

For a subset $X \subseteq V$ of vertices, let $T\langle X\rangle$ denote the minimal subtree of T that contains X (where the leaves of $T\langle X\rangle$ will be vertices in X). In this paper, we say that $T\langle X\rangle$ is *induced* from T by X.

Let (T, w) denote a tree $T = (V, E)$ such that each edge $e \in E$ is weighted by a non-negative real $w(e)$. The tree may be denoted by (T, w, v_0) if a root $v_0 \in V$ is specified. The sum of edge-weights in a subtree $T' \subseteq T$ is denoted by $w(T')$.

Let (S, h) be a specified subset S of V such that each vertex $v \in S$ has a non-negative weight $h(v)$, where we may denote $h(v) = 0$ for a vertex $v \in V \setminus S$. The sum of vertex-weights in a subset $S' \subseteq S$ is denoted by $h(S')$.

In the VSP, an edge-weight $w(u, v)(= w(v, u))$ of edge $\{u, v\} \in E$ is interpreted as the travel time of a vehicle between two adjacent vertices u and v. We extend the notation of travel time to non-adjacent vertices. For two vertices u and v, the sum of weights in a path between u and v in T is denoted by $w(u, v)$. The vertex set V also represents the job set. Thus, (V, h) denotes a set of n jobs such that each job $v \in V$ has a handling time $h(v)$, i.e., job v requires a specific amount of time $h(v)$ to process and eventually complete it. No interruption of processing is allowed. The set (V, h) may be denoted by (V, r, h) if each job v has a release time $r(v)$, i.e., it becomes available for processing at time $r(v)$. For a tree $(T = (V, E), w)$ and a job set (V, r, h), we define the sum of travel times by

$$W = 2w(T) = 2\sum_{e \in E} w(e),$$

the sum of handling times by

$$H = h(V) = \sum_{v \in V} h(v),$$

and the maximum of release times by

$$r_{max} = \max\{r(v) \mid v \in V\}.$$

Besides, for a rooted tree (T, w, v_0), we define by

$$\tilde{w} = \max\{w(v_0, v) \mid v \in V\}$$

the travel time from the root v_0 to the farthest vertex in T.

In the SCP, a collection $\mathcal{S}$ of disjoint subsets $S_1, S_2, \ldots, S_k$ of $S \subseteq V$ is called a *partition* of S if their union is S, where some S_i may be empty. A collection $\mathcal{S}$ of S is called a *p-partition* of S if $|\mathcal{S}| = p$.

Now we are ready to describe the 1-VSP-TREE, VSP-TREE and CAPA-SCP. As mentioned in Section 1, the 1-VSP-TREE and VSP-TREE are possessed of a large number of variants for time constraints and objective functions. Hereafter, unless otherwise stated, the 1-VSP-TREE and VSP-TREE represent the following problems.

1-VSP-TREE (Tour time 1-VSP-TREE)

Input: An instance $I = (T = (V, E), w, v_0, V, r, h)$ which consists of a rooted tree (T, w, v_0) and a job set (V, r, h).

Feasible solution: A schedule π of a single vehicle initially situated at the root v_0, i.e., a processing ordering of n jobs by the vehicle.

Goal: Minimize the tour time of the vehicle $C_{tour}(\pi)$ (i.e., the time to return to the root v_0 after processing all jobs).

Comments: In the makespan 1-VSP-TREE, the initial location of the vehicle is also given by the root v_0, but the vehicle does not have to return to the initial location.

VSP-TREE (Makespan VSP-TREE)

Input: An instance $I = (T = (V, E), w, V, r, h, m)$ which consists of a tree (T, w), a job set (V, r, h), and m identical vehicles.

Feasible solution: A schedule π of the m vehicles, i.e., a set π of m processing orderings of jobs where each job belongs to exactly one among the m orderings.

Goal: Minimize the makespan $C_{max}(\pi)$ (i.e., the maximum completion time of all jobs).

Comments: Initial locations of the m vehicles are not given, and the vehicles do not have to return to their initial locations. In the maximum tour time VSP-TREE, initial locations of the vehicles are also chosen so as to minimize the maximum tour time, but each vehicle must return to the initial location.

CAPA-SCP

Input: An instance $I = (T = (V, E), w, V, h, d)$ which consists of an edge-weighted tree (T, w), a vertex-weighted set (V, h) and a positive real d.

Feasible solution: A partition $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of V such that

$$cost_I(\mathcal{S}) := \max\{w(T\langle S_i\rangle) + h(S_i) \mid S_i \in \mathcal{S}\} \leq d.$$

Goal: Minimize the number $k = \mid \mathcal{S} \mid$ of subtrees.

Comments: For any instance I, without loss of generality, we assume that $\max\{h(v) \mid v \in V\} \leq d$ (otherwise, there exists no feasible solution to the instance).

Table 1 summarizes the known results for the approximability of the 1-VSP-TREE and VSP-TREE.

3 Scheduling a Single Vehicle

In this section, we review approximation algorithms to the 1-VSP-TREE. The results can be applied to the 1-VSP-PATH in the same context since a path is a special case of trees. However, a dedicated algorithm to the 1-VSP-PATH may have a better approximation ratio than that to the 1-VSP-TREE. Thus, in this section, we also discuss such a dedicated algorithm to the 1-VSP-PATH.

Table 1: Approximation Ratios for the 1-VSP-TREE and VSP-TREE

Objective	Tour time		Makespan	
	PATH	TREE	PATH	TREE
1-VSP-	$\frac{5}{3}$ [9]	2 [13]	2 [13, 24]	2 [13]
			PTAS [2]	PTAS* [2]
Objective	Maximum tour time		Makespan	
	PATH	TREE	PATH	TREE
VSP-	$\left(3 - \frac{2}{m+1}\right)$ [21]	$\left(3 - \frac{2}{m+1}\right)$ [21]	2 [15]	$\left(5 - \frac{4}{m+1}\right)$ [21]
			PTAS† [16]	PTAS*,† [16]

PTAS: Polynomial Time Approximation Scheme.
$\star$ for a constant number b of leaves in a given tree.
$\dagger$ for a constant number m of vehicles.

3.1 On Trees

Nagamochi, Mochizuki and Ibaraki [18] proved the strong NP-hardness of the 1-VSP-TREE even if all handling times are zero.

First, we consider the following depth-first routing constraint to the 1-VSP-TREE. Once the single vehicle reaches a vertex v from its parent in a given tree T, it cannot return to the parent unless it has completed all jobs in the subtrees rooted at v. Under such a constraint, each edge $\{u, v\} \in E$ is traversed exactly twice (that is, one from u to v and another from v to u). We refer to a schedule under the depth-first routing constraint as a *depth-first schedule*.

The depth-first schedules possess the following property. The tour time of a schedule consists of travel times on edges, handling times on vertices and idle times due to release time constraints. The sum of handling times is constant in a schedule. Notice that the sum of travel times is also constant in a depth-first schedule. Thus, minimizing the tour time is equivalent to minimizing the sum of idle times under the depth-first routing constraint.

Karuno, Nagamochi and Ibaraki [13] observed that an optimal depth-first schedule (i.e., a depth-first schedule with the minimum tour time among all depth-first schedules) can be obtained in $O(n \log n)$ time. The outline of the algorithm is as follows. Assume that the vehicle reaches vertex v from its parent at time t and there exist subtrees rooted at v. For each subtree rooted at v, compute the total idle time incurred by the vehicle which starts from v at time t, visits all jobs in the subtree for processing (of course, in the depth-first manner) and returns to v. The algorithm makes the vehicle traverse the subtrees one by one in a non-decreasing order of the total idle times. Ascending T from leaves to v_0, the algorithm performs such a computation recursively.

Karuno, Nagamochi and Ibaraki [13] also showed in the same paper that the optimal depth-first schedule is a 2-approximate solution to the 1-VSP-TREE, and that this ratio is asymptotically tight. An immediate lower bound on the optimal tour time C^*_{tour} can be obtained by $C^*_{tour} \geq \max\{r_{max}, W + H\}$. It is not difficult to see that the tour time of any depth-first schedule π^{DF} satisfies $C_{tour}(\pi^{DF}) \leq r_{max} + W + H$, since the vehicle can

traverse the given tree for processing with no idle time after time r_{max}. This implies the approximation ratio of 2.

It is significant to notice that the minimum travel time is guaranteed if and only if the vehicle adopts a depth-first schedule. Since the minimization of total travel time is important also in practice (for example, to reduce the power battery consumption of the vehicle), the depth-first schedules are often required. In the 1-VSP-TREE, Nagamochi, Mochizuki and Ibaraki [19] showed that once an optimal depth-first schedule with respect to a specified initial vertex has been solved, the minimum tour times for all other initial vertices can be simultaneously computed in $O(n)$ time. Karuno, Nagamochi and Ibaraki [12] considered a different variant of 1-VSP-TREE where a job has its own due date, but all jobs are available at time 0. The objective is to minimize the maximum lateness from the due dates. They showed that an optimal depth-first schedule to the maximum lateness 1-VSP-TREE can be obtained in $O(n \log n)$ time.

In the 1-VSP-TREE of minimizing the makespan, the vehicle does not have to return to the root v_0 (or, no cost is incurred to return from the last vertex to the root). For the makespan 1-VSP-TREE, an immediate lower bound on the optimal value C_{max} is given by $C^*_{max} \geq \max\{r_{max}, W + H - \tilde{w}\}$, where $\tilde{w}$ denotes the travel time between the root v_0 and the farthest vertex v_{far} in T. Since a depth-first schedule such that its last job is the v_{far} can be obtained in $O(n)$ time, the makespan 1-VSP-TREE is also 2-approximable.

Recently, Augustine and Seiden [2] showed that the makespan 1-VSP-TREE with a constant number b of leaves admits a *polynomial time approximation scheme* (i.e., a family of algorithms $\{A_\varepsilon\}$ such that for any $\varepsilon > 0$, A_ε delivers a schedule with the makespan at most $(1 + \varepsilon)$ times the optimal). Let $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ be a permutation on $\{1, 2, \ldots, n\}$, where $\pi(i)$ denotes the ith job processed by the vehicle, and let $c_\pi(i)$ be the completion time of the ith job in π. For notational convenience, we define $\pi(0) = v_0$ and $c_\pi(0) = 0$, and let $\pi^{-1}(j)$ be the position of job j in π. A schedule π *eagerly* processes job j if for all i such that job j is located on the unique path from $\pi(i-1)$ to $\pi(i)$, either $\pi^{-1}(j) \leq i$ or $r(j) > c_\pi(i-1) + w(\pi(i-1), j)$ holds. If a π eagerly processes all jobs, the π is called *eager*. Augustine and Seiden [2] proved that there exists an optimal schedule among eager schedules to the makespan 1-VSP-TREE, and based on this, they showed that an optimal schedule can be obtained in polynomial time if the number of leaves in T and the number of distinct release times are constant. The polynomial time approximation scheme is derived from these facts, and the time complexity is bounded by a linear in n (but by an exponential in $1/\varepsilon$).

3.2 On Paths

The 1-VSP-PATH is described as follows.

1-VSP-PATH (Tour time 1-VSP-PATH)

Input: An instance $I = (G = (V, E), w, v_0, V, r, h)$ which consists of a path (G, w, v_0) and a job set (V, r, h).

Feasible solution: A schedule π of a single vehicle initially situated at v_0, i.e., a processing ordering of n jobs by the vehicle.

Goal: Minimize the tour time of the vehicle $C_{tour}(\pi)$ (i.e., the time to return to v_0 after processing all jobs).

The NP-hardness of the 1-VSP-PATH was proved by Tsitsiklis [25].

Since a path is a special case of trees where the degree of each vertex is at most two, an optimal depth-first schedule to the 1-VSP-PATH can be obtained in $O(n)$, and it is a 2-approximate solution [13].

Psaraftis, Solomon, Magnanti and Kim [24] showed that the makespan 1-VSP-PATH can be solved in $O(n^2)$ time if all handling times are zero. Gaur, Gupta and Krishnamurti [9] applied the $O(n^2)$ time algorithm to the 1-VSP-PATH. Let π_1 be the schedule obtained by the $O(n^2)$ time algorithm. They showed that the performance of π_1 to the 1-VSP-PATH is guaranteed as follows.

$$C_{tour}(\pi_1) \leq \left(1 + \frac{H}{W+H}\right) C^*_{tour}. \tag{1}$$

Thus, we can easily see that the π_1 is also a 2-approximate solution to the 1-VSP-PATH.

Karuno, Nagamochi and Ibaraki [14] proved that the 1-VSP-PATH is 3/2-approximable in $O(n)$ time if a given path is end-rooted. For a real t with $0 \leq t \leq r_{max}$, let $H(t)$ (resp., $H'(t)$) be the sum of handling times of all jobs $v \in V$ with $r(v) \geq t$ (resp., $r(v) > t$). They provided the following lower bound on C^*_{tour}.

$$C^*_{tour} \geq t + H(t) \tag{2}$$

for any t with $0 \leq t \leq r_{max}$. They also showed that there always exists a t^* such that $H'(t^*) \leq t^* \leq H(t^*)$. From these,

$$C^*_{tour} \geq 2t^* \geq 2H'(t^*) \tag{3}$$

is obtained. In addition, the following lower bounds are known.

$$C^*_{tour} \geq W + H, \quad \text{and} \tag{4}$$

$$C^*_{tour} \geq r(v) + h(v) + w(v, v_0) \quad \text{for any job } v \in V. \tag{5}$$

Assume that in a given path $G = (V, E)$, the v_0 is the left end of G and v_{n-1} the right end. The approximation algorithm, called *two-phase* algorithm, makes the vehicle traverse a given path as follows. In the forward phase, the vehicle travels from the left end v_0 to the right end v_{n-1}, processing all jobs whose release times are at most t^*. In the backward phase, the vehicle returns back from v_{n-1} to v_0, processing all remaining jobs.

The approximation ratio is derived as follows. Let π' be the schedule obtained by the two-phase algorithm. If the vehicle does not wait at any vertex in the backward phase, then $C_{tour}(\pi') \leq t^* + W + H$. This is because the vehicle can process all jobs whose release times are at most t^* with no waiting if it waits at the v_0 until time t^*. By eqs. (3) and (4), we have

$$C_{tour}(\pi') \leq t^* + W + H \leq t^* + C^*_{tour} \leq \frac{3}{2} C^*_{tour}.$$

On the other hand, if the vehicle waits at some vertex v_k in the backward phase, then for such a v_k that it is the nearest one to the v_0, we have $C_{tour}(\pi') = r(v_k) + h(v_k) + w(v_k, v_0) + H_k$, where H_k denotes the sum of handling times of all jobs that are processed after v_k in π'. Note that $H_k \leq H'(t^*)$ holds. Again by eq. (3) and by eq. (5), we obtain

$$C_{tour}(\pi') = r(v_k) + h(v_k) + w(v_k, v_0) + H_k \leq C^*_{tour} + H'(t^*) \leq \frac{3}{2} C^*_{tour}.$$

Therefore, the approximation ratio of the two-phase algorithm is 3/2.

Gaur, Gupta and Krishnamurti [9] modified the two-phase algorithm to the case of interior-rooted paths. In the modified two-phase algorithm, the vehicle first goes from the root v_0 to the closer end to the v_0. Once the vehicle reaches the closer end, the vehicle follows the two-phase algorithm. Finally, the vehicle returns back from the closer end to the v_0. Let π_2 be the schedule by the modified two-phase algorithm. Gaur, Gupta and Krishnamurti [9] proved that the modified two-phase algorithm has the following performance guarantee to the 1-VSP-PATH.

$$C_{tour}(\pi_2) \leq \left(\frac{3}{2} + \frac{1}{2} \cdot \frac{W}{W+H}\right) C^*_{tour}. \tag{6}$$

Gaur, Gupta and Krishnamurti [9] showed that the approximation ratio 5/3 to the 1-VSP-PATH can be derived from eqs. (1) and (6) if one computes two schedules π_1 and π_2 and then chooses the better. This is because

$$1 + \frac{H}{W+H} \leq \frac{5}{3} \quad \text{if } H \leq 2W, \quad \text{and}$$
$$\frac{3}{2} + \frac{1}{2} \cdot \frac{W}{W+H} \leq \frac{5}{3} \quad \text{if } H > 2W.$$

The polynomial time approximation scheme to the 1-VSP-TREE with a constant number of leaves proposed by Augustine and Seiden [2] can be applied to the makespan 1-VSP-PATH.

4 Scheduling Multiple Vehicles

The first constant factor approximation algorithm to the VSP-PATH was proposed by Karuno and Nagamochi [15]. Afterward, for a constant number m of vehicles, they developed a polynomial time approximation scheme to the VSP-PATH and extended it to the VSP-TREE with a constant number b of leaves in T [16]. In this section, we also start with the following VSP-PATH.

VSP-PATH (Makespan VSP-PATH)

Input: An instance $I = (G = (V, E), w, V, r, h, m)$ which consists of a path (G, w), a job set (V, r, h), and m identical vehicles.

Feasible solution: A schedule π of the m vehicles, i.e., a set π of m processing orderings of jobs where each job belongs to exactly one among the m orderings.

Goal: Minimize the makespan $C_{max}(\pi)$ (i.e., the maximum completion time of all jobs).

It should be noted that if all edge-weights in a given path (i.e., travel times of the vehicles) are zero, then the VSP-PATH is identical to the parallel machine scheduling problem, which asks to minimize the makespan under the release time constraint. According to the traditional notation for machine scheduling problems studied by Graham, Lawler, Lenstra and Rinnooy Kan [10], this machine scheduling problem is denoted by $P/r_j/C_{max}$.

When $m = 1$, problem $P/r_j/C_{max}$ becomes the single-machine scheduling problem denoted by $1/r_j/C_{max}$. It can be solved in $O(n \log n)$ time where jobs are scheduled in a non-decreasing order of release times. However, in the makespan 1-VSP-PATH, a slight

change in an order of processing jobs may affect the makespan more dramatically due to the travel times. In fact, Tsitsiklis [25] proved the NP-hardness of the makespan 1-VSP-PATH, which indicates that introducing travel times distinguishes the computational complexity of the makespan 1-VSP-PATH from that of $1/r_j/C_{max}$.

When $m \geq 2$, problem $P/r_j/C_{max}$ is NP-hard in the strong sense since it contains the 3-PARTITION as a special case, and the problem for any fixed $m \geq 2$ is even NP-hard since it contains the PARTITION (e.g., see Garey and Johnson [8]). Hall and Shmoys [11] observed that there exist a 2-approximation algorithm and a polynomial time approximation scheme for problem $P/r_j/C_{max}$ (but the running times are not available in their paper). The VSP-PATH is NP-hard in the strong sense for m arbitrary, since it can be viewed as a generalization of $P/r_j/C_{max}$. For a similar reason, the VSP-PATH is NP-hard even for any fixed $m \geq 2$. However, the VSP-PATH or the VSP-TREE has a more intractable situation. In an optimal schedule for an instance of the VSP-PATH with $m \geq 2$, some edges may not be traversed by any vehicle. Such an edge is called a *gap*. This makes difficult for us to derive a lower bound on the total travel time.

For a schedule to the VSP-PATH, we refer to a subpath of a given path which is traversed by a certain vehicle as its *zone*. A feasible schedule for m' vehicles ($m' \leq m$) is called a *zone schedule* if no two zones intersect and thus there are $m' - 1$ gaps. Moreover, a zone schedule is called a *1-way* zone schedule if any vehicle traverses its zone in one direction (i.e., from left to right or from right to left). On the other hand, a schedule is called *gapless* if each edge is traversed at least once by some vehicle.

The first constant factor approximation algorithm to the VSP-PATH with $m \geq 2$ has been obtained by Karuno and Nagamochi [15]. They first observed that there exists a 1-way zone schedule such that it is a 2-approximate solution to the problem of finding an optimal gapless schedule (i.e., the one with the minimum of the makespan among all schedules with no gaps). Such a 1-way zone schedule can be found in $O(n)$ time. For an optimal gapless schedule, we can obtain an immediate lower bound $(W/2 + H)/m$ on its makespan. Notice that a general schedule consists of several gapless schedules for subpaths on a given path. As stated above, for such a subpath, there exists a 1-way zone schedule such that it is a 2-approximate solution. Therefore, we need to take into account all possible configurations of gaps on the given path. Karuno and Nagamochi [15] proved that an optimal 1-way zone schedule can be found in $O(mn^2)$ time by a dynamic programming procedure, which implies that there exists a 1-way zone schedule that is a 2-approximate solution to the general case. By designing an algorithm for approximating an optimal 1-way zone schedule, Karuno and Nagamochi also presented a nearly linear time $(2 + \varepsilon)$-approximation algorithm to the VSP-PATH for any fixed $\varepsilon > 0$ [15].

When a fleet of vehicles follows a zone schedule, no two vehicles interfere each other on a given path. As such non-interference among the vehicles is important to control them safely, the zone schedules are often required in practice. Augustine and Seiden [2] extended their polynomial time approximation scheme for the makespan 1-VSP-TREE with a constant number of leaves to a polynomial time approximation scheme for the VSP-PATH of finding an optimal zone schedule.

Karuno and Nagamochi [16] developed a polynomial time approximation scheme to the VSP-PATH with a constant number m of vehicles. The approximation scheme is based on the approximation of the problem by rounding given release times, and on the fact that any schedule consists of several gapless schedules for subpaths on a given path. Rounding given release times leads to a problem instance with a constant number of distinct release times. The approximation scheme is a two-fold dynamic programming. One is for computing an optimal schedule to the problem with rounded release times, and the other for finding the

best schedule to the original problem by combining several gapless schedules over all choices of gaps on the path.

The algorithm can be extended to the VSP-TREE so that a polynomial time approximation scheme is obtained if m and b in a given tree are constant [16]. The polynomial time approximation scheme by Augustine and Seiden [2] can also be extended to the VSP-TREE of finding an optimal zone schedule if b is constant, but it may not be an approximation to the VSP-TREE of finding the optimal attained by general schedules.

Nagamochi and Okada [21] observed that there exists a $(5 - 4/(m+1))$-approximation algorithm to the VSP-TREE, which is based on a constant factor approximation algorithm to the MM-SCP. We give a detail of the relation between the VSP-TREE and MM-SCP in the next section.

5 Subtree Cover Problems

In this section, we propose a 3-approximation algorithm to the capacitated subtree cover problem CAPA-SCP. This algorithm exploits a few properties of the minmax subtree cover problem MM-SCP obtained in the previous work by Nagamochi and Okada [20, 21]. In addition, there is a certain relation between the MM-SCP and VSP-TREE [21]. So we review the results of the MM-SCP before considering the CAPA-SCP.

5.1 Previous Results

In this subsection, we describe the MM-SCP more generally.

MM-SCP

Input: An instance $I = (T = (V, E), w, S, h, p)$ which consists of an edge-weighted tree (T, w), a vertex-weighted subset (S, h) of V and an integer $p \in [2, n]$.

Feasible solution: A p-partition $\mathcal{S} = \{S_1, S_2, \ldots, S_p\}$ of S.

Goal: Minimize the cost $cost_I(\mathcal{S})$ of a partition $\mathcal{S}$ in I, where

$$cost_I(\mathcal{S}) := \max\{w(T\langle S_i\rangle) + h(S_i) \mid S_i \in \mathcal{S}\}.$$

For an instance $I = (T, w, S, h, p)$ of the MM-SCP, we denote the optimal value by $opt(I)$, and we say that a partition $\mathcal{S}$ of S *induces* edge-disjoint (resp., vertex-disjoint) subtrees if for any two S_i, $S_j \in \mathcal{S}$, subtrees $T\langle S_i\rangle$ and $T\langle S_j\rangle$ are edge-disjoint (resp., vertex-disjoint).

It is a simple observation that, for an instance $I = (T, w, S, h, p)$ of the MM-SCP, $opt(I) \geq \max\{(w(T) + h(S))/p, \max_{u \in S} h(u)\}$ holds provided that each edge is contained in some subtree $T\langle S_i\rangle$ for an optimal solution $\mathcal{S}^* = \{S_1, S_2, \ldots, S_p\}$. However, the inequality does not hold in general. So, for a tree (T, w), a subset (S, h) of V, and an integer $p \leq \mid S \mid$, Nagamochi and Okada [21] introduced a *valued subtree collection* of (T, w, S, h, p) as a set $\mathcal{T}$ of vertex-disjoint subtrees $T_1, T_2, \ldots, T_k \subseteq T$ such that $S \subseteq V(\mathcal{T})$ holds and a positive integer $p_{[T_i]}$ with $\sum_{T_i \in \mathcal{T}} p_{[T_i]} = p$ is associated with each T_i. They defined

$$\lambda(\mathcal{T}) = \max\left\{ \frac{w(T_i) + h(V(T_i))}{p_{[T_i]}} \mid T_i \in \mathcal{T} \right\}$$

and

$$\lambda^*(T, w, S, h, p) = \min\{\lambda(\mathcal{T}) \mid \text{all valued subtree collections } \mathcal{T} \text{ of } (T, w, S, h, p)\}.$$

The following results have been shown by Nagamochi and Okada [21].

Lemma 1 [21] *For an instance $I = (T, w, S, h, p)$ of the* MM-SCP*, there exists a p-partition $\mathcal{S}$ of S with*

$$cost_I(\mathcal{S}) \leq \max\left\{\left(2 - \frac{2}{p+1}\right) \cdot \frac{w(T)}{p},\ \max_{u \in S} h(u)\right\} \tag{7}$$

that induces edge-disjoint subtrees. Such an $\mathcal{S}$ can be obtained in $O(n)$ time.

Lemma 2 [21] *For an instance $I = (T, w, S, h, p)$ of the* MM-SCP*, $\lambda^*(T, w, S, h, p)$ is a lower bound on $opt(I)$.*

Lemma 3 [21] *For an instance $I = (T, w, S, h, p)$ of the* MM-SCP*, there exists a p-partition $\mathcal{S}$ of S with*

$$cost_I(\mathcal{S}) \leq \max\left\{\left(2 - \frac{2}{p+1}\right) \lambda^*(T, w, S, h, p),\ \max_{u \in S} h(u)\right\} \tag{8}$$

that induces edge-disjoint subtrees. Such an $\mathcal{S}$ can be obtained in $O((p-1)!n)$ time.

Afterward, Nagamochi and Okada [20] improved the time complexity in Lemma 3 when $S = V$ in the MM-SCP.

Lemma 4 [20] *For an instance $I = (T, w, V, h, p)$ of the* MM-SCP*, there exists a p-partition $\mathcal{S}$ of V with*

$$cost_I(\mathcal{S}) \leq \left(2 - \frac{2}{p+1}\right) opt(I) \tag{9}$$

that induces edge-disjoint subtrees. Such an $\mathcal{S}$ can be obtained in $O(p^2 n)$ time.

Nagamochi and Kawada [22] extended the underlying graphs from trees to cacti, and presented an $O(p^2 n)$ time $(4 - 4/(p+1))$-approximation algorithm to the MM-SCP on cacti.

When a rooted tree (T, w, v_0) is given and each subtree is required to contain the root v_0, the MM-SCP is called the *minmax rooted-subtree cover problem* (MM-RSCP for short). Similarly, the MM-RSCP is described as follows.

MM-RSCP

Input: An instance $I = (T = (V, E), w, v_0, S, h, p)$ which consists of a rooted edge-weighted tree (T, w, v_0), a vertex-weighted subset (S, h) of V and an integer $p \in [2, n]$.

Feasible solution: A p-partition $\mathcal{S} = \{S_1, S_2, \ldots, S_p\}$ of S.

Goal: Minimize the cost $cost_I(\mathcal{S})$ of a partition $\mathcal{S}$ in I, where

$$cost_I(\mathcal{S}) := \max\{w(T\langle S_i \cup \{v_0\}\rangle) + h(S_i) \mid S_i \in \mathcal{S}\}.$$

Table 2: Approximation Ratios for the MM-SCP and MM-RSCP

Underlying graphs	trees	cacti	general
MM-SCP	$\left(2-\frac{2}{p+1}\right)^{\dagger}$ [20]	$\left(4-\frac{4}{p+1}\right)^{\dagger}$ [22]	–
MM-RSCP	$(2+\varepsilon)$ [20]	$\left(3-\frac{2}{p+1}\right)^{\dagger}$ [23]	$\left(3-\frac{2}{p+1}\right)^{\dagger}$ [23]

† $S = V$ is assumed.

Nagamochi and Okada [20] proposed an $O(n \log\log_{1+\frac{\varepsilon}{2}} 3)$ time $(2+\varepsilon)$-approximation algorithm to the MM-RSCP. Nagamochi [23] considered the MM-RSCP with $S = V$ on general graphs, and presented a $(3-2/(p+1))$-approximation algorithm. Table 2 summarizes these approximation ratios for the MM-SCP and MM-RSCP.

Nagamochi and Okada [21] observed that by ignoring release times r the VSP-TREE can be regarded as the MM-SCP. From this observation, we obtain the following relation between the MM-SCP and VSP-TREE.

Theorem 1 *Assume that there exists an $O(f(n))$ time ρ-approximation algorithm to the* MM-SCP. *Then a $(1+2\rho)$-approximate solution to the* VSP-TREE *can be obtained in $O(f(n))$ time. If each vehicle is required to return to the initial location (i.e., the maximum tour time* VSP-TREE*), a $(1+\rho)$-approximate solution can be obtained in $O(f(n))$ time.*

Proof. For an instance $I = (T, w, V, r, h, m)$ of the VSP-TREE, let $I' = (T, w, V, h, m)$ be the instance of the MM-SCP converted from I. By assumption, we can find in $O(f(n))$ time an m-partition $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of V such that $cost_{I'}(\mathcal{S}) \leq \rho \cdot opt(I')$. For each $S_i \in \mathcal{S}$, the ith vehicle can process all jobs in S_i along the subtree $T\langle S_i\rangle$ until time $r_{max} + 2w(T\langle S_i\rangle) + h(S_i) \leq r_{max} + 2cost_{I'}(\mathcal{S})$ at latest. Since $\max\{r_{max},\ opt(I')\}$ is a lower bound on C^*_{max}, the schedule along these subtrees is a $(1+2\rho)$-approximate solution.

Next, we consider the maximum tour time VSP-TREE. Then each edge must be traversed even number of times. let $I'' = (T, 2w, V, h, m)$ be the instance of the MM-SCP converted from I by doubling all the edge-weights in T. By assumption, we can find an m-partition $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of V to I'' with $cost_{I''}(\mathcal{S}) \leq \rho \cdot opt(I'')$. For each $S_i \in \mathcal{S}$, the ith vehicle can process all jobs in S_i along the subtree $T\langle S_i\rangle$ until time $r_{max} + 2w(T\langle S_i\rangle) + h(S_i) \leq r_{max} + cost_{I''}(\mathcal{S})$ at latest. This gives a $(1+\rho)$-approximate solution. □

From this and Lemma 4, we obtain the following result.

Corollary 1 *A $(5-4/(m+1))$-approximate solution to the* VSP-TREE *can be obtained in $O(m^2 n)$ time. If each vehicle is required to return to the initial location, then a $(3-2/(m+1))$-approximate solution can be obtained in $O(m^2 n)$ time.*

5.2 A New Algorithm

In this subsection, we present an approximation algorithm to the CAPA-SCP.

Algorithm APPROX

Input: An instance $I = (T, w, V, h, d)$ of the CAPA-SCP, where $\max\{h(v) \mid v \in V\} \leq d$ is assumed.

Output: A $\tilde{p}$-partition $\tilde{\mathcal{S}} = \{S_1^1, S_1^2, S_1^3, \ldots, S_q^1, S_q^2, S_q^3\}$ of V such that $cost_I(\tilde{\mathcal{S}}) \leq d$.

STEP 1. If $w(T) + h(V) \leq 2d$, then $q := 1$, $\mathcal{S}_{[1]} := \{S_1\} := \{V\}$ and go to STEP 3. Otherwise go to STEP 2.

STEP 2. Find a $p \in [2, n]$ such that

$$cost_{I_{[p]}}(\mathcal{S}_{[p]}) \leq 2d < cost_{I_{[p-1]}}(\mathcal{S}_{[p-1]}) \tag{10}$$

holds for a p-partition $\mathcal{S}_{[p]}$ of V to $I_{[p]}$ and a $(p-1)$-partition $\mathcal{S}_{[p-1]}$ of V to $I_{[p-1]}$ obtained by the $O(p^2 n)$ time algorithm in Lemma 4, where $I_{[p]} = (T, w, V, h, p)$ and $I_{[p-1]} = (T, w, V, h, p-1)$ are instances of the MM-SCP constructed from I. Let q denote such a p.

STEP 3. Based on the q-partition $\mathcal{S}_{[q]} = \{S_1, S_2, \ldots, S_q\}$ of V obtained in STEP 1 or STEP 2, generate q instances of the MM-SCP, $I_i = (T\langle S_i\rangle, w, S_i, h, 3)$, $i = 1, 2, \ldots, q$. For each instance I_i, find a 3-partition $\mathcal{S}_i = \{S_i^1, S_i^2, S_i^3\}$ of S_i by the $O(n)$ time algorithm in Lemma 1.

STEP 4. Output the number of subtrees $\tilde{p} := 3q$ and the $\tilde{p}$-partition $\tilde{\mathcal{S}} = \{S_1^1, S_1^2, S_1^3, \ldots, S_q^1, S_q^2, S_q^3\}$ of V.

We should remark that

$$cost_{I_{[n]}}(\mathcal{S}_{[n]}) = \max\{h(v) \mid v \in V\} \leq d$$

holds by assumption on I, and

$$2d < w(T) + h(V) = cost_{I_{[1]}}(\mathcal{S}_{[1]})$$

holds in STEP 2 of APPROX. Hence there exists such a q (≥ 2) that satisfies eq. (10) in STEP 2.

For algorithm APPROX, the following lemmas hold.

Lemma 5 *For an instance $I = (T, w, V, h, d)$ of the* CAPA-SCP*, let p^* be the minimum number of subtrees, and q the number of subtrees obtained by* STEP 1 *or* STEP 2 *in* APPROX*. Then the q is a lower bound on p^*, i.e., it holds that*

$$q \leq p^*.$$

Proof. If $q = 1$, then the lemma obviously holds. For $q \geq 2$, by eq. (10), it holds that $cost_{I_{[q-1]}}(\mathcal{S}_{[q-1]}) > 2d$, where $\mathcal{S}_{[q-1]}$ is a $(q-1)$-partition of V obtained by the algorithm in Lemma 4. By the same lemma and the above inequality, we have

$$opt(I_{[q-1]}) > \frac{cost_{I_{[q-1]}}(\mathcal{S}_{[q-1]})}{2} \ (> d).$$

By the optimality of p^*, $p^* - 1$ is the largest integer such that $opt(I_{[p^*-1]}) > d$. Thus, we obtain $q - 1 \leq p^* - 1$, which completes the proof. □

Lemma 6 *For an instance $I = (T, w, V, h, d)$ of the* CAPA-SCP *with* $\max\{h(v) \mid v \in V\} \leq d$ *and* $w(T) + h(V) \leq 2d$, *there exists a 3-partition $\mathcal{S}$ of V with*

$$cost_I(\mathcal{S}) \leq d$$

that induces edge-disjoint subtrees. Such a partition can be obtained in $O(n)$ time.

Proof. In Lemma 1, consider the case of $p = 3$. Thus, there exists a 3-partition $\mathcal{S}$ of V such that

$$cost_I(\mathcal{S}) \leq \left(2 - \frac{2}{3+1}\right) \cdot \frac{w(T)}{3} \leq \frac{3}{2} \cdot \frac{2d}{3} = d.$$

The time complexity also follows Lemma 1. □

From Lemmas 5 and 6, we obtain an approximation ratio of APPROX as follows.

Theorem 2 *For an instance $I = (T, w, V, h, d)$ of the* CAPA-SCP, *let p^* be the minimum number of subtrees. Then there exists a partition $\mathcal{S} = \{S_1, S_2, \ldots, S_{\tilde{p}}\}$ with*

$$\tilde{p} \leq 3 \cdot p^*,$$

that induces edge-disjoint subtrees. Such a partition can be obtained in $O((p^*)^2 n \log p^*)$ time.

Proof. The approximation ratio of 3 can be derived immediately from Lemma 5 and Lemma 6.

The time complexity is evaluated as follows. STEP 1 clearly requires $O(n)$ time. In STEP 2, we obtain a desired q as follows. First find the smallest $\ell = \ell^*$ such that for $p_\ell = 2^\ell$ $(\ell = 1, 2, \ldots)$, $cost_{I_{[p_\ell]}}(\mathcal{S}_{[p_\ell]}) \leq 2d$ holds. By assumption of ℓ^*, a desired q satisfies that

$$2^{\ell^*-1} < q \leq 2^{\ell^*} \leq 2p^*.$$

By calling the $O(p^2 n)$ time algorithm in Lemma 4 for each $p_\ell = 2^\ell$ $(\ell = 1, 2, \ldots, \ell^*)$, the ℓ^* can be found in $O(\sum_{\ell=1}^{\ell^*}((2^\ell)^2 n)) = O((2^{\ell^*})^2 n) = O((p^*)^2 n)$ time. Then a binary search is used to find q from the interval $(2^{\ell^*-1},\ 2^{\ell^*}]$. It contains $O(p^*)$ integers. So the binary search requires $O(\log p^*) \times O((p^*)^2 n) = O((p^*)^2 n \log p^*)$ time to find the q from the interval. Thus, STEP 2 requires $O((p^*)^2 n \log p^*)$ time. In STEP 3, we call the $O(n)$ time algorithm in Lemma 1 $O(q)$ $(= O(p^*))$ times. Thus, STEP 3 requires $O(p^* n)$ time. It is obvious that STEP 4 also requires $O(p^* n)$ time. Therefore, we conclude that the time complexity of APPROX is $O((p^*)^2 n \log p^*)$ time. □

6 Concluding Remarks

In this paper, we gave a brief review of approximation algorithms to the 1-VSP-TREE and VSP-TREE obtained by the previous work. As a related topic, we also discussed the SCP. After reviewing the recent results of the MM-SCP, we proposed an $O((p^*)^2 n \log p^*)$ time 3-approximation algorithm to the CAPA-SCP, where p^* denotes the optimum of the number of subtrees. As provided in this paper, there exists a relation between the VSP-TREE and MM-SCP with respect to the approximability of these problems. Therefore, it becomes more important to study approximation algorithms to the SCP as well as these to the VSP. It is left for the future research to develop algorithms that achieve a better performance or work for more general underlying graphs.

Acknowledgment

The authors would like to express their gratitude to Emeritus Professor Toshihide Ibaraki of Kyoto University for his guidance and encouragement. The authors also wish to thank two anonymous referees for their helpful comments.

This research was partially supported by a Scientific Grant in Aid from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] T. Asano, N. Katoh and K. Kawashima, A new approximation algorithm for the capacitated vehicle routing problem on a tree, *J. Comb. Optim.* 5 (2001) 213–231.

[2] J.E. Augustine and S.S. Seiden, Linear time approximation schemes for vehicle scheduling, in *Lecture Notes in Comput. Sci. 2368: Algorithm Theory – SWAT 2002*, M. Penttonen and E. Meineche Schmidt (eds.), Springer, Berlin, 2002, pp. 30–39.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer, Berlin, 1999.

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

[5] I. Averbakh and O. Berman, A heuristic with worst-case analysis for minmax routing of two traveling salesmen on a tree, *Discrete Appl. Math.* 68 (1996) 17–32.

[6] I. Averbakh and O. Berman, $(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective, *Discrete Appl. Math.* 75 (1997) 201–216.

[7] J. Desrosiers, Y. Dumas, M.M. Solomon and F. Soumis, Time constrained routing and scheduling, in *Handbooks in Operations Research and Management Science Volume 8: Network Routing*, M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser (eds.), North-Holland, Amsterdam, 1995, pp. 35–139.

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

[9] D.R. Gaur, A. Gupta and R. Krishnamurti, A 5/3-approximation algorithm for scheduling vehicles on a path with release and handling times, *Inform. Process. Lett.* 86 (2003) 87–91.

[10] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, *Ann. Discrete Math.* 5 (1979) 287–326.

[11] L.A. Hall and D.B. Shmoys, Approximation schemes for constrained scheduling problems, *Proc. 30th IEEE Symp. on Foundations of Computer Science*, 1989, pp. 134–139.

[12] Y. Karuno, H. Nagamochi and T. Ibaraki, Vehicle scheduling on a tree to minimize maximum lateness, *J. Oper. Res. Soc. Japan* 39 (1996) 345–355.

[13] Y. Karuno, H. Nagamochi and T. Ibaraki, Vehicle scheduling on a tree with release and handling times, *Ann. Oper. Res.* 69 (1997) 193–207.

[14] Y. Karuno and H. Nagamochi and T. Ibaraki, Better approximation ratios for the single-vehicle scheduling problems on line-shaped networks, *Networks* 39 (2002) 203–209.

[15] Y. Karuno and H. Nagamochi, 2-Approximation algorithms for the multi-vehicle scheduling problem on a path with release and handling times, *Discrete Appl. Math.* 129 (2003) 433–447.

[16] Y. Karuno and H. Nagamochi, An approximability result of the multi-vehicle scheduling problem on a path with release and handling times, *Theoret. Comput. Sci.* 312 (2004) 267–280.

[17] M. Labbé, G. Laporte and H. Mercure, Capacitated vehicle routing on trees, *Oper. Res.* 39 (1991) 616–622.

[18] H. Nagamochi, K. Mochizuki and T. Ibaraki, Complexity of the single vehicle scheduling problem on graphs, *Inform. Syst. Oper. Res.* 35 (1997) 256–276.

[19] H. Nagamochi, K. Mochizuki and T. Ibaraki, Solving the single-vehicle scheduling problems for all home locations under depth-first routing on a tree, *Inst. Electron. Inform. Comm. Eng. Trans. Fundamentals* E84-A (2001) 1135–1143.

[20] H. Nagamochi and K. Okada, Polynomial time 2-approximation algorithms for the minmax subtree cover problem, in *Lecture Notes in Comput. Sci. 2906: Algorithms and Computation – ISAAC 2003*, T. Ibaraki, N. Katoh and H. Ono (eds.), Springer, Berlin, 2003, pp. 138–147.

[21] H. Nagamochi and K. Okada, A faster 2-approximation algorithm for the minmax p-traveling salesmen problem on a tree, *Discrete Appl. Math.* 140 (2004) 103–114.

[22] H. Nagamochi and T. Kawada, Approximating the minmax subtree cover problem in a cactus, in *Lecture Notes in Comput. Sci. 3341: Algorithms and Computation – ISAAC 2004*, R. Fleischer and G. Trippen (eds.), Springer, Berlin, 2004, pp. 705–716.

[23] H. Nagamochi, Approximating the minmax rooted-subtree cover problem, *Inst. Electron. Inform. Comm. Eng. Trans. Fundamentals* E88-A (2005) 1335–1338.

[24] H. Psaraftis, M. Solomon, T. Magnanti and T. Kim, Routing and scheduling on a shoreline with release times, *Management Sci.* 36 (1990) 212–223.

[25] J.N. Tsitsiklis, Special cases of traveling salesman and repairman problems with time windows, *Networks* 22 (1992) 263–282.

Manuscript received 31 August 2004
revised 26 January 2005
accepted for publication 28 January 2005

Yoshiyuki Karuno
Department of Mechanical and System Engineering, Faculty of Engineering and Design,
Kyoto Institute of Technology, Sakyo, Kyoto-city, Kyoto 606-8585, Japan
E-mail address: karuno@kit.ac.jp

Hiroshi Nagamochi
Department of Applied Mathematics and Physics, Graduate School of Informatics,
Kyoto University, Sakyo, Kyoto-city, Kyoto 606-8501, Japan
E-mail address: nag@amp.i.kyoto-u.ac.jp