Journal of the Operations Research
Society of Japan
Vol. 39, No. 3, September 1996

VEHICLE SCHEDULING ON A TREE TO MINIMIZE MAXIMUM LATENESS

Yoshiyuki Karuno
Kyoto Institute of Technology

Hiroshi Nagamochi   Toshihide Ibaraki
Kyoto University

(Received October 5, 1994; Revised March 8, 1995)

Abstract In this paper we deal with a single-vehicle scheduling problem on a tree-shaped road network. Let $T = (V, E)$ be a tree, where V is a set of n vertices and E is a set of edges. A task is located at each vertex $v \in V$, which is also denoted as v. Each task v has its due date $d(v)$ and processing time $p(v)$. The travel times $w(u, v)$ and $w(v, u)$ are associated with each edge $(u, v) \in E$. The vehicle starts from an initial vertex $v_0 \in V$, visits all tasks $v \in V$ for their processing (no preemption is allowed) and returns to v_0. The problem asks to find a routing schedule of the vehicle that minimizes the maximum lateness from the due dates of tasks. This problem is called TREE-VSP($L_{\max}$). We prove that TREE-VSP($L_{\max}$) is strongly *NP*-hard in general, but show that it can be solved in polynomial time if only depth-first routing is allowed for the vehicle.

1. Introduction

In this paper we consider a single-vehicle scheduling problem, in which the vehicle is assumed to process all tasks at different locations by following a tree-shaped road network that connects such locations (represented as vertices). Each task has its own due date and processing time. The problem asks to find a routing schedule of the vehicle (in other words, a processing sequence of tasks) that minimizes the maximum lateness, denoted as L_{max}, from the due dates of tasks. We call this problem TREE-VSP(L_{max}) (Vehicle Scheduling Problem on a Tree).

The problems of routing and scheduling are found in various applications such as material handling systems and computer communication networks. Recently, AGVs (automated guided vehicles) and handling robots are often used not only in manufacturing systems, but also in offices and hospitals, in order to reduce the material handling efforts. The tree-shaped network can be typically found in buildings with simple structures of corridors (each floor corresponds to a subtree and each room a leaf vertex).

The companion paper [8] considered a similar vehicle scheduling problem on a tree, in which each task has its release time and processing time, and the objective is to find a routing schedule of the vehicle that minimizes the completion time, denoted as C. This problem is denoted as TREE-VSP(C) in this paper. It was shown in [8] that TREE-VSP(C) is *NP*-hard even if all processing times are zero, but it is polynomially solvable if depth-first routing constraint is imposed. Moreover, Nagamochi et al. [11] proved that TREE-VSP(C) is strongly *NP*-hard. If the road network is given by a straight-line (i.e., a special case of tree) and all processing times are zero, it was already shown in Psaraftis et al. [12] that TREE-VSP(C) has a polynomial time algorithm. However, Tsitsiklis [15] showed that the complexity of TREE-VSP(C) on a straight-line is *NP*-hard if each task has positive processing time. Of course, the problem is *NP*-hard if the network is general graph, since it contains the Traveling Salesman Problem (TSP) as a special case. Another similar problem called the Delivery Man Problem (DMP) has also been studied (e.g., see Minieka [10]), which contains neither release times nor due dates, and minimizes total weighted waiting time (i.e., the weighted sum of completion times). In particular, Averbakh and Berman [2] showed that there is a polynomial time algorithm if the underlying graph is a tree and if depth-first routing constraint is imposed.

Our problem TREE-VSP(L_{max}) is mathematically defined as follows. Let $T = (V, E)$ be a tree that represents the road network, where V is a set of n vertices and E is a set of edges. We assume that the initial location of the vehicle (at time $t = 0$) is a particular vertex v_0 (which can be viewed as the root of T). The travel time of the vehicle is $w(u, v) \geq 0$ to traverse $(u, v) \in E$ in this direction, and is $w(v, u) \geq 0$ to traverse it in the opposite direction. There is a task at each vertex $v \in V$, which must be processed by the vehicle (such as picking up an item). The task at vertex v is also denoted as v. Each task v has its due date $d(v)$ and processing time $p(v)$. That is, the vehicle needs $p(v)$ time units for processing task v (no preemption is allowed), and it is desirable to complete task v by its due date $t = d(v)$. The vehicle at a vertex v is allowed to move to other vertices without processing task v if necessary (in this case, the vehicle has to come back to vertex v later to process task v). A routing schedule of the vehicle is completely specified by a sequence $\pi = (v_{j_1}, v_{j_2}, ..., v_{j_n})$ of tasks to be processed, i.e., the vehicle first moves to vertex v_{j_1} (which may possibly be v_0) along the unique path from v_0 to v_{j_1} in T, taking the travel time of the length of the path, and processes task v_{j_1}, then it moves to v_{j_2} along the unique path from v_{j_1} to v_{j_2} in T, and processes v_{j_2}, and so on, until it returns to v_0 after completing v_{j_n}. The lateness of task v_{j_i} in π is defined by

$$L(v_{j_i}) = C(v_{j_i}) - d(v_{j_i}), \tag{1}$$

where $C(v_{j_i})$ denotes the completion time of processing task v_{j_i}. Here we can assume without loss of generality that the vehicle does not wait at any vertex and on any edge. Thus, let $l(u, v)$ denote the travel time of the unique path from u to v, then the completion time of task v_{j_i} in π can be given by

$$C(v_{j_i}) = \sum_{k=1}^{i}\{l(v_{j_{k-1}}, v_{j_k}) + p(v_{j_k})\}, \tag{2}$$

where $v_{j_0} = v_0$ for notational convenience. Our objective is to find an optimal schedule π^* that minimizes the maximum lateness, i.e.,

$$L_{max}(\pi) = \max_{1\leq i\leq n} L(v_{j_i}). \tag{3}$$

The problem TREE-VSP(L_{max}) defined in this paper is different from TREE-VSP(C) discussed in [8]. However, we prove that TREE-VSP(L_{max}) is also strongly *NP*-hard. Then, we show that it can be solved in polynomial time if only depth-first routing is allowed for the vehicle.

2. NP-hardness of TREE-VSP(L_{max})

In this section, we show that TREE-VSP(L_{max}) is *NP*-hard by a reduction from TREE-VSP(C), which is known to be strongly *NP*-hard. Note that TREE-VSP(C) is strongly *NP*-hard even if processing times are all zero (see Nagamochi et al. [11]). The decision problem of TREE-VSP(C) is represented as follows:

TREE-VSP(C)
INSTANCE: A tree $T = (V, E)$, where $V = \{v_0, v_1, ..., v_{n-1}\}$ is a set of n vertices (n

tasks) and E is a set of edges, release times $r(v_i)$ for $v_i \in V$, travel times $w(v_i, v_j)(\geq 0)$ for $(v_i, v_j) \in E$, and a positive integer K. All processing times are assumed to be zero, and travel times are assumed to satisfy $w(v_i, v_j) = w(v_j, v_i)$ for all edges $(v_i, v_j) \in E$.
QUESTION: Is there a schedule of the vehicle $\pi = (v_{j1}, v_{j2}, ..., v_{jn})$ with $C(\pi) \leq K$ such that the start time of processing task v_i is not earlier than $r(v_i)$ for each $v_i \in V$?

In the above TREE-VSP(C), the completion time of each task v_{j_i} and the time of returning to v_0 after completing all tasks, by a schedule $\pi = (v_{j1}, v_{j2}, ..., v_{j_n})$, are

$$C(v_{j_i}) = \max\{\max_{1\leq h\leq i}\{r(v_{j_h}) + \sum_{k=h+1}^{i} l(v_{j_{k-1}}, v_{j_k})\},\ \sum_{k=1}^{i} l(v_{j_{k-1}}, v_{j_k})\},\ i = 1, 2, ..., n, \quad (4)$$

$$C(\pi) = \max\{\max_{1\leq h\leq n}\{r(v_{j_h}) + \sum_{k=h+1}^{n} l(v_{j_{k-1}}, v_{j_k})\},\ \sum_{k=1}^{n} l(v_{j_{k-1}}, v_{j_k})\} + l(v_{j_n}, v_0), \quad (5)$$

where $v_{j_0} = v_0$.

Theorem 1. TREE-VSP(L_{max}) is strongly *NP*-hard.
Proof: Given the above instance of TREE-VSP(C), we transform it into the following instance of TREE-VSP(L_{max}): The road network tree is given by the same $T = (V, E)$, where $v_0 \in V$ is the initial vertex of the vehicle, and has the same travel times $w(v_i, v_j)$ for $(v_i, v_j) \in E$. Also the processing times $p(v_i)$ are all zero. The due dates are given by

$$d(v_i) = K - \max\{r(v_i),\ l(v_0, v_i)\}, \qquad i = 0, 1, ..., n-1. \quad (6)$$

We now show that, for this instance, there is an optimal schedule $\pi^* = (v^*_{j1}, v^*_{j2}, ..., v^*_{j_n})$ with $L_{max}(\pi^*) \leq 0$ if and only if the instance of TREE-VSP(C) has a solution. (That is, TREE-VSP(C) is reduced to TREE-VSP(L_{max}), and hence TREE-VSP(L_{max}) is strongly *NP*-hard.)

(I) The instance of TREE-VSP(C) has a solution:
Let $\pi = (v_{j1}, v_{j2}, ..., v_{j_n})$ be the solution to TREE-VSP(C), and let $\pi^R = (v_{j_n}, v_{j_{n-1}}, ..., v_{j1})$ be its reversal. If we apply π^R to the instance of TREE-VSP(L_{max}), then the completion time of task v_{j_i} satisfies, for $i = 1, 2, ..., n$,

$$\begin{aligned} C(v_{j_i}) &= l(v_0, v_{j_n}) + \sum_{k=i+1}^{n} l(v_{j_{k-1}}, v_{j_k}) \\ &\leq C(\pi) - \max\{r(v_{j_i}),\ \sum_{k=1}^{i} l(v_{j_{k-1}}, v_{j_k})\} \\ &\leq K - \max\{r(v_{j_i}),\ l(v_0, v_{j_i})\} \\ &= d(v_{j_i}), \end{aligned}$$

where $v_{j_0} = v_0$ and the second line follows from (5). Thus, in this case, there exists an optimal schedule π^* such that $L_{max}(\pi^*) \leq 0$.

(II) The instance of TREE-VSP(C) does not have a solution:
Assume that there exists a schedule $\pi^* = (v^*_{j1}, v^*_{j2}, ..., v^*_{j_n})$ such that $L_{max}(\pi^*) \leq 0$. Then $C(v^*_{j_i}) \leq d(v^*_{j_i})$ for all i, and we obtain by (6) that

$$\sum_{k=1}^{i} l(v^*_{j_{k-1}}, v^*_{j_k}) \leq K - \max\{r(v^*_{j_i}),\ l(v_0, v^*_{j_i})\}, \qquad i = 1, 2, ..., n, \quad (7)$$

where $v^*_{j_0} = v_0$. Now consider the reversed schedule $\pi^{*R} = (v^*_{j_n}, v^*_{j_{n-1}}, ..., v^*_{j_1})$ of π^*, and apply it to TREE-VSP(C). Then, by using (5) and (7),

$$C(\pi^{*R}) = \max\{\max_{1\le h\le n}\{r(v^*_{j_h}) + \sum_{k=1}^{h} l(v^*_{j_{k-1}}, v^*_{j_k})\},\ l(v_0, v^*_{j_n}) + \sum_{k=1}^{n} l(v^*_{j_{k-1}}, v^*_{j_k})\} \le K.$$

This is a contradiction, and hence $L_{max}(\pi^*) > 0$. □

3. A Solvable Case

In this section, we introduce the depth-first routing constraint: once the vehicle reaches a vertex v from its parent in T, it cannot return to the parent unless it has completed all tasks in the subtree rooted at vertex v. Thus each edge $(u, v) \in E$ is traversed exactly two times (that is, one from u to v and another from v to u), in order to process all tasks.

To describe a schedule under depth-first routing constraint, we define some notations. Let $v_{q,i}$ denote the i-th child of vertex $v_q \in V$, $i = 1, 2, ..., k_q$, where k_q is the number of v_q's children, and $T(v_{q,i})$ the subtree of T rooted at $v_{q,i}$ (see Figure.1). Note that $k_q = 0$ if v_q is a leaf vertex. For convenience, we denote v_q by $v_{q,0}$, and the graph consisting only of $v_{q,0}$ (and no edge) by $T(v_{q,0})$. A schedule $\pi = \pi(v_0)$ under depth-first routing constraint is recursively defined by

$$\pi(v_q) = (\pi(v_{q,\psi(0)}), \pi(v_{q,\psi(1)}), ..., \pi(v_{q,\psi(k_q)})) \quad \text{for } v_q \in V - V_{\text{leaf}}, \tag{8}$$

where $\psi = (\psi(0), \psi(1), ..., \psi(k_q))$ is a permutation on $\{0, 1, ..., k_q\}$, $\pi(v_{q,i}) = (v_{q,i})$ if $T(v_{q,i})$ consists only of one vertex (i.e., $i = 0$ or $T(v_{q,i})$ is a leaf vertex), and V_{leaf} is the set of leaf vertices in V.

In Figure.2, for example, we consider that $v_{01} = v_1$, $v_{02} = v_2$, $v_{03} = v_3$, $v_{11} = v_4$, $v_{12} = v_5$, $v_{31} = v_6$, $v_{32} = v_7$ and $v_{i,0} = v_i$ for all i. A schedule π defined by

$$\begin{aligned}\pi(v_0) &= (\pi(v_{01}), \pi(v_{00}), \pi(v_{03}), \pi(v_{02})),\\ \pi(v_{01}) &= (\pi(v_{10}), \pi(v_{11}), \pi(v_{12})),\\ \pi(v_{03}) &= (\pi(v_{32}), \pi(v_{31}), \pi(v_{30}))\end{aligned}$$

gives the sequence of tasks, $\pi = (v_1, v_4, v_5, v_0, v_7, v_6, v_3, v_2)$.

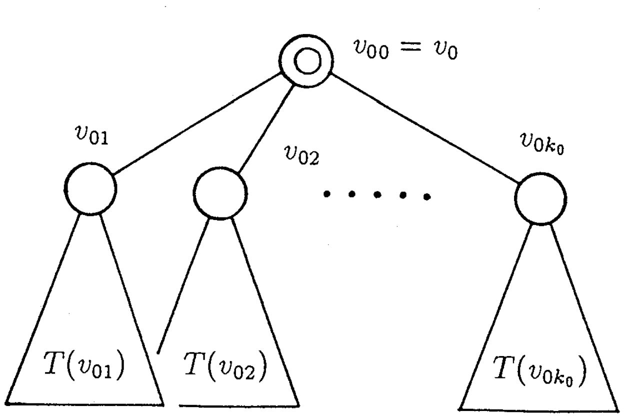

Figure.1 The subtrees rooted at v_0.

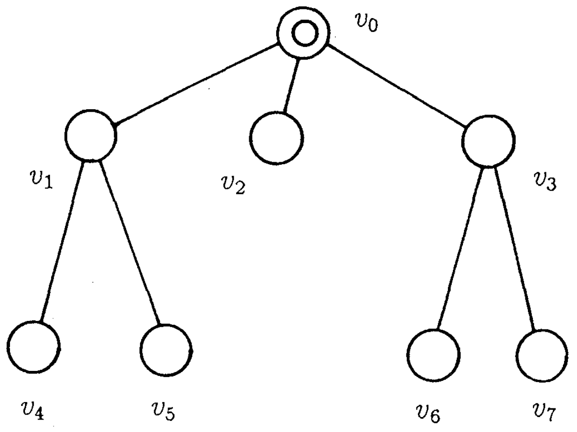

Figure.2 An example tree.

3.1 Basic properties

Let $L_t(v_{q,i})$ denote the maximal lateness, which is incurred when the vehicle starting from the parent v_q at time t first visits $v_{q,i}$ and then processes all tasks in $T(v_{q,i})$ (including also $v_{q,i}$ itself) according to an optimal schedule under depth-first routing constraint, and returns to v_q. And let $W(v_{q,i})$ and $P(v_{q,i})$ denote the sum of travel times and the sum of processing times, respectively, which are also computed as $L_t(v_{q,i})$ is computed. Note that, for $v_{q,0} = v_q$, this definition states that, starting at time t, the vehicle processes $T(v_{q,0})$ (i.e., v_q itself) immediately; hence $L_t(v_{q,0}) = t + p(v_q) - d(v_q)$, and that $P(v_{q,i})$ does not include $p(v_q)$. $W(v_{q,i})$ and $P(v_{q,i})$ are constants, which are independent of time t and routing schedule of the vehicle, under depth-first routing constraint.

Lemma 1. For any $v_{q,i} \in V$ and times $t > t'$, it holds

$$L_t(v_{q,i}) = L_{t'}(v_{q,i}) + t - t'. \tag{9}$$

Proof: By simply changing the start time from t' to t, and keeping the schedule in $T(v_{q,i})$ the same as the optimal schedule of start time t', the maximal lateness in $T(v_{q,i})$ increases by $t - t'(> 0)$, and hence

$$L_t(v_{q,i}) \leq L_{t'}(v_{q,i}) + (t - t').$$

Similarly, if we start from v_q at time t' and process $T(v_{q,i})$ by the same schedule as an optimal schedule of start time t, then

$$L_{t'}(v_{q,i}) \leq L_t(v_{q,i}) - (t - t').$$

This proves the lemma. □

For a task $v_{q,i} \in V$ and time t, define

$$M_t(v_{q,i}) = L_t(v_{q,i}) - W(v_{q,i}) - P(v_{q,i}). \tag{10}$$

Lemma 2. For TREE-VSP(L_{max}) with depth-first routing constraint, there is an optimal schedule π^{DF} such that

$$\pi^{DF}(v_0) = (\pi^{DF}(v_{0,\psi(0)}), \pi^{DF}(v_{0,\psi(1)}), \ldots, \pi^{DF}(v_{0,\psi(k_0)})), \tag{11}$$

where permutation $\psi = (\psi(0), \psi(1), ..., \psi(k_0))$ satisfies

$$M_0(v_{0,\psi(0)}) \geq M_0(v_{0,\psi(1)}) \geq \ldots \geq M_0(v_{0,\psi(k_0)}), \tag{12}$$

and $\pi^{DF}(v_{0,i})$, $i = 0, 1, ..., k_0$, are optimal schedules for subtrees $T(v_{0,i})$ with start time $t = 0$.

Proof: Let $\pi(v_0) = (\pi^{DF}(v_{0,\sigma(0)}), \pi^{DF}(v_{0,\sigma(1)}), ..., \pi^{DF}(v_{0,\sigma(k_0)}))$ be any schedule defined by a permutation σ, and $\pi^{DF}(v_0)$ be a schedule defined by ψ in (12). If $\sigma(j) = \psi(0)$ and $j \neq 0$, we construct the schedule π' from π by

$$\pi'(v_0) = (\pi^{DF}(v_{0,\sigma(j)}), \pi^{DF}(v_{0,\sigma(0)}), ..., \pi^{DF}(v_{0,\sigma(j-1)}), \pi^{DF}(v_{0,\sigma(j+1)}), ..., \pi^{DF}(v_{0,\sigma(k_0)})).$$

Distiguish the maximum latenesses in π and π' by L_t and $L'_{t'}$, and the start times in π and π' at v_0 to $v_{0,\sigma(i)}$ by t_i and t'_i, $i = 0, 1, ..., k_0$, respectively. Then

$$t_i = \sum_{l=0}^{i-1}\{W(v_{0,\sigma(l)}) + P(v_{0,\sigma(l)})\}, \tag{13}$$

$$t'_i = \begin{cases} 0, & i = j, \\ W(v_{0,\sigma(j)}) + P(v_{0,\sigma(j)}) + t_i, & 0 \leq i \leq j-1, \\ t_i, & j+1 \leq i \leq k_0, \end{cases} \tag{14}$$

and hence

$$\begin{aligned} L'_{t'_i}(v_{0,\sigma(i)}) &= L_0(v_{0,\sigma(i)}) + W(v_{0,\sigma(j)}) + P(v_{0,\sigma(j)}) \\ &\quad + \sum_{l=0}^{i-1}\{W(v_{0,\sigma(l)}) + P(v_{0,\sigma(l)})\} \\ &\leq L_0(v_{0,\sigma(j)}) + \sum_{l=0}^{i}\{W(v_{0,\sigma(l)}) + P(v_{0,\sigma(l)})\} \\ &\leq L_0(v_{0,\sigma(j)}) + \sum_{l=0}^{j-1}\{W(v_{0,\sigma(l)}) + P(v_{0,\sigma(l)})\} \\ &= L_{t_j}(v_{0,\sigma(j)}), \qquad 0 \leq i \leq j-1. \end{aligned} \tag{15}$$

Note that Lemma 1 and $M_0(v_{0,\sigma(i)}) \leq M_0(v_{0,\sigma(j)})$ for $0 \leq i \leq j-1$ are used to derive the above relation. Moreover it is clear that $L'_{t'_j}(v_{0,\sigma(j)}) = L'_0(v_{0,\sigma(j)}) \leq L_{t_j}(v_{0,\sigma(j)})$, and $L'_{t'_i}(v_{0,\sigma(i)}) = L_{t_i}(v_{0,\sigma(i)})$ for $i = j+1, ..., k_0$. This shows that $L_{max}(\pi'(v_0)) \leq L_{max}(\pi(v_0))$, and therefore we can state without loss of generality that $T(v_{0,\psi(0)})$ is the first subtree to be processed in an optimal schedule. (Of course, if $\sigma(0) = \psi(0)$, $T(v_{0,\sigma(0)})$ is the first subtree in an optimal schedule.)

Now t'_0 is the start time of the second subtree to be processed in the optimal schedule. By applying the above argument to the rest subtrees, we see that the second subtree in the optimal schedule maximizes $M_{t'_0}(v_{0,i})$ among all $i \neq \psi(0)$. However, by Lemma 1, the one that maximizes $M_{t'_0}(v_{0,i})$ also maximizes $M_0(v_{0,i})$. This shows that $T(v_{0,\psi(1)})$ can be considered as the second subtree in the optimal schedule. By repeating this argument, we can transform π into π^{DF} without increasing the maximum lateness, and show straightforwardly that π^{DF} is optimal. □

Theorem 2. TREE-VSP(L_{max}) with depth-first routing constraint has an optimal schedule π^{DF} such that

$$\pi^{DF}(v_q) = (\pi^{DF}(v_{q,\psi(0)}), \pi^{DF}(v_{q,\psi(1)}), ..., \pi^{DF}(v_{q,\psi(k_q)})) \tag{16}$$

satisfies

$$M_0(v_{q,\psi(0)}) \geq M_0(v_{q,\psi(1)}) \geq \ldots \geq M_0(v_{q,\psi(k_q)}) \tag{17}$$

for every $v_q \in V - V_{\text{leaf}}$.

Proof: By generalizing the argument of Lemma 2, it is not difficult to see that an optimal schedule π^{DF} in $T(v_q)$, starting at time t_q, is given by a permutation ψ with $M_{t_q}(v_{q,\psi(0)}) \geq M_{t_q}(v_{q,\psi(1)}) \geq \ldots \geq M_{t_q}(v_{q,\psi(k_q)})$. However, the latter condition is equivalent to $M_0(v_{q,\psi(0)}) \geq M_0(v_{q,\psi(1)}) \geq \ldots \geq M_0(v_{q,\psi(k_q)})$, by property $M_{t_q}(v_{q,\psi(i)}) = M_0(v_{q,\psi(i)}) + t_q$. □

3.2 A polynomial time algorithm

Based on Theorem 2, we present a polynomial time algorithm, called DF(L_{max}), to solve TREE-VSP(L_{max}) with depth-first routing constraint. The optimal subschedule $\pi^{DF}(v_q)$ is computed in SEARCH$(v_q; L_0(v_q), W(v_q), P(v_q), \pi^{DF}(v_q))$, by recursively calling this procedure. We can assume without loss of generality that the parent of v_0 is v_0 itself in the procedure. $LATE$, $WEIGHT$ and $PROC$ are temporary storages used to calculate $L_0(v_q)$, $W(v_q)$ and $P(v_q)$, respectively.

```
Algorithm DF($L_{max}$)
  begin
    SEARCH($v_0; L_0(v_0), W(v_0), P(v_0), \pi^{DF}(v_0)$)
  end.

procedure SEARCH($v_q; L_0(v_q), W(v_q), P(v_q), \pi^{DF}(v_q)$)
  begin
    $k_q$ :=(the number of children of $v_q$ in $T$);
    $v_p$ :=(the parent of $v_q$ in $T$. Let $v_p = v_0$ if $v_q = v_0$.);
    if $k_q = 0$ then
      $L_0(v_q) := w(v_p, v_q) + p(v_q) - d(v_q)$;
      $W(v_q) := w(v_p, v_q) + w(v_q, v_p)$;  $P(v_q) := p(v_q)$;
      $\pi^{DF}(v_q) := (v_q)$;
      return
    else
      (Computation of optimal schedule at $v_q$)
      $L_0(v_{q,0}) := p(v_q) - d(v_q)$;  $W(v_{q,0}) := 0$;  $P(v_{q,0}) := p(v_q)$;
      $\pi^{DF}(v_{q,0}) := (v_q)$;
      for $i := 1$ to $k_q$
        begin
          SEARCH($v_{q,i}$;  $L_0(v_{q,i}), W(v_{q,i}), P(v_{q,i}), \pi^{DF}(v_{p,i})$);
        end;
      Compute a permutation $\psi$ on $\{0, 1, .., k_q\}$ such that it satisfies
      $M_0(v_{q,\psi(0)}) \geq M_0(v_{q,\psi(1)}) \geq \ldots \geq M_0(v_{q,\psi(k_q)})$;
```

```
        $\pi^{DF}(v_q) := (\pi^{DF}(v_{q,\psi(0)}), \pi^{DF}(v_{q,\psi(1)}), ..., \pi^{DF}(v_{q,\psi(k_q)}))$;
        **(Computation of $L_0(v_q), W(v_q)$ and $P(v_q)$)**
        $LATE := 0$; $WEIGHT := w(v_p, v_q)$; $PROC := 0$;
        **for** $i := 0$ **to** $k_q$
          **begin**
            $LATE := \max\{LATE, L_0(v_{q,\psi(i)}) + WEIGHT + PROC\}$;
            $WEIGHT := WEIGHT + W(v_{q,\psi(i)})$;
            $PROC := PROC + P(v_{q,\psi(i)})$
          **end**;
        $L_0(v_q) := LATE$; $W(v_q) := WEIGHT + w(v_q, v_p)$; $P(v_q) := PROC$;
        return
    **end.**
```

Theorem 3. The algorithm DF(L_{max}) solves TREE-VSP(L_{max}) with depth-first routing constraint in $O(n \log n)$ time.
Proof: The DF(L_{max}) constructs π^{DF} of Theorem 2 by searching T in the depth-first manner. Therefore its correctness follows from Theorem 2. To analyze its time complexity, note that traversing all edges in T requires $O(n)$ time. At each vertex v_q, it sorts (k_q+1) elements. Since $\sum_{q=0}^{n-1} k_q = n-1$, the total time for sorting is $\sum_{q=0}^{n-1} O((k_q+1)\log(k_q+1)) = O(n \log n)$. Hence the time complexity of DF(L_{max}) is $O(n \log n)$. □

3.3 DF(L_{max}) as an approximate algorithm

Algorithm DF(L_{max}) can be used as an approximate algorithm for the original TREE-VSP(L_{max}) without depth-first routing constraint. To evaluate its performance, we define the following notations:

$L_{max}(\pi^*)$: The optimal maximum lateness for the original TREE-VSP(L_{max}).
$L_{max}(\pi^{DF})$: The maximum lateness obtained by algorithm DF(L_{max}).
$W = \sum_{(u,v)\in E}(w(u,v) + w(v,u))$: The sum of travel times for all edges.
$P = \sum_{v \in V} p(v)$: The sum of all processing times.
$d_{max} = \max_{v \in V} d(v)$: The maximum due date.
$d_{min} = \min_{v \in V} d(v)$: The minimum due date.

Furthermore we represent the completion time of the last task v_{last} being processed by π^* as $W^* + P$, where W^* denotes the sum of travel times until π^* visits v_{last} for its processing. Hence we have

$$L_{max}(\pi^*) \geq \max\{W^* + P - d_{max}, \ -d_{min}\}. \qquad (18)$$

Let l^* be the length of the shortest path from v_{last} to v_0, then it satisfies $W^* + l^* \geq W$.

Now consider any schedule π', whose last task is also v_{last}, under depth-first routing constraint, then the completion time of v_{last} is $W - l^* + P \leq W^* + P$ (recall that the total travel time is exactly W in any schedule under depth-first routing constraint). Since the schedule π^{DF} is optimal under depth-first routing constraint, it holds that

$$L_{max}(\pi^{DF}) \leq L_{max}(\pi') \leq W^* + P - d_{min}. \qquad (19)$$

Similarly, we obtain

$$L_{max}(\pi^{DF}) \leq W + P - d_{min}, \qquad (20)$$

and hence

$$L_{max}(\pi^{DF}) \leq \min\{W,\ W^*\} + P - d_{min}. \tag{21}$$

Therefore the absolute difference of L_{max} between π^{DF} and π^* is bounded by

$$L_{max}(\pi^{DF}) - L_{max}(\pi^*) \leq \min\{W + P,\ d_{max} - d_{min}\}. \tag{22}$$

Next, we would evaluate performance ratio $L_{max}(\pi^{DF})/L_{max}(\pi^*)$. However, this measure may not be appropriate for problem instances with $L_{max}(\pi^*) \leq 0$. To avoid such situation, we transform a given problem instance into another one with $L_{max}(\pi^*) > 0$ such that the two instances have the same optimal schedules and the same approximate schedules obtained by DF(L_{max}). Kise et al. [9] suggested such transformation for the one-machine maximum lateness scheduling problem.

Let $\tilde{Q}$ and Q be the instances of TREE-VSP(L_{max}) such that $\tilde{Q}$ is defined on a tree $\tilde{T}$, and has travel times $\tilde{w}(u,v)$, due dates $\tilde{d}(v)$ and processing times $\tilde{p}(v)$, while Q is defined on a tree T, and has $w(u,v)$, $d(v)$ and $p(v)$. Also let $\tilde{L}_{max}$ and L_{max} be the maximum latenesses, and $\tilde{\pi}^*$ and π^* the optimal schedules, and $\tilde{\pi}^{DF}$ and π^{DF} the approximate schedules of $\tilde{Q}$ and Q, respectively. Then we have the following lemma (if we break ties in (17) by an appropriate manner):

Lemma 3. If problem instances Q is obtained from $\tilde{Q}$ by $T = \tilde{T}$, $w(u,v) = \tilde{w}(u,v)$ for all edges (u,v), $d(v) = \tilde{d}(v) - c$ and $p(v) = \tilde{p}(v)$ for all tasks v, where c is a given integer, then the two instances Q and $\tilde{Q}$ have the same π^* and the same π^{DF}. □

From this lemma, without changing the optimal and the approximate schedules, we can transform any problem instance $\tilde{Q}$ into Q by using $c = \tilde{d}_{max}$. If the original instance $\tilde{Q}$ has $\tilde{d}_{max} > \tilde{d}_{min}$, the new instance Q satisfies

$$d_{max} = 0, \quad d_{min} < 0. \tag{23}$$

Theorem 4. For any problem instance Q subject to (23), the performance ratio of the approximate schedule π^{DF} obtained by DF(L_{max}) to the optimal schedule π^* is bounded by

$$\frac{L_{max}(\pi^{DF})}{L_{max}(\pi^*)} \leq 2, \tag{24}$$

and this bound is best possible.

Proof: For the instance Q, the lower bound on the optimal maximum lateness (18) is also positive, i.e., $L_{max}(\pi^*) \geq \max\{W^* + P,\ -d_{min}\} > 0$. Thus, by using this lower bound and (21), the performance ratio satisfies

$$\frac{L_{max}(\pi^{DF})}{L_{max}(\pi^*)} \leq \frac{W^* + P - d_{min}}{\max\{W^* + P,\ -d_{min}\}} \leq 2.$$

To show that this bound is tight, we present a problem instance such that $V = \{v_0, v_1, v_2, v_3, v_4\}$ and $E = \{(v_0,v_1),(v_0,v_2),(v_1,v_3),(v_2,v_4)\}$. The travel times are given by $w(v_0,v_1) = w(v_1,v_0) = w(v_0,v_2) = w(v_2,v_0) = w(v_1,v_3) = w(v_2,v_4) = \varepsilon(>0)$ and $w(v_3,v_1) = w(v_4,v_2) = w(>0)$. The due dates are $d(v_0) = d(v_3) = d(v_4) = 0$ and $d(v_1) = d(v_2) = -w$, and all of the processing times are zero. Since we obtain $\pi^* = (v_0,v_1,v_2,v_4,v_3)$ and $\pi^{DF} = (v_0,v_1,v_3,v_2,v_4)$, $L_{max}(\pi^*) = 7\varepsilon + w$ and $L_{max}(\pi^{DF}) = 4\varepsilon + 2w$. This shows $L_{max}(\pi^{DF})/L_{max}(\pi^*) \to 2$ when $\varepsilon \to 0$. □

4. Conclusion

In this paper we considered the single-vehicle scheduling problem on a tree, called TREE-VSP(L_{max}). After proving its *NP*-hardness by reducing TREE-VSP(C) to TREE-VSP(L_{max}), we pointed out that TREE-VSP(L_{max}) with depth-first routing constraint is polynomially solvable. As our future work, it remains to examine other criteria involving due dates such as total tardiness and the number of tardy tasks. It may also be interesting to develop fuzzy inference-based algorithms, which makes use of fuzzy information of due dates (e.g., see [3]).

Among other interesting topics are the vehicle scheduling problem with different types of road networks such as circular paths and planar networks, and the multi-vehicle problems.

Acknowledgment

The authors wish to thank Assistant Professor Mutsunori Yagiura of Kyoto University for his comment about Theorem 3.

References

[1] Atallah, M. and S. Kosaraju, Efficient Solutions to Some Transportation Problems with Application to Minimizing Robot Arm Travel, *SIAM J. Comput.*, 17(1988), 849-869.

[2] Averbakh, I. and O. Berman, Sales-Delivery Man Problems on Treelike Networks, *Networks*, 25(1995), 45-58.

[3] Cheng, J. L., H. Kise and H. Matsumoto, A Branch-and-Bound Algorithm with Fuzzy Inference for the 3-Machine Flowshop Scheduling Problem, *Proceedings of 1994 Japan-U.S.A. Symposium on Flexible Automation, ISCIE*, (1994), 791-797.

[4] Frederickson, G., A Note on the Complexity of a Simple Transportation Problem, *SIAM J. Comput.*, 22-1(1993), 57-61.

[5] Frederickson, G. and D. Guan, Preemptive Ensemble Motion Planning on a Tree, *SIAM J. Comput.*, 21-6(1992), 1130-1152.

[6] Frederickson, G., M. Hecht and C. Kim, Approximation Algorithms for Some Routing Problems, *SIAM J. Comput.*, 7-2(1978), 178-193.

[7] Garey, M. and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, (1979).

[8] Karuno, Y., H. Nagamochi and T. Ibaraki, Vehicle Scheduling on a Tree with Release and Handling Times, *Lecture Notes in Computer Science 762*, Springer-Verlag, (1993), 486-495.

[9] Kise, H., T. Ibaraki and H. Mine, Performance Analysis of Six Approximation Algorithms for the One-Machine Maximum Lateness Scheduling Problem with Ready Times, *Journal of the Operations Research Society of Japan*, 22-3(1979), 205-224.

[10] Minieka, E., The Delivery Man Problem on a Tree Network, *Annals of Operations Research*, 18(1989), 261-266.

[11] Nagamochi, H., K. Mochizuki and T. Ibaraki, Complexity of Single Vehicle Scheduling Problem on Graphs, *Technical Report of IEICE*, COMP94-19(1994), 11-20.

[12] Psaraftis, H., M. Solomon, T. Magnanti and T. Kim, Routing and Scheduling on a Shoreline with Release Times, *Management Science*, 36-2(1990), 212-223.

[13] Solomon, M., Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints, *Operations Research*, 35-2(1987), 254-265.

[14] Solomon, M. and J. Desrosiers, Time Window Constrained Routing and Scheduling Problems: A Survey, *Transportation Sci.*, 22(1988), 1-13.

[15] Tsitsiklis, J. N., Special Cases of Traveling Salesman and Repairman Problems with Time Windows, *Networks*, 22(1992), 263-282.

Yoshiyuki KARUNO
Department of Mechanical and System Engineering,
Faculty of Engineering and Design,
Kyoto Institute of Technology,
Matsugasaki, Sakyo-ku, Kyoto 606, JAPAN
e-mail: karuno@ipc.kit.ac.jp

Hiroshi NAGAMOCHI and Toshihide IBARAKI
Department of Applied Mathematics and Physics,
Graduate School of Engineering,
Kyoto University,
Kyoto 606, JAPAN