Journal of the Operations Research Society of Japan
2005, Vol. 48, No. 3, 196-206

A TREE PARTITIONING PROBLEM ARISING FROM AN EVACUATION PROBLEM IN TREE DYNAMIC NETWORKS

Satoko Mamada
Osaka University

Takeaki Uno
National Institute of Informatics

Kazuhisa Makino
Osaka University

Satoru Fujishige
Kyoto University

(Received December 1, 2004; Revised February 25, 2005)

Abstract In this paper, we present a first polynomial time algorithm for the monotone min-max tree partitioning problem and show that the min-max tree partitioning problem is NP-hard if the cost function is not monotone, and that the min-sum tree partitioning problem is NP-hard even if the cost function is monotone. We also consider an evacuation problem in dynamic networks as an application of the tree partitioning problem. The evacuation problem is one of the basic studies on crisis management systems for evacuation guidance of residents against large-scale disasters. We restrict our attention to tree networks and consider flows such that all the supplies going through a common vertex are sent out through a single arc incident to it, since one of the ideal evacuation plans makes everyone to be evacuated fairly and without confusion.

Keywords: Combinatorial optimization, algorithm, partitioning problem, tree network, evacuation problem, dynamic flow

1. Introduction

We consider the monotone min-max tree partitioning problem. Let $T = (V, E)$ be a tree with a vertex set V and an edge set E, S be a subset of V and $C : \{W \subseteq V \mid |W \cap S| = 1\} \to \mathbf{R}_+$ be a cost function. Here, $\mathbf{R}_+$ denotes the set of all nonnegative reals. Our problem is to find a partition of V into k sets $V_1, V_2, \cdots, V_k$ such that for each $i = 1, 2, \cdots, k$ the subgraph induced by V_i is connected, V_i contains a single sink s_i, and $\max_{i=1,\cdots,k} C(V_i)$ is minimized. For general k, it is known that the problem can be solved in $n(c \log n)^{k+1}$ time for some constant c [15], where n is the number of vertices in a given network. In this paper, we first present a polynomial time algorithm for the problem when k is general. More precisely, we show that if the cost function is of min-max type and is monotone, then the problem can be solved in polynomial time. We also show that the partitioning problem becomes intractable if the min-max cost function is not monotone, and the min-sum partitioning problem is intractable even if the cost function is monotone. Our results strengthen the previous work [2] for the partitioning problems with general cost functions, where it is shown that the partitioning problem (without specified vertices) can be solved in polynomial time if the cost function is either (i) max-min and monotone, or (ii) min-max and invariant.

We also investigate an evacuation problem in dynamic networks as an application of the tree partitioning problem. The evacuation problem is considered as one of the basic studies on crisis management systems for evacuation guidance of residents against large-scale disasters. Recently, it has widely been recognized how important it is to establish crisis management systems against large-scale disasters such as big earthquakes. It is one of the most important issues in the crisis management against disasters to secure evacuation

pathways and to effectively guide residents to safe places.

Mathematical models for evacuation problems are classified into two groups: microscopic models and macroscopic models. Microscopic models are used for experimental analyses by simulation of behaviors of individual residents. Typical such microscopic models are cellular automata simulation models [3] and probabilistic models [9] for pedestrians and traffic movement. In macroscopic models behaviors of individual residents are not directly treated but residents are regarded as a homogeneous group. There are several classes of mathematical macroscopic models such as static networks, dynamic networks, and traffic assignments [4, 6, 13, 17, 18].

In this paper, we adopt dynamic networks as a model for evacuation [6, 11]. Namely, we regard evacuation problems as flow problems on dynamic networks. A dynamic network is defined by a directed graph $G = (V, A)$ with capacities $u(a)$ and transit times $\tau(a)$ on its arcs $a \in A$. For example, if we consider building evacuation [5, 7], vertices $v \in V$ model workplaces, hallways, stairwells, and so on, and arcs $a \in A$ model the connection between these parts of the building. For each arc $a = (v, w)$, $u(a)$ represents the maximum number of people per unit time which can traverse the component corresponding to a per unit time, and $\tau(a)$ denotes the time it takes to traverse $a = (v, w)$ from v to w.

The quickest transshipment problem is defined by a dynamic network with several sources and sinks; each source has a specified supply and each sink has a specified demand. The problem is to send exactly the right amount of flow out of each source and into each sink in the minimum overall time. Here sources can be regarded as places where the people to evacuate are staying, and sinks as emergency exits. Hoppe and Tardos [12] constructed a polynomial time algorithm for the problem, which is the first and still only one polynomial time algorithm for the problem.[1] Unfortunately, their algorithm requires polynomial time of high degree complexity, and hence is not practical.

In our evacuation problem, we restrict our attention to tree networks and consider flows such that

(∗) all the supplies going through a common vertex are sent out through a single arc incident to it toward a single sink,

since one of the ideal evacuation plans makes everyone to be evacuated fairly and without confusion. Although our dynamic flow in the above sense may not directly be used for actual evacuation, it gives us guidelines to evacuation plans.

Given a dynamic tree network $\mathcal{N} = (T = (V, E), u, \tau, b)$, where b is a supply function, and a sink set $S \subseteq V\,(|S| = k)$, we define $C(W)$ for $W \subseteq V$ to be the completion time for a quickest flow f in the network $\mathcal{N}[W]$ with a sink set $S \cap W$, i.e., the time in which f can send all the supplies $b(v)\,(v \in W \setminus (S \cap W))$ to a sink set $S \cap W$ as quickly as possible, where $\mathcal{N}[W]$ denotes the network induced by W. From restriction (∗) given above, our evacuation problem can be regarded as the tree partitioning problem of a tree with specified vertices. Namely, our evacuation problem is to find a k-partition of V minimizing $\max_i C(V_i)$ such that each component contains exactly one vertex in S. It is known that the problem when $k = 1$ can be solved in $O(n \log^2 n)$ time [16], i.e., we compute the completion time in which all the initial supplies can be sent to a single sink. Note that restriction (∗) on flows is automatically satisfied when $k = 1$. For general k, it is known that the problem can be solved in $n(c \log n)^{k+1}$ time for some constant c [15]. In this paper, we first present a polynomial time algorithm for the problem when k is general. We remark that Fleischer

[1] The continuous version of the quickest transshipment problem was solved by Fleischer and Tardos [8] in polynomial time.

and Tardos [8] and Hoppe and Tardos [12] construct polynomial time algorithms without restriction $(*)$ given above.

The rest of the paper is organized as follows. The next section formally defines the problem and introduces some notations. Section 3 considers the evacuation problem as an application of the tree partitioning problem, Section 4 presents a polynomial time algorithm for our problem, and Section 5 discusses the complexity of some other tree partitioning problems. Finally, we conclude the paper with Section 6.

2. Min-max Tree Partitioning Problem

Let $T = (V, E)$ be an undirected tree, $S = \{s_1, \cdots, s_k\}$ be a subset of V and $C : \{W \subseteq V \mid |W \cap S| = 1\} \to \mathbf{R}_+$ be a cost function. We note here that the cost function C is defined on a set of vertex subsets containing exactly one vertex in S. A partition $\{V_1, \cdots, V_k\}$ of V is called *feasible* if V_i contains s_i for each $i = 1, \cdots, k$ and the subgraph $T[V_i]$ of T induced by V_i is connected. Here V_i can be regarded as the area which is covered with (or controlled by) a facility $s_i \in S$, for example. We denote by $\mathcal{P}(V)$ the family of all feasible partitions of V. Then the *min-max tree partitioning problem* computes a partition that attains

$$\min_{\{V_1,\cdots,V_k\}\in\mathcal{P}(V)} \max_i C(V_i). \tag{2.1}$$

For example, let us regard S as the set of facilities in some network T. Then V_i can be regarded as the area which is covered with (or controlled by) $s_i \in S$, and thus the problem computes a minimum-cost partition with respect to the min-max criterion.

For the tree partitioning problem with a cost function C, we call C *monotone* if for all vertex subsets W and W' with $W \subseteq W'$ and $|W \cap S| = |W' \cap S| = 1$ we have $C(W) \leq C(W')$. The monotonicity is quite a natural assumption on the cost function. Intuitively, cost $C(W')$ is greater than or equal to $C(W)$ if the facility s in $S \cap W$ covers area W' larger than W. For example, the evacuation problem has a monotone cost function.

3. Evacuation Problem

We consider a dynamic tree network $\mathcal{N} = (T = (V, A), u, \tau, b)$, where V is a set of vertices, A is a set of arcs, $u : A \to \mathbf{R}_+$ is the upper bound for the rate of a flow that enters each arc per unit time, $\tau : A \to \mathbf{R}_+$ is a transit time function, and $b : V \to \mathbf{R}_+$ is a supply function. Here, T contains an arc (v, w) if (w, v) is an arc of T, where we obtain an undirected tree from T by ignoring the direction of arcs and then identifying parallel edges.

The problem considered in this paper is to compute a quickest flow which sends given initial supplies $b(v)$ $(v \in V \setminus S)$ to a given sink set $S \subseteq V$. Here we assume that flows satisfy restriction $(*)$ given in Section 1, i.e., all the supplies that go through a common vertex are sent to a single sink.

For any arc $a \in A$ and any $\theta \in \mathbf{R}_+$, we denote by $f_a(\theta)$ the rate of a flow entering arc a at time θ which arrives at the head of a at time $\theta + \tau(a)$. We call $f_a(\theta)$ $(a \in A, \theta \in \mathbf{R}_+)$ a *continuous dynamic flow* in T (with a sink set S) if it satisfies the following three conditions (a), (b), and (c):

(a) (Capacity constraints): For any arc $a \in A$ and $\theta \in \mathbf{R}_+$,

$$0 \leq f_a(\theta) \leq u(a).$$

(b) (Flow conservation): For any $v \in V \setminus S$ and $\Theta \in \mathbf{R}_+$,

$$\sum_{a\in\delta^+ v} \int_0^\Theta f_a(\theta)d\theta - \sum_{a\in\delta^- v} \int_{\tau(a)}^\Theta f_a(\theta - \tau(a))d\theta \leq b(v).$$

(c) (Flow completion): There exists a time $\Theta \in \mathbf{R}_+$ such that

$$\sum_{a\in\Delta^-(S)} \int_{\tau(a)}^{\Theta} f_a(\theta - \tau(a))d\theta - \sum_{a\in\Delta^+(S)} \int_0^{\Theta} f_a(\theta)d\theta = \sum_{v\in V\setminus S} b(v).$$

Here δ^+v and δ^-v are, respectively, the set of arcs having v as their tails and heads, and $\Delta^+(S) = \{(v,w) \in A \mid v \in S, w \notin S\}$ and $\Delta^-(S) = \{(v,w) \in A \mid v \notin S, w \in S\}$.

Based on restriction $(*)$ given above, we consider continuous dynamic flows that satisfy

(d-1) For any arc $a = (v,w) \in A$ with $v \in S$ and $\theta \in \mathbf{R}_+$, we have $f_a(\theta) = 0$.

(d-2) For each $v \in V \setminus S$ there exists at most one arc $a \in \delta^+v$ such that $f_a(\theta) \neq 0$ for some $\theta \in \mathbf{R}_+$.

We call the flow satisfying (d-1) and (d-2) *feasible*. For a feasible (continuous dynamic) flow f, let θ_f denote the completion time for f, i.e., the minimum Θ in Condition (c). A feasible flow f with the minimum θ_f is called a *quickest* flow. Our problem, called the *evacuation problem*, is to compute a quickest flow in a given network $\mathcal{N}$ with a sink set $S \subseteq V$. From restriction $(*)$ on flows, the evacuation problem can be regarded as a special case of the tree partitioning problem in Section 2.

For the evacuation problem, if we define $C(W)$ for $W \subseteq V$ to be the completion time for a quickest flow f in the network $\mathcal{N}[W]$ with a sink set $S \cap W$, where $\mathcal{N}[W]$ denotes the network induced by W, then the problem of computing $C(W)$ can be regarded as a min-max tree partitioning problem given by (2.1) with V being replaced by W. Note that the cost function C for the evacuation problem can thus be defined on the set of all vertex subsets W, and it is in fact a min-max function

$$C(W) = \min_{\{W_1,\cdots,W_h\}\in\mathcal{P}(W)} \max_i C(W_i), \tag{3.1}$$

where we define $C(W) = +\infty$ if $\mathcal{P}(W) = \emptyset$. We further note that $C(W)$ for the evacuation problem can be computed in $O(|W| \log^2 |W|)$ time if $|W \cap S| = 1$ ([16]).

In the next section, we solve the monotone min-max tree partitioning problem that includes our evacuation problem as a special case.

4. Monotone Min-max Tree Partitioning Problem

In this section, we show that the min-max tree partitioning problem is solvable in polynomial time if the cost function is monotone. For the sake of simplicity, we first assume without loss of generality that any vertex in S is a leaf. This assumption can be validated as follows.

Let S_l and $S_{non\text{-}l}$ denote the set of leaves and non-leaves in S, respectively. For $T = (V,E)$ and $S \subseteq V$, we construct a tree $T^* = (V^*, E^*)$ such that $V^* = V \cup \{v_i \mid s_i \in S_{non\text{-}l}\}$ and $E^* = E \cup \{(s_i, v_i) \mid s_i \in S_{non\text{-}l}\}$, where v_i is a new vertex (i.e., $v_i \notin V$). Let $S^* = S_l \cup \{v_i \mid s_i \in S_{non\text{-}l}\}$. Clearly T^* is a tree with at most $2n$ vertices and any $s \in S^*$ is a leaf, where n denotes the number of vertices in T. Furthermore, we define a cost function $C^* : \{W \subseteq V^* \mid |W \cap S^*| = 1\} \to \mathbf{R}_+$ by

$$C^*(W) = \begin{cases} 0 & \text{if } |W \cap S| = 0 \\ C(W \cap V) & \text{if } |W \cap S| = 1 \\ +M & \text{if } |W \cap S| \geq 2 \end{cases}$$

for each $W \subseteq V^*$ with $|W \cap S^*| = 1$, where M denotes a sufficiently large real. It is not difficult to see that C^* is monotone, and the cost of a feasible partition $\mathcal{P}(V^*)$ of V^* is less

than $+M$ if and only if $\{W \cap V \mid W \in \mathcal{P}(V^*)\}$ is a feasible partition of V. Hence, we assume without loss of generality that any vertex in S is a leaf. Throughout the remainder of this section, we always assume this property.

Let us extend the domain $\{W \subseteq V \mid |W \cap S| = 1\}$ of the cost function C to 2^V as (3.1). Then we apply dynamic programming to T, in order to compute the optimal objective function value $C(V)$.

Let us regard T as a tree rooted at an arbitrary internal vertex r, and let V_v denote the set of vertices that are descendants of v (including v). As usual, we perform dynamic programming in a bottom-up manner, i.e., we compute $C(V_v)$ by using $C(V_w)$ $(w \in V_v \setminus \{v\})$. For a vertex $s \in V_v \cap S$, let $C(V_v; s)$ denote the minimum cost among all possible feasible partitions of V_v such that s and v belong to the same component. Namely,

$$C(V_v; s) = \min_{\{W_1,\cdots,W_h\} \in \mathcal{P}(V_v): s, v \in W_1} \max_i C(W_i), \tag{4.1}$$

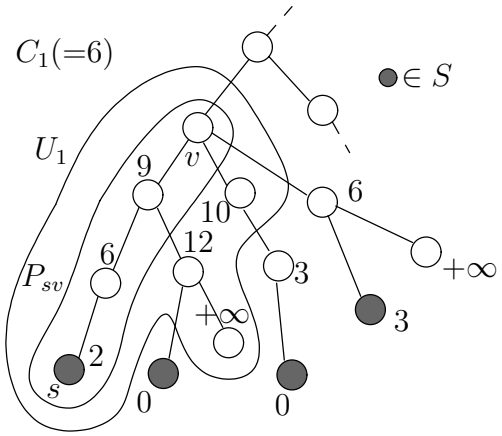
where $C(V_v; s) = +\infty$ if no such partition $\mathcal{P}(V_v)$ exists. It is easy to see that $C(V_v) = \min_{s \in V_v \cap S} C(V_v; s)$. Let us order the costs in

$$\{C(V_w) \mid w \in V_v \setminus \{v\}, C(V_w) \geq \max_{s' \in (V_v \cap S) \setminus \{s\}} C(\{s'\})\} \tag{4.2}$$

as $C_0 < C_1 < \cdots < C_\ell$. For an $s \in V_v \cap S$, P_{sv} denotes the set of vertices in the path between s and v, and for $i = 0, 1, \cdots, \ell$, let U_i denote the set of vertices

$$U_i = V_v \setminus \Big(\{w \in V_t \mid t \in V_v \setminus P_{sv},\ C(V_t) \leq C_i\}\Big) \tag{4.3}$$

(see an illustrative example given in Figure 1). Note that $v, s \in U_i$, $P_{sv} \subseteq U_i$ and $U_i \subseteq V_v$. Then we have the following lemma.



C_0=3, C_1=6, C_2=9, C_3=10, C_4=12, C_5=$+\infty$

Figure 1: An example of $C_0, \cdots, C_\ell$, P_{sv}, and U_i, where $C(V_v)$ is attached for each vertex v

Lemma 4.1 *Let v be a non-leaf in V and s be a vertex in $S \cap V_v$. For $i = 0, 1, \cdots, \ell$, let C_i and U_i be defined as above. Then we have*

$$C(V_v; s) = \min_{i=0,\cdots,\ell} \Big\{ \max\{C(U_i), C_i\} \Big\}. \tag{4.4}$$

Proof. Let us first show that

$$C(V_v; s) \leq \max\{C(U_i), C_i\} \tag{4.5}$$

holds for all i. Let $W_1, \cdots, W_r$ be connected components of the graph that is obtained by removing U_i from $T[V_v]$. Then $P = \{W_1, \cdots, W_r, U_i\}$ is a partition of V_v, and it is easy to see that we can create from this P a feasible partition P^* such that $\max_{W \in P^*} C(W) = \max_{W \in P} C(W)$. Since $\max_{W \in P} C(W) \leq \max\{C(U_i), C_i\}$, we have $C(V_v; s) \leq \max\{C(U_i), C_i\}$.

We next prove (4.4). Let us assume that $C(V_v; s)$ satisfies $C_{i^*} \leq C(V_v; s) < C_{i^*+1}$, where $C_{i^*+1} = +\infty$ if $i^* = \ell$ and $C_\ell < +\infty$. Note that such an i^* exists, since we have $C(V_v; s) \geq C(\{s\}; s)$ from (4.1) and the monotonicity of C. Let P be a feasible partition of V_v that attains $C(V_v; s)$. Then U_{i^*} is contained in some set W in P, i.e., U_{i^*} must be covered with s, since otherwise $C(V_v; s) \geq C_{i^*+1}$ holds. This implies

$$C(V_v; s) \geq C(W) \geq C(U_{i^*}), \tag{4.6}$$

and hence we have $C(V_v; s) \geq \max\{C(U_{i^*}), C_{i^*}\}$. This together with (4.5) proves (4.4). □

The above lemma implies that $C(V_v; s)$ can be computed from $C(V_w)$ $(w \in V_v \setminus \{v\})$ in polynomial time, and $C(V_v)$ can be computed as $C(V_v) = \min_{s \in V_v \cap S} C(V_v; s)$. Therefore, dynamic programming approach based on Lemma 4.1 solves our partitioning problem by computing $C(V)$ in polynomial time.

More precisely, for any set $W \subseteq V$ with $|W \cap S| = 1$, denote by η the time sufficient to compute $C(W)$. Assuming that $C(V_w)$ $(w \in V_v \setminus \{v\})$ have already been computed, we can compute $C(V_v; s)$ in $\mathrm{O}(n(\log n + \eta))$ time, since $\ell \leq |V_v| \leq n$ and it takes $\mathrm{O}(n \log n)$ time to sort $C(V_w)$ $(w \in V_v \setminus \{v\})$. This means that $C(V_v)$ can be computed in $\mathrm{O}(nk(\log n + \eta))$ time, where $k = |S|$, and hence we have an $\mathrm{O}(n^2 k(\log n + \eta))$ time algorithm for the monotone min-max tree partitioning problem. However, this is rather time-consuming. The following lemma helps us to reduce the time complexity.

Lemma 4.2 *Let $g : \{0, 1, \cdots, \ell\} \to \mathbf{R}_+$ be a function defined by $g(i) = \max\{C(U_i), C_i\}$, and let i^* be the smallest i with $C(U_i) < C_{i+1}$. Then we have $g(0) \geq \cdots \geq g(i^*) \leq g(i^*+1) \leq \cdots \leq g(\ell)$, i.e., g is lower unimodal and its minimum is attained at $i = i^*$.*

Proof. Let us first show that g is lower unimodal. Since $U_0 \supseteq U_1 \supseteq \cdots \supseteq U_\ell$, we have $C(U_0) \geq C(U_1) \geq \cdots \geq C(U_\ell)$ from the monotonicity of C. Moreover, we have $C_0 \leq C_1 \leq \cdots \leq C_\ell$. These two facts imply that g is lower unimodal.

We then show that the minimum of g is attained at $i = i^*$. By the definition of i^*, we have $g(i^*) < g(i)$ for all i with $i > i^*$. For $i\,(< i^*)$, we have $g(i) \geq g(i+1)$, since $C(U_i) \geq C(U_{i+1})$ and $C(U_i) \geq C_{i+1}$. Thus $g(i^*) = \min_i g(i)$ holds. □

Lemmas 4.1 and 4.2 immediately imply that we can compute $C(V_v; s)$ from $C(V_w)$ $(w \in V_v \setminus \{v\})$ in $\mathrm{O}((n + \eta) \log n)$ time, by applying binary search to (4.4). Hence our partitioning problem is solvable in $\mathrm{O}(nk(n + \eta) \log n)$ time. However, this computes $C(V_v; s)$ independently. To solve the problem more efficiently, we compute $C(V_v; s)$ by using the information about $C(V_w; s)$ for a vertex w which is a child of v and an ancestor of s.

Algorithm *DP*

Step 1: $W := V$; /* W denotes the set of vertices v whose $C(V_v)$ are not computed yet.*/
for each leaf v in T **do** compute $C(V_v)$ and update $W := W \setminus \{v\}$;
/* we regard T as a tree rooted at an arbitrary internal vertex r. */
for each $s \in S$ **do** $\lambda(s) := C(V_s)$; /* Note that any $s \in S$ is a leaf. */
Step 2: **for each** leaf v of $T[W]$ **do**
begin
if $V_v \cap S = \emptyset$ **then** $C(V_v) := +\infty$, $W := W \setminus \{v\}$ and go to Step 2;
sort the elements in $\{C(V_w) \mid w \in V_v \setminus \{v\}\}$ as $C_0 < C_1 < \cdots < C_\ell$;
for each s in $V_v \cap S$ **do**
begin
(2-1) compute i such that $C_i \leq \lambda(s) < C_{i+1}$;
(2-2) compute U_i (as (4.3)) and $C(U_i)$;
(2-3) **if** $|U_i \cap S| \geq 2$ or $C(U_i) \geq C_{i+1}$ **then** $i := i + 1$ and go to (2-2);
(2-4) $\lambda(s) := \max\{C(U_i), C_i\}$;
end;
$C(V_v) := \min\{\lambda(s) \mid s \in V_v \cap S\}$ and $W := W \setminus \{v\}$;
end;
Step 3: output $C(V)$ and halt. □

Here Step 2 computes $C(V_v; s)$ (and hence $C(V_v)$) by using the information about $C(V_w)$ for $w \in V_v \setminus \{v\}$. From Lemma 4.2, Step 2 finds the smallest i^* with $C(U_{i^*}) < C_{i^*+1}$ by checking conditions $C(U_i) < C_{i+1}$ for $i = i_0, i_0 + 1, \cdots, i^*$, where i_0 is an integer with $C_{i_0} \leq \lambda(s) < C_{i_0+1}$. We note that the set $\{C(V_w) \mid w \in V_v \setminus \{v\}\}$ whose elements are sorted in Step 2 contains an element $C(V_w)$ with $C(V_w) < C(\{s'\})$ for some s' in $(S \cap V_v) \setminus \{s\}$, and hence $|U_i \cap S| \geq 2$ might hold (see the difference from (4.2)). Thus (2-3) checks the condition for the next integer, if this is the case. Now it is not difficult to see that $i_0 \leq i^*$ implies the correctness of the algorithm. The next lemma shows that $\lambda(s) = C(V_w; s)$ for a child w of v in P_{sv}, which implies $i_0 \leq i^*$ by the monotonicity of C.

Lemma 4.3 *For a vertex v and $s \in S \cap V_v$, let w denote a child of v in P_{sv}. Then we have $\lambda(s) = C(V_w; s)$ in* (2-1) *of Algorithm* DP *for these v and s.*

Proof. We prove the statement by induction. The statement holds for a vertex v such that all the children of v are leaves of T, since Step 1 sets $\lambda(s)$ as $\lambda(s) := C(V_s)$.

Assuming that the statement holds for a child w of v in P_{sv}, we show that it also holds for v. Let w' be a child of w in P_{sw}. Then, since $C(V_w; s) \geq C(V_{w'}; s)$ from the monotonicity of C, the second for-loop in Step 2 for w and s updates $\lambda(s)$ as $\lambda(s) := C(V_w; s)$. This completes the proof. □

Theorem 4.1 *Algorithm* DP *solves the monotone min-max tree partitioning problem in* $O(nk(n + \eta))$ *time.*

Proof. Since the correctness follows from Lemma 4.3 and the discussion made above, we only show its time complexity.

Steps 1 and 3 can clearly be executed in $O(n\eta)$ time. Let us then consider Step 2. The first and the last lines in the outer for-loop of Step 2 can be done in $O(n)$ time, and therefore they require $O(n^2)$ time in total. The second line sorts the elements in $\{C(V_w) \mid w \in V_v \setminus \{v\}\}$. By applying a standard argument to our sortings, we can see that it can be done in $O(n^2)$ time, in total.

As for the inner for-loop in Step 2, each of (2-1) $\sim$ (2-4) requires $O(n + \eta)$ time. For each s and v, the number of iterations between (2-2) and (2-3) is $i^* - i_0 + 1$, where i^* and

i_0 are those defined in Lemma 4.2 and before Lemma 4.3, respectively. By Lemma 4.3, for any $s \in S$, the number of the iterations is at most $2n$, where all possible $v \in P_{sr}$ are taken into account. Hence we have $O(nk)$ iterations between (2-2) and (2-3). This implies that the inner for-loop requires $O(nk(n+\eta))$ time in total.

Therefore, the algorithm can be executed in $O(nk(n+\eta))$ time. □

By a standard argument of dynamic programming, an optimal partition can also be computed in $O(nk(n+\eta))$ time. We also remark that $\eta = \Omega(n)$ is required in most settings, since the domain of C is a family of sets in 2^V. Moreover, for our evacuation problem we have $\eta = O(n \log^2 n)$ [16]. Hence we have the following.

Corollary 4.4 *The evacuation problem can be solved in* $O(n^2 k \log^2 n)$ *time.*

5. Related Tree Partitioning Problems

In this section, we consider tree partitioning problems with different cost functions. In Section 4 we have presented a polynomial time algorithm for the problem with a monotone min-max cost function. However, the problem with a non-monotone cost function is intractable as shown below.

Theorem 5.1 *The min-max tree partitioning problem is NP-hard in general.*

Proof. We show that the problem is NP-hard, by reducing to it the satisfiability problem (SAT), which is known to be NP-complete [10].

SAT

Input: A CNF $\varphi = \bigwedge_{c_j \in \mathcal{C}} c_j$, where c_j is a clause on a variable set $\{x_1, x_2, \cdots, x_n\}$.
Question: Is φ satisfiable, i.e., does there exist an assignment $y^* \in \{0,1\}^n$ such that $\varphi(y^*) = 1$? □

Given a problem instance I of SAT, we construct the corresponding instance of our problem as follows (see Figure 2).

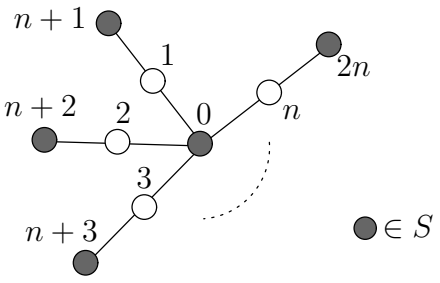


Figure 2: The corresponding instance of our partitioning problem

Let $T = (V, E)$ be a tree with $V = \{0, 1, 2, \cdots, 2n\}$ and $E = \{(0, i), (i, n+i) \mid i = 1, \cdots, n\}$. Let $S = \{0, n+1, n+2, \cdots, 2n\}$, and define a cost function $C : \{W \subseteq V \mid |W \cap S| = 1\} \to \mathbf{R}_+$ by

$$C(W) = \begin{cases} 0 & \text{if } W \cap S = \{n+i\} \text{ for } i = 1, 2, \cdots, n \\ 0 & \text{if } W \cap S = \{0\} \text{ and } \varphi(y) = 1 \text{ for } y \in \{0,1\}^n \text{ defined by } y_j = 1 \text{ if} \\ & \quad j \in W, \text{and 0 otherwise} \\ M & \text{otherwise (i.e., } W \cap S = \{0\} \text{ and } \varphi(y) = 0 \text{ for } y \text{ defined above),} \end{cases}$$

where M denotes a sufficiently large real. Then it is not difficult to see that φ is satisfiable if and only if the corresponding problem instance has cost 0, and φ is unsatisfiable if and only if the corresponding problem instance has cost M. This shows the NP-hardness of our problem. □

We finally consider a min-sum cost function:

$$\min_{\{V_1,\cdots,V_k\}\in\mathcal{P}(V)} \sum_i C(V_i). \tag{5.1}$$

Unlike the problem with a min-max cost function, the problem with a min-sum cost function is intractable, even if the cost function is monotone.

Theorem 5.2 *The min-sum tree partitioning problem is NP-hard, even if the cost function is monotone.*

Proof. We show that the problem is NP-hard, by reducing to it the minimum vertex cover problem, which is known to be NP-hard [10].

Minimum Vertex Cover

Input: An undirected graph $G = (V, E)$, where $V = \{1, 2, 3, \cdots, n\}$.
Find: A minimum vertex cover $W^* \subseteq V$, i.e., a vertex set W^* of minimum size among all W such that $W \cap \{i, j\} \neq \emptyset$ for all $\{i, j\} \in E$. □

Given a problem instance I of Minimum Vertex Cover, we construct the corresponding instance of our problem as follows.

Let $T = (\tilde{V}, \tilde{E})$ be a tree with $\tilde{V} = \{0, 1, 2, \cdots, 2n\}$ and $\tilde{E} = \{(0, i), (i, n + i) \mid i = 1, \cdots, n\}$. Let $S = \{0, n + 1, n + 2, \cdots, 2n\}$ and define a cost function $C : \{W \subseteq \tilde{V} \mid |W \cap S| = 1\} \to \mathbf{R}_+$ by

$$C(W) = \begin{cases} |W| - 1 & \text{if } W \cap S = \{n + i\} \text{ for } i = 1, 2, \cdots, n \\ 0 & \text{if } W \cap S = \{0\} \text{ and } W \setminus \{0\} \text{ is an independent set of } G \text{ (i.e. } V \setminus W \text{ is a vertex cover of } G) \\ n + 1 & \text{otherwise.} \end{cases}$$

We can see that this cost function C is monotone. We claim that the size of a minimum vertex cover of G is equal to the minimum cost for the partitioning problem, which completes the proof.

Let $\{P_0, P_1, \cdots, P_n\}$ be a feasible partition of $\tilde{V}$ such that $0 \in P_0$ and $n + i \in P_i$ for $i = 1, 2, \cdots, n$. Then it is not difficult to see that the cost $\sum_{i=0}^{n} C(P_i)$ is at most n if and only if $P_0 \setminus \{0\}$ is an independent set, and moreover, it is $k\,(\leq n)$ if and only if $V \setminus P_0$ is a vertex cover of the size k. This proves the claim. □

6. Concluding Remarks

In this paper, we have constructed a polynomial time algorithm for the tree partitioning problem with a monotone min-max cost function, and as a corollary we have shown that the evacuation problem can be solved in polynomial time. We have also shown that without the monotonicity the partitioning problem becomes NP-hard. Furthermore, we have shown that the min-sum tree partitioning problem is NP-hard, even if the cost function is monotone.

Acknowledgements

This research is partially supported by MEXT under Grant-in-Aid for Creative Scientific Research. We are grateful to the reviewers for their helpful and constructive comments which improved the presentation of this paper.

References

[1] J.E. Aronson: A survey of dynamic network flows. *Annals of Operations Research*, **20** (1989), 1-66.

[2] R. Becker and Y. Perl: Shifting algorithms for tree partitioning with general weighting functions. *Journal of algorithms*, **4** (1983), 101–120.

[3] C. Burstedde, A. Kirchner, K. Klauck, A. Schadschneider, and J. Zittartz: Cellular automaton approach to pedestrian dynamics — Applications. In: *Pedestrian and Evacuation Dynamics* (M. Schreckenberg and S.D. Sharma, eds., Springer, 2002), 87–97.

[4] M. Carey and E. Subrahmanian: An approach to modelling time-varying flows on congested networks. *Transportation Research*, B **34** (2000), 157–183.

[5] L.G. Chalmet, R.L. Francis, and P.B. Saunders: Network models for building evacuation. *Management Science*, **28** (1982), 86–105.

[6] H.K. Chen and C.F. Hsueh: A model and an algorithm for the dynamic user-optimal route choice problem. *Transportation Research*, B **32** (1998), 219–234.

[7] W. Choi, H.W. Hamacher, and S. Tufekci: Modeling of building evacuation problems with side constraints. *European Journal of Operations Research*, **35** (1988), 98–110.

[8] L. Fleischer and É. Tardos: Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, **23** (1998), 71-80.

[9] E.R. Galea: Simulating evacuation and circulation in planes, trains, buildings and ships using the EXODUS software. In: *Pedestrian and Evacuation Dynamics* (M. Schreckenberg and S.D. Sharma, eds., Springer, 2002), 203–225.

[10] M.R. Garey and D.S. Johnson: *Computers and Intractability* (Freeman, 1979).

[11] H.W. Hamacher and S.A.Tjandra: Mathematical modelling of evacuation problems: A state of the art, In: *Pedestrian and Evacuation Dynamics* (Springer, 2002), 227–266.

[12] B. Hoppe and É. Tardos: The quickest transshipment problem. *Mathematics of Operations Research*, **25** (2000), 36–62.

[13] B.N. Janson: Dynamic traffic assignment for urban road networks. *Transportation Research*, B **25** (1991), 143–161.

[14] S. Mamada, K. Makino, and S. Fujishige: Optimal sink location problem for dynamic flows in a tree network. *IEICE Trans. Fundamentals*, **E85-A** (2002), 1020–1025.

[15] S. Mamada, K. Makino, and S. Fujishige: An evacuation problem in tree dynamic networks with multiple exits. *Proceedings of International Symposium on Systems & Human Science for Safety, Security, and Dependability* (2003), 347–351.

[16] S. Mamada, T. Uno, K. Makino, and S. Fujishige: An $O(n \log^2 n)$ algorithm for a sink location problem in dynamic tree networks. *Discrete Applied Mathematics* (to appear).

[17] Y. Sheffi, H. Mahmassani, and W.B. Powell: A transportation network evacuation model. *Transportation Research*, A **16** (1982), 209–218.

[18] D.J. Zawack and G.L. Thompson: A dynamic space-time network flow model for city traffic congestion. *Transportation Science*, **21** (1987), 153–162.

Satoko Mamada
Division of Mathematical Science for Social Systems,
Graduate School of Engineering Science,
Osaka University,
Toyonaka, Osaka, 560-8531, Japan.
E-mail: mamada@inulab.sys.es.osaka-u.ac.jp

