PAPER Special Section on Foundations of Computer Science — Mathematical Foundations and Applications of Algorithms and Computer Science —

Minimum Spanning Tree Problem with Label Selection

Akio FUJIYOSHI[†a)] ***and*** **Masakazu SUZUKI**[††b)]**,** ***Members***

SUMMARY In this paper, we study the minimum spanning tree problem with label selection, that is, the problem of finding a minimum spanning tree of a vertex-labeled graph where the weight of each edge may vary depending on the selection of labels of vertices at both ends. The problem is especially important as the application to mathematical OCR. It is shown that the problem is NP-hard. However, for the application to mathematical OCR, it is sufficient to deal with only graphs with small tree-width. In this paper, a linear-time algorithm for series-parallel graphs is presented. Since the minimum spanning tree problem with label selection is closely related to the generalized minimum spanning tree problem, their relation is discussed.

key words: *minimum spanning tree problem, NP-hardness, series-parallel graph, mathematical OCR*

1. Introduction

The minimum spanning tree problem is one of the most famous combinatorial problems in computer science. Fast algorithms to solve the problem are well-known. In this paper, we study a generalization of the problem for vertex-labeled graphs, where the weight of each edge may vary depending on the selection of labels of vertices at both ends. We call this problem *the minimum spanning tree problem with label selection*.

Variations of the minimum spanning tree problem have been extensively studied. Myung, Lee and Tcha introduced the generalized minimum spanning tree problem (GMSTP) [1], where the vertices of a graph are partitioned into clusters and exactly one vertex from each cluster must be connected. It is known that GMSTP is NP-hard [1], and the problem is still NP-hard even on trees [2]. Thus, linear programming relaxations are considered [1]–[4]. Chang and Leu introduced the minimum labeling spanning tree problem (MLSTP) [5], where the edges of a graph are colored and the number of colors of a spanning tree should be minimized. MLSTP is also NP-hard, and the problem is still NP-hard even on complete graphs [5]. Consequently, heuristic algorithms [5]–[7] and restricted versions [8] of the problem are considered. Makino, Uno and Ibaraki introduced the minimum edge-ranking spanning tree problem (MERSTP) [9], that is, the problem of finding a spanning tree of a graph whose edge-ranking is minimum. MERSTP is also NP-hard [9], and the problem is still NP-hard even on series-parallel graphs [10].

An instance of the minimum spanning tree problem with label selection is illustrated in Fig. 1 (a): The set of vertex labels is $\Sigma = \{a, b, c, d\}$; vertices are indicated by dotted rectangles; each vertex has at least one label candidates represented by circled symbols; each weighted edge connects label candidates that belong to different vertices; and some pairs of label candidates may not be connected. For this instance of the problem, a minimum spanning tree is illustrated in Fig. 1 (b): Exactly one label is selected from candidates for each vertex; and the graph induced by selected

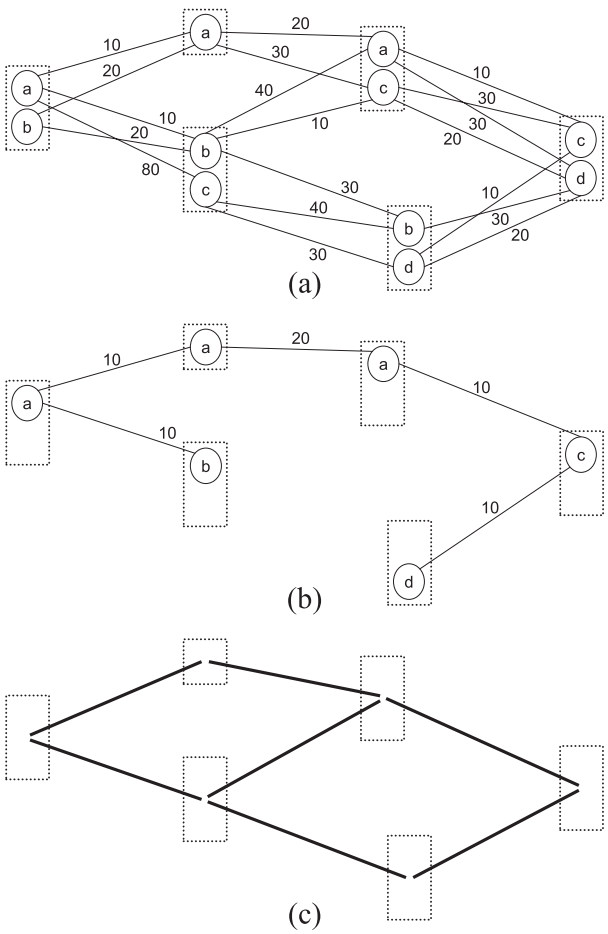

Fig. 1 (a) An instance of the minimum spanning tree problem with label selection, (b) the minimum spanning tree, and (c) the base graph.

Manuscript received April 1, 2010.
Manuscript revised August 1, 2010.

[†]The author is with the Department of Computer and Information Sciences, Ibaraki University, Hitachi-shi, 316–8511 Japan.
[††]The author is with the Graduate School of Mathematics, Kyushu University, Fukuoka-shi, 819–0395 Japan.
a) E-mail: fujiyosi@mx.ibaraki.ac.jp
b) E-mail: suzuki@math.kyushu-u.ac.jp

DOI: 10.1587/transinf.E94.D.233

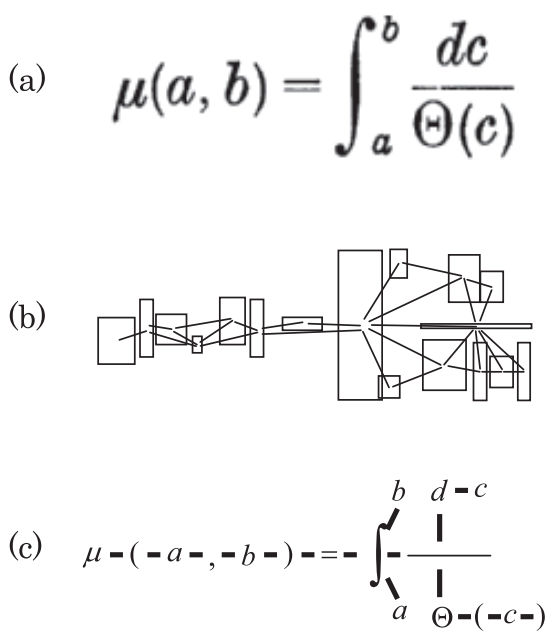

Fig. 2 (a) A scanned image, (b) the graph expressing possible adjacency connections of bounding boxes, (c) the correct recognition result.

label candidates and selected edges becomes a spanning tree where the sum of weights of edges is the minimum. We also introduce the notion of a base graph. The corresponding base graph is illustrated in Fig. 1 (c).

For the development of mathematical OCR [11]–[13], the problem is especially important. As shown in Fig. 2 (a) and (b), a mathematical OCR system constructs a graph that expresses possible adjacency connections of bounding boxes from a scanned image. Then, there may exist several candidates of character recognition for each bounding box because the precision of a current character recognition engine is not high enough, and a mathematical formula contains various kinds of symbols. For example, the left-most bounding box has plural character recognition candidates such as 'μ', 'u', 'v' and 'ρ'. Each edge is weighted by size, positional relation, and bigram statistics of character recognition candidates. In order to output a better recognition result as shown in Fig. 2 (c), the system should find a minimum spanning tree of the graph not only by selecting character recognition candidates for bounding boxes but also by determining adjacency connections of bounding boxes.

This paper shows that the minimum spanning tree problem with label selection is NP-hard. The NP-hardness is proved by reducing the Boolean satisfiability problem (SAT) to this problem. Therefore, it is difficult to solve the problem for the general case. However, by surveying adjacency connections of bounding boxes in mathematical images, we found that it is sufficient to deal with only graphs with small tree-width for the application to mathematical OCR.

In this paper, a linear-time algorithm for graphs whose base graph is a series-parallel graph (SPG) is presented. SPGs are typical graphs with tree-width at most 2 since it is known that a graph has tree-width at most 2 if and only if every biconnected component is an SPG. They are of interest in algorithmic graph theory because a number of standard problems on graphs are solvable in linear time for SPGs including some NP-complete problems [14].

Since the minimum spanning tree problem with label selection is closely related to GMSTP [1], we discuss their relation in this paper. The results of this paper can be translated into some new results for GMSTP.

This paper is organized as follows: In Sect. 2, some definition are given; in Sect. 3, the NP-hardness of the problem is proved; in Sect. 4, a linear-time recognition algorithm of series-parallel graphs is introduced; in Sect. 5, the relation to GMSTP is discussed; and in Sect. 6, the conclusion is drawn.

2. Preliminaries

In this section, we give some definitions and formally redefine the problem.

A *graph* is an ordered pair $G = (V, E)$, where V is a finite set, called *vertices*, and E is a finite set of unordered pairs of distinct vertices, called *edges*. A connected graph is a *tree* if it has no cycles. Unless explicitly stated, we assume that every graph is connected in this paper. A tree $T = (V', E')$ is a *spanning tree* of G if $V' = V$ and $E' \subseteq E$. Let Σ be a finite set of *vertex labels*. In this paper, we think that $|\Sigma|$ is a constant. Let R_+ be the set of positive real numbers. The *weight function* of G is a function $w : V \times V \times \Sigma \times \Sigma \to R_+ \cup \{\infty\}$ such that $w(v_1, v_2, l_1, l_2) = w(v_2, v_1, l_2, l_1)$ for any $v_1, v_2 \in V$ and $l_1, l_2 \in \Sigma$, and if $\{v_1, v_2\} \notin E$, then $w(v_1, v_2, l_1, l_2) = \infty$ for any $l_1, l_2 \in \Sigma$. A *vertex-labeling* of G is a function $\sigma : V \to \Sigma$. For a spanning tree $T = (V, E')$ and a vertex-labeling σ, the *weight* of T is defined as follows:

$$w(T) = \sum_{\{v_1,v_2\} \in E'} w(v_1, v_2, \sigma(v_1), \sigma(v_2)).$$

An *edge-selection* of G is a subset of E.

For a graph $G = (V, E)$ and its weight function w, *the minimum spanning tree problem with label selection* is the problem to find a vertex-labeling σ and an edge-selection E' such that $T = (V, E')$ is a spanning tree of G, and T has the minimum weight.

3. NP-Hardness

In this section, we will see the minimum spanning tree problem with label selection is NP-hard.

Theorem 1: The minimum spanning tree problem with label selection is NP-hard.

Proof. We will show the NP-hardness by reducing the Boolean satisfiability problem (SAT) to this problem.

Let $\mathcal{F}$ be a given Boolean formula in conjunctive normal form (CNF), where $C = \{c_1, \ldots, c_m\}$ is the set of clauses composing $\mathcal{F}$, and $\mathcal{X} = \{x_1, \ldots, x_n\}$ is the set of Boolean

variables appearing in $\mathcal{F}$.

The set of vertex labels is $\Sigma = \{T, F\}$. From $\mathcal{F}$, we construct a graph $G = (V, E)$ and its weight function w as follows: $V = \{x_1, \ldots, x_n\} \cup \{c_1, \ldots, c_m\}$, and $E = E_1 \cup E_2$, where $E_1 = \{\{x_i, x_{i+1}\} \mid 1 \le i \le n-1\}$ and $E_2 = \{\{x_i, c_j\} \mid x_i$ or $\bar{x}_i$ appears in c_j for $1 \le i \le n$ and $1 \le j \le m\}$. For all $\{x_i, x_{i+1}\} \in E_1$ and $l_1, l_2 \in \{T, F\}$, $w(x_i, x_{i+1}, l_1, l_2) = 1$, and for $\{x_i, c_j\} \in E_2$, if x_i appears in c_j, then $w(x_i, c_j, T, T) = 1$ and $w(x_i, c_j, F, T) = w(x_i, c_j, T, F) = w(x_i, c_j, F, F) = \infty$, or else if $\bar{x}_i$ appears in c_j, then $w(x_i, c_j, F, T) = 1$ and $w(x_i, c_j, T, T) = w(x_i, c_j, T, F) = w(x_i, c_j, F, F) = \infty$. This construction can be done in polynomial time.

For example, the graph corresponding to the CNF formula $(x_1 \vee \bar{x}_2 \vee x_5) \wedge (x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_3 \vee \bar{x}_4 \vee \bar{x}_5)$ is illustrated in Fig. 3 (a). Weighted edges with the infinite weight and label candidates whose all connecting weighted edges have the infinite weight are omitted. One of the minimum spanning trees of the graph is illustrated in Fig. 3 (b).

We will prove the following statement: *$\mathcal{F}$ has a truth assignment if and only if there exists a spanning tree of G with the weight $m + n - 1$.*

The 'only if' part is proved as follows. Suppose that $(x_1 = a_1, \ldots, x_n = a_n)$ is a truth assignment of $\mathcal{F}$, where $a_1, \ldots, a_n \in \{T, F\}$. Then, for each c_j, there is at least one variable in c_j which makes the clause true. Let $x_{c_1}, \ldots, x_{c_m}$ be such variables. If we set $E' = E_1 \cup \{\{x_{c_j}, c_j\} \mid 1 \le j \le m\}$ and $\sigma = \{(x_i, a_i) \mid 1 \le i \le n\} \cup \{(c_j, T) \mid 1 \le j \le m\}$, then

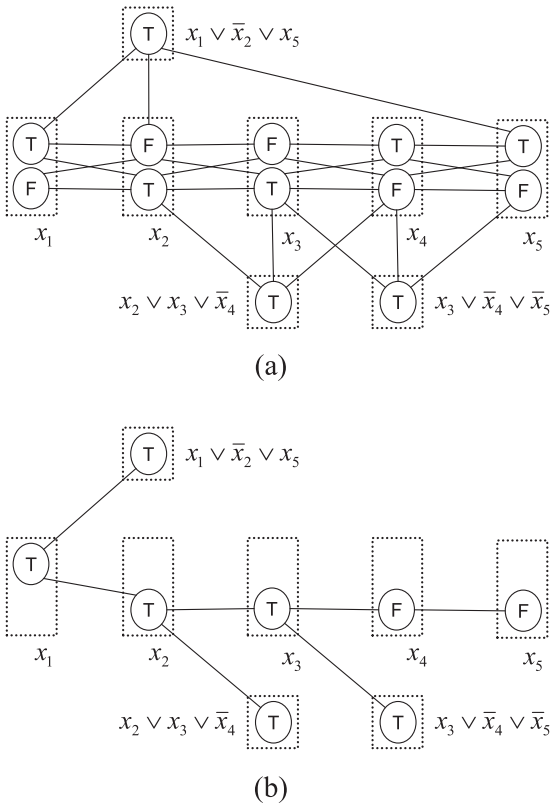

Fig. 3 (a) The graph corresponding to the formula, and (b) one of the minimum spanning trees of it.

$T = (V, E')$ is a spanning tree of G with the weight $m + n - 1$.

The 'if' part is proved as follows. Suppose that there exist a vertex-labeling σ and a spanning tree $T = (V, E')$ of G with the weight $m + n - 1$. If we set $a_i = \sigma(x_i)$ for each $1 \le i \le n$, then $(x_1 = a_1, \ldots, x_n = a_n)$ is a truth assignment of $\mathcal{F}$. □

4. Linear-Time Algorithm for Series-Parallel Graphs

In this section, we present a linear-time algorithm for series-parallel graphs (SPGs) [14]. For the application to mathematical OCR [11]–[13], the linear-time algorithm for SPGs is useful enough because, by surveying adjacency connections of bounding boxes in mathematical images, we found that it is sufficient to deal with only graphs with small tree-width, and SPGs are typical graphs with tree-width at most 2. They are of interest in algorithmic graph theory because a number of standard problems on graphs are solvable in linear time for SPGs including some NP-complete problems [14].

4.1 Series-Parallel Graphs

Let us write $G(s, t)$ to mean that the graph G has two distinguished vertices, namely, the source s and the sink t. A graph $G(s, t)$ is a *series-parallel graph* (SPG) if (1) it consists of a single edge connecting s and t, i.e., $G = (\{s, t\}, \{\{s, t\}\})$, or (2) it can be produced by a sequence of the following two operations:

Series Composition:
Given two series-parallel graphs $G_1(s_1, t_1)$ and $G_2(s_2, t_2)$, form a new graph $G(s, t)$ by identifying $s = s_1$, $t_1 = s_2$ and $t = t_2$.

Parallel Composition:
Given two series-parallel graphs $G_1(s_1, t_1)$ and $G_2(s_2, t_2)$, form a new graph $G(s, t)$ by identifying $s = s_1 = s_2$ and $t = t_1 = t_2$.

Due to the recursive definition of SPGs, we can obtain a vertex-labeled ordered tree corresponding to a decomposition of an SPG.

A *series-parallel tree* (SPT) T for an SPG $G(s, t) = (V, E)$ is a vertex-labeled ordered tree defined as follows: The set of vertex labels is $\{S, P\} \cup E$.

- If $G(s, t)$ consists of a single edge, then $T = (\{r\}, \emptyset)$ where r is a new vertex (the root of T), and the label of r is (s, t).
- If $G(s, t)$ is obtained by a series composition of $G_1(s_1, t_1)$ and $G_2(s_2, t_2)$, and $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ are SPTs of them, then $T = (\{r\} \cup V_1 \cup V_2, \{(r, r_1), (r, r_2)\} \cup E_1 \cup E_2)$ where r is a new vertex (the root of T), and r_1 and r_2 are the roots of T_1 and T_2, the label of r is S, the first child of r is r_1, and the second child of r is r_2.
- If $G(s, t)$ is obtained by a parallel composition of $G_1(s_1, t_1)$ and $G_2(s_2, t_2)$, and $T_1 = (V_1, E_1)$ and $T_2 =$

(V_2, E_2) are SPTs of them, then $T = (\{r\} \cup V_1 \cup V_2, \{(r, r_1), (r, r_2)\} \cup E_1 \cup E_2)$ where r is a new vertex (the root of T), and r_1 and r_2 are the roots of T_1 and T_2, the label of r is P, the first child of r is either r_1 or r_2, and the second child of r is the remaining one.

Note that all edges in E appear exactly once as a label of leaves. An SPG may have many corresponding SPTs since the above decomposition is not unique in general. It is known that an SPT is obtained from any SPG in linear time depending on the number of edges of an SPG [14].

4.2 Minimum Spanning Trees and Minimum Spanning Pair-Trees of Series-Parallel Graphs

In order to describe the idea behind the algorithm, we need the notion of a spanning pair-tree. For an SPG $G(s, t) = (V, E)$, a *spanning pair-tree* of G is an ordered pair of trees $(T_1 = (V_1, E_1), T_2 = (V_2, E_2))$ such that $s \in V_1$, $t \in V_2$, $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ and $E_1 \cup E_2 \subseteq E$. Its weight is defined as $w(T_1) + w(T_2)$.

We also introduce the notions of a terminal-fixed minimum spanning tree and a terminal-fixed minimum spanning pair-tree. Let $G(s, t) = (V, E)$ be an SPG, and let Σ be the set of vertex labels. For $a, b \in \Sigma$, a *terminal-fixed minimum spanning tree* (a, b)-MST of G is a spanning tree of G with weight less than or equal to the weight of the other spanning trees of G providing that $\sigma(s) = a$ and $\sigma(t) = b$, and likewise a *terminal-fixed minimum spanning pair-tree* (a, b)-MSPT of G is a spanning pair-tree of G with weight less than or equal to the weight of the other spanning pair-trees of G providing that $\sigma(s) = a$ and $\sigma(t) = b$.

The algorithm is based on the following lemma about a terminal-fixed minimum spanning tree and a terminal-fixed minimum spanning pair-tree of an SPG.

Lemma 4.1: For an SPG $G = (\{s, t\}, \{\{s, t\}\})$, the (a, b)-MST of G is $T = (\{s, t\}, \{\{s, t\}\})$ for any $a, b \in \Sigma$, and the (a, b)-MSPT of G is $(T_1 = (\{s\}, \emptyset), T_2 = (\{t\}, \emptyset))$ for any $a, b \in \Sigma$.

When an SPG G is obtained by a series composition of G_1 and G_2, (See also Fig. 4)

(1) an (a, b)-MST of G is obtained by a composition of an (a, c)-MST of G_1 and a (c, b)-MST of G_2 for some $c \in \Sigma$, and
(2) an (a, b)-MSPT of G is obtained by a composition of either an (a, c)-MSPT of G_1 and a (c, b)-MST of G_2 for some $c \in \Sigma$, or an (a, c)-MST of G_1 and a (c, b)-MSPT of G_2 for some $c \in \Sigma$.

On the other hand, when G is obtained by a parallel composition of G_1 and G_2, (See also Fig. 5)

(3) an (a, b)-MST of G is obtained by a composition of either an (a, b)-MST of G_1 and an (a, b)-MSPT of G_2, or an (a, b)-MSPT of G_1 and an (a, b)-MST of G_2, and
(4) an (a, b)-MSPT of G is obtained by a composition of an (a, b)-MSPT of G_1 and an (a, b)-MSPT of G_2.

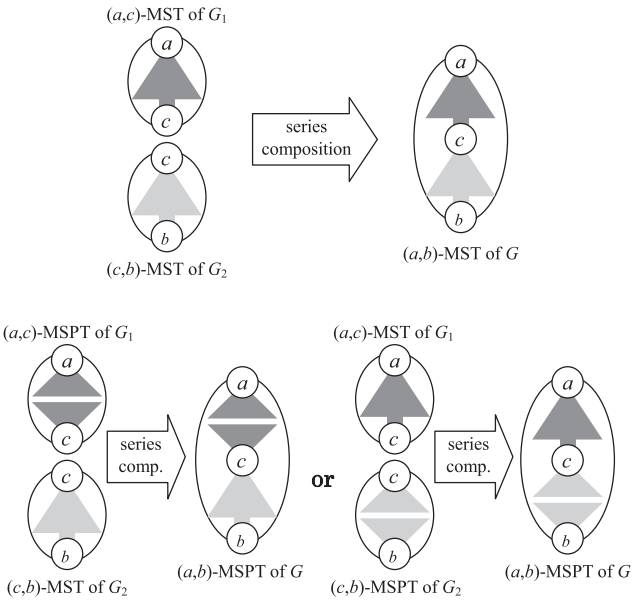

Fig. 4 An (a, b)-MST of G and an (a, b)-MSPT of G when G is obtained by a series composition of G_1 and G_2.

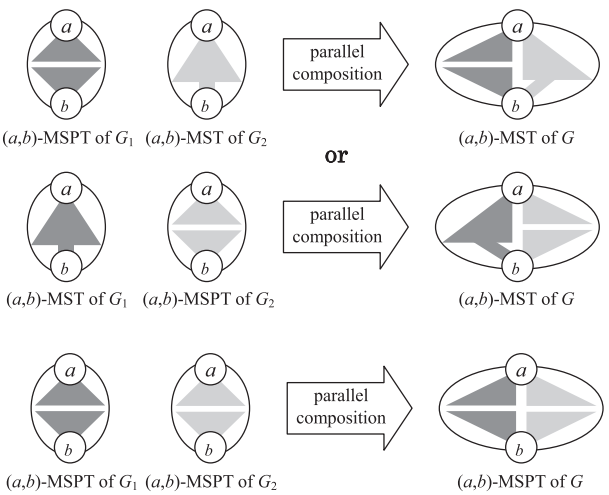

Fig. 5 An (a, b)-MST of G and an (a, b)-MSPT of G when G is obtained by a parallel composition of G_1 and G_2.

Proof. We will prove (1) by contradiction. Assume that T_1 is an (a, c)-MST of G_1 and T_2 is a (c, b)-MST of G_2 for some $c \in \Sigma$. Assume also that the sum of weights $w(T_1) + w(T_2)$ is less than or equal to the sum of weights of any (a, d)-MST of G_1 and any (d, b)-MST of G_2 for any $d \in \Sigma - \{c\}$. Let T be a spanning tree of G obtained by a composition of T_1 and T_2. Since $w(T) = w(T_1) + w(T_2)$, if T is not an (a, b)-MST of G, then there exists an (a, b)-MST of G whose weight is less than $w(T_1) + w(T_2)$, which contradicts the assumption.

The proofs of (2), (3) and (4) are similar. □

4.3 The Algorithm

Let $\Sigma = \{a_1, a_2, \ldots, a_m\}$ be the set of vertex labels. We think that $|\Sigma| = m$ is a constant.

The algorithm consists of the functions **Main** and **Calculate**. The function **Main** takes as input an SPG $G(s,t) = (V,E)$ and its weight function w, and returns the minimum weight of spanning trees of G. With the root vertex of T, the function **Calculate** returns two-dimensional arrays of real numbers A and B so that $A[i,j]$ stores the weight of (a_i,a_j)-MSTs of G, and $B[i,j]$ stores the weight of (a_i,a_j)-MSPTs of G. Thus, the minimum weight of spanning trees of G is the minimum value of $A[i,j]$ for $1 \le i \le m$ and $1 \le j \le m$.

Function Main
Input: an SPG $G(s,t) = (V,E)$ and its weight function w;
Output: the minimum weight of spanning trees of G;

```
1: Construct an SPT T = (V_T, E_T) corresponding to the SPG G;
2: Let u be the root vertex of T;
3: (A, B) := Calculate(u);
4: min = ∞
5: for i := 1 to m do
6:    for j := 1 to m do
7:       if A[i, j] < min then
8:          min := A[i, j];
9:       end if
10:   end for
11: end for
12: return min;
```

The function **Calculate** takes as input a vertex $u \in V_T$, and returns two-dimensional arrays of real numbers A and B. Let G' be the SPG corresponding to the subtree of T rooted at u. The arrays A and B are to store $A[i,j]$ with the weight of (a_i,a_j)-MSTs of G' and $B[i,j]$ with the weight of (a_i,a_j)-MSPTs of G'.

Function Calculate
Input: a vertex $u \in V_T$;
Output: arrays of real numbers $A[1\ldots m, 1\ldots m]$ and $B[1\ldots m, 1\ldots m]$;

```
1: if the label of u is (v_1, v_2) ∈ E then
2:    for i := 1 to m do
3:       for j := 1 to m do
4:          A[i, j] := w(v_1, v_2, i, j);
5:          B[i, j] := 0;
6:       end for
7:    end for
8: else if the label of u is S then
9:    Let u_1 and u_2 be the first and second child of u, respectively;
10:   (A_1, B_1) := Calculate(u_1);
11:   (A_2, B_2) := Calculate(u_2);
12:   for i := 1 to m do
13:      for j := 1 to m do
14:         minA = ∞;
15:         minB = ∞;
16:         for k := 1 to m do
17:            if A_1[i, k] + A_2[k, j] < minA then
18:               minA := A_1[i, k] + A_2[k, j];
19:            end if
20:            if A_1[i, k] + B_2[k, j] < minB then
21:               minB := A_1[i, k] + B_2[k, j];
22:            end if
23:            if B_1[i, k] + A_2[k, j] < minB then
24:               minB := B_1[i, k] + A_2[k, j];
25:            end if
26:         end for
27:         A[i, j] := minA;
28:         B[i, j] := minB;
29:      end for
30:   end for
31: else if the label of u is P then
32:   Let u_1 and u_2 be the children of u;
33:   (A_1, B_1) := Calculate(u_1);
34:   (A_2, B_2) := Calculate(u_2);
35:   for i := 1 to m do
36:      for j := 1 to m do
37:         if A_1[i, j] + B_2[i, j] < B_1[i, j] + A_2[i, j] then
38:            A[i, j] := A_1[i, j] + B_2[i, j];
39:         else
40:            A[i, j] := B_1[i, j] + A_2[i, j];
41:         end if
42:         B[i, j] := B_1[i, j] + B_2[i, j];
43:      end for
44:   end for
45: end if
46: return (A, B);
```

Theorem 2: The algorithm works correctly and terminates in linear time depending on the number of edges of G.

Proof. The correctness of the algorithm is clear from Lemma 4.1.

We will prove that the algorithm terminates in $O(|E|)$ time. Recall that it is known that we can obtain an SPT of $O(|E|)$ size in $O(|E|)$ time from any SPG. The total running time of the algorithm can be computed by counting the number of calls of the function **Calculate** and by evaluating the maximum running time for each call. The number of calls of the function **Calculate** is $|V_T|$ because the function **Calculate** is called exactly once for each $u \in V_T$. Clearly, $|V_T| = 2|E| - 1$. Recall that we think that $|\Sigma| = m$ is a constant. Since the construction of the arrays A and B does not depend on the size of input, we may evaluate that the construction of A and B is done in constant time. Therefore, the total running time is $O(|E|)$. □

5. Relation to the Generalized Minimum Spanning Tree Problem

In this section, the relation between the minimum spanning tree problem with label selection and the generalized minimum spanning tree problem [1] is discussed. We start introducing the problem.

A *graph with clusters* $V_1,\ldots,V_m$ is a graph $G = (V,E)$

where $V = V_1 \cup \cdots \cup V_m$ and $V_i \cap V_j = \emptyset$ for all $i, j \in \{1, \ldots, m\}$ such that $i \neq j$. The weight function of G is a function $w : E \to R_+$. A tree $T = (V', E')$ is a *generalized spanning tree* of G if $|V'| = m$, $|V' \cap V_i| = 1$ for all $1 \leq i \leq m$, and $E' \subseteq E$, i.e., V' contains exactly one vertex from each cluster.

For a graph $G = (V, E)$ with clusters $V_1, \ldots, V_m$ and its weight function w, *the generalized minimum spanning tree problem* (GMSTP) is the problem to find a generalized spanning tree $T = (V', E')$ of G which minimizes $\sum_{e \in E'} w(e)$. The ordinary minimum spanning tree problem is a special case of GMSTP where each cluster consists of exactly one node.

The minimum spanning tree problem with label selection is also a special case of GMSTP where the size of each cluster is bound by a constant number taking clusters as vertices and vertices as label candidates. On the other hand, the conversion from GMSTP to the minimum spanning tree problem with label selection is possible by taking the following notion[†]: For a graph $G = (V, E)$ with clusters $V_1, \ldots, V_m$, *the base graph* of G is the graph $G' = (\{V_1, \ldots, V_m\}, \{\{V_i, V_j\} \mid \exists v \in V_i, \exists v' \in V_j, \{v, v'\} \in E\})$.

The following results are known for GMSTP [2]:

- GMSTP is NP-hard, and the problem is still NP-hard even on trees.
- If the number of clusters is fixed, then GMSTP can be solved in polynomial-time with respect to the number of vertices.

The results of this paper can be translated into the following new results for GMSTP:

Theorem 3: GMSTP is still NP-hard even if the size of each cluster is at most 2.

Theorem 4: For a graph G with clusters, if the base graph of G is a series-parallel graph, then GMSTP for G can be solved in polynomial-time with respect to the number of vertices of G.

6. Conclusion

In order to improve structural analysis methods for mathematical OCR, we have studied the minimum spanning tree problem with label selection. Though the problem was shown to be NP-hard, we could obtain a linear-time algorithm for series-parallel graphs. Series-parallel graphs are typical graphs with tree-width at most 2. By surveying adjacency connections of bounding boxes in mathematical images, we found that it is sufficient to deal with graphs with tree-width at most 3. Therefore, the algorithm for series-parallel graphs should be extended by the time of the implementation of a new recognition engine for mathematical OCR.

Moreover, solving the minimum spanning tree problem with label selection itself is not enough for the practical application to mathematical OCR because a minimum spanning tree only reflects local weights defined on images. In other words, a minimum spanning tree may not be the best recognition result in many cases. Other factors should be considered such as total balance, n-gram statistics with $n \geq 3$, and grammatical adequacy of a recognition result. As future works, we want to study the extensions of the problem with those factors and give them practical solutions.

[†]When an instance of GMSTP does not have a feasible solution, the corresponding instance of the minimum spanning tree problem with label selection must have edges with the infinite weight, and all spanning trees must have the infinite weight.

References

[1] Y.S. Myung, C.H. Lee, and D.W. Tcha, "On the generalized minimum spanning tree problem," Networks, vol.26, no.4, pp.231–241, 1995.
[2] P.C. Pop, The generalized minimum spanning tree problem, Ph.D. thesis, Twente University Press, http://doc.utwente.nl/38643/, Dec. 2002.
[3] C. Feremans, M. Labbé, and G. Laporte, "The generalized minimum spanning tree: Polyhedra and branch-and-cut," 6th Twente Workshop on Graphs and Combinatorial Optimization, Electronic Notes in Discrete Mathematics, vol.3, pp.45–50, 1999.
[4] C. Feremans, M. Labbé, and G. Laporte, "A comparative analysis of several formulations for the generalized minimum spanning tree problem," Networks, vol.39, no.1, pp.29–34, 2002.
[5] R.S. Chang and S.J. Leu, "The minimum labeling spanning trees," Inf. Process. Lett., vol.63, no.5, pp.277–282, 1997.
[6] S.O. Krumke and H.C. Wirth, "On the minimum label spanning tree problem," Inf. Process. Lett., vol.66, no.2, pp.81–85, 1998.
[7] Y. Wan, G. Chert, and Y. Xu, "A note on the minimum label spanning tree," Inf. Process. Lett., vol.84, no.2, pp.99–101, 2002.
[8] T. Brügemann, J. Monnot, and G.J. Woeginger, "Local search for the minimum label spanning tree problem with bounded color classes," Oper. Res. Lett., vol.31, no.3, pp.195–201, 2003.
[9] K. Makino, Y. Uno, and T. Ibaraki, "On minimum edge ranking spanning trees," J. Algorithms, vol.38, no.2, pp.411–437, 2001.
[10] A.S. Arefin and M.A.K. Mia, "Np-completeness of the minimum edge-ranking spanning tree problem on series-parallel graphs," Proc. 10th International Conference on Computer and Information Technology (ICCIT 2007), 2007.
[11] K.F. Chan and D.Y. Yeung, "Mathematical expression recognition: A survey," Int. J. Document Analysis and Recoginition, vol.3, no.1, pp.3–15, 2000.
[12] Y. Eto and M. Suzuki, "Mathematical formula recognition using virtual link network," Proc. 6th International Conference on Document Analysis and Recognition (ICDAR 2001), pp.430–437, 2001.
[13] A. Fujiyoshi, M. Suzuki, and S. Uchida, "Verification of mathematical formulae based on a combination of context-free grammar and tree grammar," Proc. 7th International Conference on Mathematical Knowledge Management (MKM 2008), pp.415–429, LNCS (LNAI) 5144, 2008.
[14] K. Takamizawa, T. Nishizeki, and N. Saito, "Linear-time computability of combinatorial problems on series-parallel graphs," J. ACM, vol.29, no.3, pp.623–641, 1982.

Akio Fujiyoshi was born in Tokyo, Japan in 1971. He received the B.E., M.E., and Dr. Sci. degrees from the University of Electro-Communications, Tokyo, Japan, in 1995, 1997, and 2000, respectively. He is presently a lecturer in the Department of Computer and Information Sciences, Ibaraki University. His main interests are formal language theory and algorithmic learning theory.

Masakazu Suzuki received B.Sci. and M.Sci. degrees from Kyoto University in 1969 and 1971 respectively and degree of D.d'Eta ès Sci. at Univ. Paris VII in 1977. During his career in CNRS from 1975 to 1977 and in Kyushu University from 1977, his main research subjects were complex analysis and algebraic geometry. He is currently a professor at Graduate School of Mathematics, Kyushu University. In recent years, his research interests include mathematical document recognition and mathematical knowledge management. Dr. Suzuki is a member of MSJ and IPSJ.