数理解析研究所講究録
第 1691 巻 2010 年 102-107

ラベル選択を有する最小全域木問題

藤芳 明生
茨城大学工学部情報工学科
fujiyosi@mx.ibaraki.ac.jp

鈴木 昌和
九州大学大学院数理学研究院
suzuki@math.kyushu-u.ac.jp

Abstract

In this paper, we study the minimum spanning tree problem for vertex-labeled graphs, where the weights of edges 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, there exists a linear-time algorithm for graphs of small tree-width. In this paper, a linear-time algorithm for series-parallel graphs is presented. The relation to the generalized minimum spanning tree problem is discussed.

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 edges may vary depending on the selection of labels of vertices at both ends.

An instance of the problem is illustrated in Fig. 1 (a): a finite set of vertex labels is $\Sigma = \{a, b, c, d\}$; vertices are indicated by dotted rectangles; each vertex has at least one candidates of labels represented by circled symbols; each weighted edge connects labels that belong to different vertices; and some pairs of labels may not be connected. For this instance of the problem, the 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 labels 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). We say that two vertices are connected if some candidates of labels of two vertices are connected.

For the development of mathematical OCR [1], 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. At this moment, several character recognition candidates may remain for each bounding box. Each edge is weighted by the positional relation and co-occurrence of character recognition candidates. In order to output a better recognition result as shown in Fig. 2 (c), the system wants to find the minimum spanning tree from the graph not only by selecting character recognition candidates for bounding boxes but also by determining adjacency connections of bounding boxes.

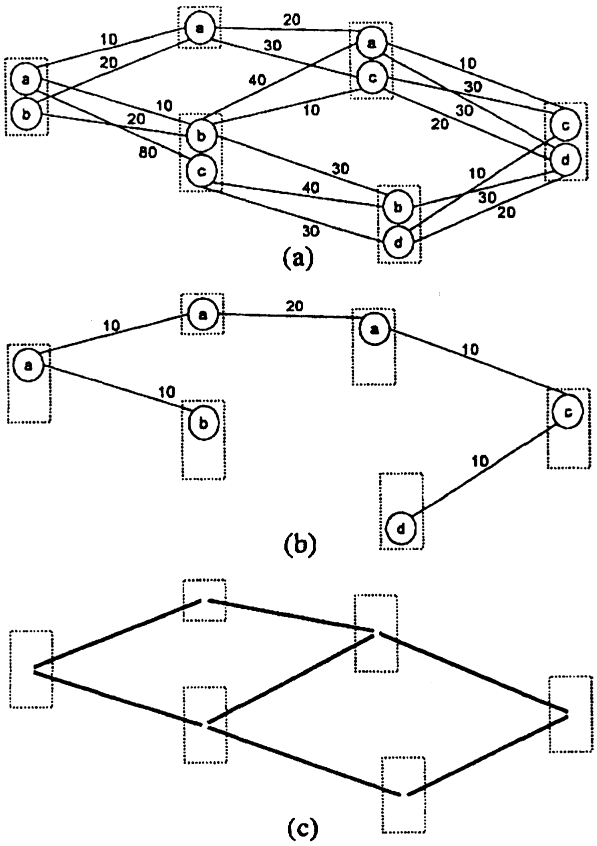

Figure 1: (a) an instance of the minimum spanning tree problem with label selection, (b) the minimum spanning tree, and (c) the base graph.

2 NP-Hardness

To show the NP-hardness, we reduce the Boolean satisfiability problem (SAT) to this problem.

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

Sketch of proof. Given a CNF formula, we can construct a graph such that it has a spanning tree if and only if the formula has a truth assignment. 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. All edges are weighted by the unit value. □

3 Linear-Time Algorithm

In this section, we present a linear-time algorithm for graphs whose base graph is a series-parallel graph [4]. The idea behind the algorithm is described as follows:

- When two graphs are connected in series, the minimum spanning tree is obtained by simply connecting the two minimum spanning trees of original graphs.

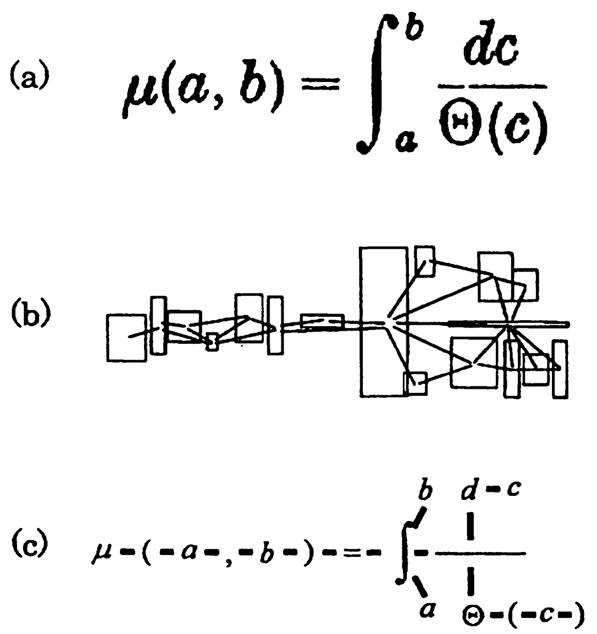

Figure 2: (a) a scanned image, (b) the graph expressing possible adjacency connections of bounding boxes, (c) the correct recognition result.

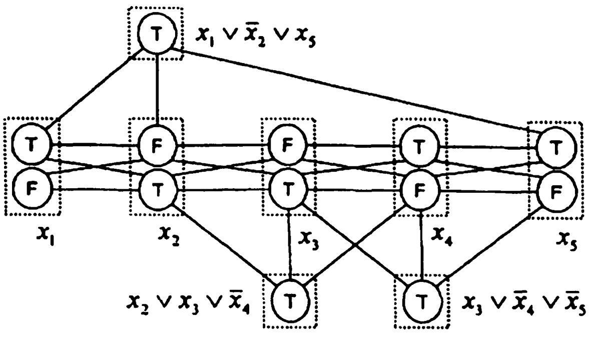
T
$x_1 \vee \bar{x}_2 \vee x_5$
T
F
F
T
F
T
T
F
T
F
x_1
x_2
x_3
x_4
x_5
$x_2 \vee x_3 \vee \bar{x}_4$
T
T
$x_3 \vee \bar{x}_4 \vee \bar{x}_5$

Figure 3: the DAG corresponding to the formula.

- When two graphs are connected in parallel, the minimum spanning tree is obtained by connecting the minimum spanning tree of one original graph and the minimum two disconnected spanning subtrees of the other graph.

Let us write $G(s,t)$ to mean that the graph G has two distinguished vertices, namely, the source s and the sink t.

Definition 1 A graph is a *series-parallel graph (SPG)* if (1) it consists of a single edge connecting s and 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 an vertex-labeled ordered tree in accordance with a decomposition of an SPG.

Definition 2 A *series-parallel tree (SPT)* T for an SPG $G(s, t) = (V, E)$ is an edge-labeled rooted 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 order of children of r is ignored.

Note that the children of a vertex labeled S are ordered, while the children of a vertex labeled P are unordered. All edges of D 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 [4].

Let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ be the set of vertex lables. Note that the number of vertex labels m is a constant number.

The algorithm consists of the following functions **Main** and **Calculate**:

```
Function Main
Input: an SPG $G(s,t) = (V, E)$ and a corresponding SPT $T = (V_T, E_T)$;
Output: the minimum weight of spanning trees of $G$;
Let $u$ be the root vertex of $T$;
$(A, B)$ := Calculate($u$);
$min = \infty$
for $i := 1$ to $m$ do
    for $j := 1$ to $m$ do
        if $A[i, j] < min$ then
            $min := A[i, j]$;
        end if
    end for
end for
return $min$;
```

```
Function Calculate
Input: a vertex $u \in V_T$;
```

```
Output: arrays of real numbers $A[1 \ldots m, 1 \ldots m]$ and $B[1 \ldots m, 1 \ldots m]$;
if the label of $u$ is $(v_1, v_2) \in E$ then
  for $i := 1$ to $m$ do
    for $j := 1$ to $m$ do
      if there extsts an weighted edge $e$ between the lable candidate $\sigma_i$ of $v_1$ and the
      lable candidate $\sigma_j$ of $v_2$ then
        $A[i, j]$ := the weight of $e$;
      else
        $A[i, j] := \infty$;
      end if
      $B[i, j] := 0$;
    end for
  end for
else if the label of $u$ is $S$ then
  Let $u_1$ and $u_2$ be the first and second child of $u$, respectively;
  $(A_1, B_1)$ := Calculate($u_1$);
  $(A_2, B_2)$ := Calculate($u_2$);
  for $i := 1$ to $m$ do
    for $j := 1$ to $m$ do
      $minA = \infty$;
      $minB = \infty$;
      for $k := 1$ to $m$ do
        if $A_1[i, k] + A_2[k, j] < minA$ then
          $minA := A_1[i, k] + A_2[k, j]$;
        end if
        if $A_1[i, k] + B_2[k, j] < minB$ then
          $minB := A_1[i, k] + B_2[k, j]$;
        end if
        if $B_1[i, k] + A_2[k, j] < minB$ then
          $minB := B_1[i, k] + A_2[k, j]$;
        end if
      end for
      $A[i, j] := minA$;
      $B[i, j] := minB$;
    end for
  end for
else if the lable of $u$ is $P$ then
  Let $u_1$ and $u_2$ be the children of $u$;
  $(A_1, B_1)$ := Calculate($u_1$);
  $(A_2, B_2)$ := Calculate($u_2$);
  for $i := 1$ to $m$ do
    for $j := 1$ to $m$ do
      if $A_1[i, j] + B_2[i, j] < B_1[i, j] + A_2[i, j]$ then
        $A[i, j] := A_1[i, j] + B_2[i, j]$;
      else
        $A[i, j] := B_1[i, j] + A_2[i, j]$;
```

```
            end if
            $B[i,j] := B_1[i,j] + B_2[i,j]$;
        end for
    end for
end if
return $(A,B)$;
```

4 Relation to the Generalized Minimum Spanning Tree Problem

The minimum spanning tree problem with label selection is closely related to the generalized minimum spanning tree problem (GMSTP) [2, 3].

The following results are known for GMSTP [3]:

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

- GMSTP is still NP-hard even if the size of each cluster is at most 2.
- If the tree-width of the base graph is small, then GMSTP can be solved in polynomial-time with respect to the number of vertices.

References

[1] Yuko Eto and Masakazu Suzuki. Mathematical formula recognition using virtual link network. In *Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR 2001)*, pp. 430–437, 2001.

[2] Young-Soo Myung, Chang-Ho Lee, and Dong-Wan Tcha. On the generalized minimum spanning tree problem. *Networks*, Vol. 26, No. 4, pp. 231–241, 1995.

[3] Petrica Claudiu Pop. *The generalized minimum spanning tree problem.* PhD thesis, Twente University Press, `http://doc.utwente.nl/38643/`, December 2002.

[4] Kazuhiko Takamizawa, Takao Nishizeki, and Nobuji Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *Journal of the ACM*, Vol. 29, No. 3, pp. 623–641, 1982.