PAPER *Special Section on Foundations of Computer Science*

Polynomial Time Inductive Inference of TTSP Graph Languages from Positive Data

Ryoji TAKAMI[†], **Yusuke SUZUKI**[††a)], ***Nonmembers***, **Tomoyuki UCHIDA**[††], ***and*** **Takayoshi SHOUDAI**[†††], ***Members***

SUMMARY Two-Terminal Series Parallel (TTSP, for short) graphs are used as data models in applications for electric networks and scheduling problems. We propose a TTSP term graph which is a TTSP graph having structured variables, that is, a graph pattern over a TTSP graph. Let $\mathcal{TG}_{\mathcal{TTSP}}$ be the set of all TTSP term graphs whose variable labels are mutually distinct. For a TTSP term graph g in $\mathcal{TG}_{\mathcal{TTSP}}$, the TTSP graph language of g, denoted by $L(g)$, is the set of all TTSP graphs obtained from g by substituting arbitrary TTSP graphs for all variables in g. Firstly, when a TTSP graph G and a TTSP term graph g are given as inputs, we present a polynomial time matching algorithm which decides whether or not $L(g)$ contains G. The minimal language problem for the class $\mathcal{L}_{\mathcal{TTSP}} = \{L(g) \mid g \in \mathcal{TG}_{\mathcal{TTSP}}\}$ is, given a set S of TTSP graphs, to find a TTSP term graph g in $\mathcal{TG}_{\mathcal{TTSP}}$ such that $L(g)$ is minimal among all TTSP graph languages which contain all TTSP graphs in S. Secondly, we give a polynomial time algorithm for solving the minimal language problem for $\mathcal{L}_{\mathcal{TTSP}}$. Finally, we show that $\mathcal{L}_{\mathcal{TTSP}}$ is polynomial time inductively inferable from positive data.

key words: *inductive inference, computational learning theory, TTSP graph, graph languages*

1. Introduction

We consider the learnability of Two-Terminal Series Parallel (TTSP, for short) graph languages from positive data. A TTSP graph is a connected directed graph constructed by recursively applying "series" and "parallel" operations. Since a TTSP graph is an edge-colored planar graph allowed multiple edges but having no cycles, it is often used in applications as a data model of electrical networks and scheduling. In the fields of data mining and knowledge discovery, many researchers have been developing data mining techniques based on machine learning methods for analyzing data. From the viewpoint of algorithmic learning theory, the purpose of this paper is to show that the class of languages on TTSP graphs is polynomial time inductively inferable from positive data.

Uchida et al. [11] proposed the concepts of a graph pattern having graph structures and structured variables, called a *term graph*, and a graph pattern language, called a *graph language*. Based on the concept of a term graph and a graph language given in [11], we define a *TTSP term graph* as a graph pattern having a TTSP graph structure and structured variables. A variable in a TTSP term graph is a pair of its distinct vertices. For a TTSP term graph g, we also define a *TTSP graph language* of g, denoted by $L(g)$, as the set of all TTSP graphs obtained from g by substituting arbitrary TTSP graphs for all variables in g. We denote by $\mathcal{TG}_{\mathcal{TTSP}}$ the set of all TTSP term graphs whose variable labels are mutually distinct. In Fig. 1, we give TTSP graphs G_1, G_2, G_3, G_4, G_5 and a TTSP term graph g in $\mathcal{TG}_{\mathcal{TTSP}}$ as examples. And the TTSP graph G_1 in Fig. 1 is obtained from the TTSP term graph g by replacing variables having label x and y with TTSP graphs G_4 and G_5 so that u_1, u_2, v_1, v_2 of g are identified with w_1^4, w_2^4, w_1^5, w_2^5 of TTSP graphs G_4 and G_5, respectively.

Angluin [2] and Shinohara [7] gave the framework of inductive inference from positive data and showed that if a class C has finite thickness, and the membership problem and the minimal language (MINL) problem for C are computable in polynomial time then C is polynomial time inductively inferable from positive data. Based on this framework, in this paper, we consider the polynomial time learnability of $\mathcal{L}_{\mathcal{TTSP}} = \{L(g) \mid g \in \mathcal{TG}_{\mathcal{TTSP}}\}$ from positive data.

Firstly, we show that, for any nonempty set S of TTSP graphs, the cardinality of the set $\{L \in \mathcal{L}_{\mathcal{TTSP}} \mid S \subseteq L\}$ is fi-

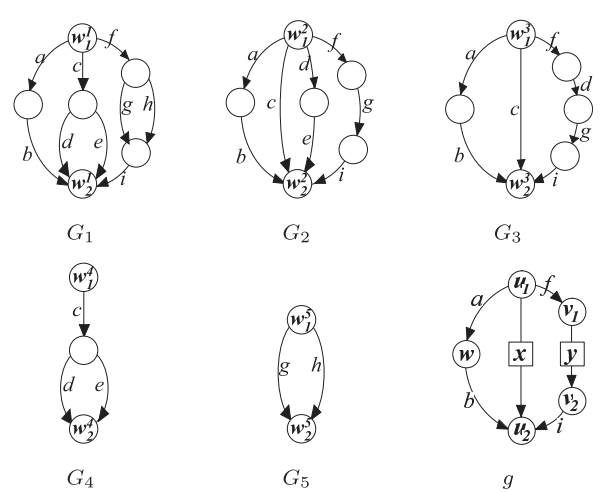

Fig. 1 TTSP graphs G_1, G_2, G_3, G_4, G_5 and a TTSP term graph g. A variable is drawn by a box with lines to its elements and the symbol inside a box shows the label of the variable.

Manuscript received March 28, 2008.
Manuscript revised June 29, 2008.

[†]The author is with the Faculty of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
[††]The authors are with the Graduate School of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
[†††]The author is with Department of Informatics, Kyushu University, Fukuoka-shi, 819–0395 Japan.
a) E-mail: y-suzuki@hiroshima-cu.ac.jp

DOI: 10.1587/transinf.E92.D.181

nite, that is, $\mathcal{L}_{\mathcal{TTSP}}$ has finite thickness. Secondly, we consider the membership problem for $\mathcal{L}_{\mathcal{TTSP}}$ which is, given a TTSP term graph $g \in \mathcal{TG}_{\mathcal{TTSP}}$ and a TTSP graph G, to decide whether or not $L(g)$ contains G. In [5], [8], and [9], we presented a term tree all of whose internal vertices are ordered, called an *ordered term tree*, and a term tree all of whose internal vertices are unordered, called an *unordered term tree*. In order to show that the membership problem for $\mathcal{L}_{\mathcal{TTSP}}$ is solvable in polynomial time, we present a polynomial time matching algorithm for solving the membership problem for the set of languages of term trees each of whose internal vertices has ordered or unordered children, by modifying polynomial time matching algorithms for ordered term trees and unordered term trees in [5] and [9]. In Fig. 2, we give a term tree t as an example. A tree T in Fig. 2 is obtained from t by replacing two variables having labels x and y with trees T_1 and T_2 in Fig. 2, respectively. Thirdly, we consider the minimal language problem, MINL problem for short, for $\mathcal{L}_{\mathcal{TTSP}}$ which is, given a set S of TTSP graphs, to find a TTSP term graph $g \in \mathcal{TG}_{\mathcal{TTSP}}$ such that $S \subseteq L(g)$ and there exists no TTSP term graph $g' \in \mathcal{TG}_{\mathcal{TTSP}}$ with $S \subseteq L(g') \subsetneq L(g)$, that is, $L(g)$ is minimal in the set $\{L \in \mathcal{L}_{\mathcal{TTSP}} \mid S \subseteq L\}$. Here g is called a *minimally generalized TTSP term graph* explaining S. For example, the TTSP graph g in Fig. 1 is a minimally generalized TTSP term graph such that $\{G_1, G_2, G_3\} \subseteq L(g)$, where G_1, G_2, G_3 are TTSP graphs in Fig. 1. Finally, we show that the class $\mathcal{L}_{\mathcal{TTSP}}$ is polynomial time inductively inferable from positive data.

There are many studies [10], [12], [13] for many graph theoretical problems on TTSP graphs such as *Recognition*, *Decomposition*, *Maximum independent set*, *Minimum dominating set*, *Maximum matching*. We considered the polynomial time learnabilities of ordered term tree languages and unordered term tree languages from positive data in [5], [8], and [9]. In other learning models such as query learning, the learnability of the class of finite unions of ordered term tree languages, unordered term tree languages and TTSP term graph languages were considered in [4] and [6].

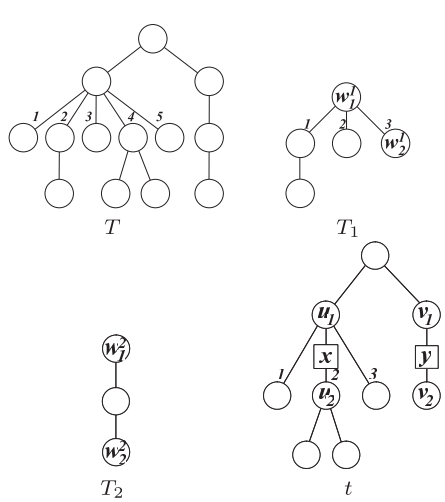

Fig. 2 Trees T, T_1, T_2 and a term tree t.

This paper is organized as follows. In Sect. 2, we give a TTSP graph pattern having structured variables and its graph language. Then, we formally define a membership problem and a minimal language problem for the class $\mathcal{L}_{\mathcal{TTSP}}$ of TTSP graph languages. In Sect. 3, we summarize our results of this paper. Then, we present polynomial time algorithms for solving the membership problem and the MINL problem for $\mathcal{L}_{\mathcal{TTSP}}$ in Sects. 4 and 5, respectively. In Sect. 6, we conclude this paper with future works.

2. Preliminaries

In this section, we present a two-terminal series parallel (TTSP, for short) term graph as a graph pattern having structured variables and its TTSP graph language by restricting the notion of a term graph given in [11]. For a set S, $|S|$ denotes the cardinality of S, that is the number of elements of S.

A *multidag* is a directed connected graph which allows multiple edges and does not contain any cycle. Let Λ be finite alphabet whose element is called an *edge label*. A directed edge labeled with an edge label is called an *edge*. An edge labeled with an edge label $a \in \Lambda$ from a vertex u to a vertex v is denoted by a triplet (u, a, v). Let $\mathcal{X}$ be infinite alphabet with $\Lambda \cap \mathcal{X} = \emptyset$. An element of $\mathcal{X}$ is called a *variable label*. A directed edge having a variable label is called a *variable*. A variable labeled with a variable label $x \in \mathcal{X}$ from a vertex u to a vertex v is denoted by a triplet $[u, x, v]$. For a vertex v in a TTSP term graph g, $indeg(v)$ denotes the sum of all edges entering v and all variables whose child port is v, and $outdeg(v)$ denotes the sum of all edges leaving v and all variables whose parent port is v. A vertex v with $indeg(v) = 0$ (resp., $outdeg(v) = 0$) is called a *source* (resp., a *sink*) of g. A multidag is said to be *two-terminal* if it has exactly one source and one sink.

Definition 1: A TTSP term graph is a two-terminal multidag recursively defined as follows.

(1) A directed connected graph consisting of two vertices u and v, and a single edge from u to v or a single variable is a TTSP term graph. The vertices u and v are its source and its sink, respectively.
(2) For $i = 1, 2$, let G_i be a TTSP term graph which has u_i as its source and v_i as its sink. Then the graph obtained by either of the following two operations is a TTSP term graph.
    (a) *Parallel operation*: Identify u_1 with u_2, and identify v_1 with v_2. The resulting graph, denoted by $G_1//G_2$, has $u_1(= u_2)$ as its source and $v_1(= v_2)$ as its sink.
    (b) *Series operation*: Identify u_2 with v_1. The source and the sink of the resulting graph, denoted by $G_1 * G_2$, are u_1 and v_2, respectively.

From the above definition, it is easy to see that a TTSP term graph has exactly one source and one sink. A TTSP term

graph g is denoted by a triplet $(V_g,\ E_g,\ H_g)$, where V_g, E_g and H_g are sets of all vertices, all edges and all variables in g, respectively. A TTSP term graph g is said to be *ground* if g has no variable. A ground TTSP term graph is called a TTSP graph simply. A TTSP graph g is denoted by $(V_g,\ E_g)$, where V_g and E_g are sets of all vertices and all edges in g, respectively. A TTSP term graph g is *linear* if all variables in g have mutually distinct variable labels in $\mathcal{X}$. The set of all linear TTSP term graphs is denoted by $\mathcal{TG}_{\mathcal{TTSP}}$ and the set of all ground TTSP term graphs, namely all TTSP graphs, is denoted by $\mathcal{TTSP}$. From the definitions of a TTSP term graph and a TTSP graph, we note that $\mathcal{TTSP} \subset \mathcal{TG}_{\mathcal{TTSP}}$ and every vertex has no label. In this paper, we deal with linear TTSP term graphs only. Hence, unless otherwise indicated, we call a linear TTSP term graph a TTSP term graph simply. For example, g given in Fig. 1 is a TTSP term graph $(\{u_1, u_2, v_1, v_2, w\}, \{(u_1, a, w), (w, b, u_2), (u_1, f, v_1), (v_2, i, u_2)\}, \{[u_1, x, u_2], [v_1, y, v_2]\})$ having the source u_1 and the sink u_2. For each i $(1 \le i \le 5)$, G_i given in Fig. 1 is a TTSP graph having w_1^i as the source and w_2^i as the sink, respectively. For two TTSP graphs G_4 and G_5 given in Fig. 1, we give an example of a TTSP graph $G_4//G_5$ (resp., $G_4 * G_5$) in Fig. 3 which is obtained by applying the Parallel operation (resp., the Series operation) to G_4 and G_5.

For an edge $(u, a, v) \in E_g$, u is said to be the *parent* of v and v is a *child* of u. For a variable $[u, x, v] \in H_g$, we call u the *parent port* of $[u, x, v]$ and v the *child port* of $[u, x, v]$. We call a sequence $v_1, v_2, \ldots, v_i$ of distinct vertices of g a *path* from v_1 to v_i if for any j with $1 \le j < i$, there exists an edge or a variable which consists of v_j and v_{j+1}.

For two TTSP term graphs $g = (V_g, E_g, H_g)$ and $f = (V_f, E_f, H_f)$, g and f are *isomorphic*, denoted by $g \equiv f$, if there exists a bijection $\pi : V_g \to V_f$ such that (i) the vertices u and v are the source and the sink of g, respectively, if and only if the vertices $\pi(u)$ and $\pi(v)$ are the source and the sink of f, respectively, (ii) $(u, a, v) \in E_g$ if and only if $(\pi(u), a, \pi(v)) \in E_f$, (iii) $[u, x, v] \in H_g$ if and only if $[\pi(u), y, \pi(v)] \in H_f$ for some x and y in $\mathcal{X}$.

Let g be a TTSP term graph with at least two vertices and x a variable label in $\mathcal{X}$. Let $\sigma = [u, u']$ be a list of two vertices in g, where u is the source of g and u' is the sink of g. The form $x := [g, \sigma]$ is called a *binding* for x. A *substitution* is a finite collection of bindings $\{x_1 := [g_1, \sigma_1], \cdots, x_n := [g_n, \sigma_n]\}$, where x_i's are mutually distinct variable labels in $\mathcal{X}$, g_i's are ground TTSP term graphs, and σ_i's are lists of two vertices in g_i.

Let $f = (V_f, E_f, H_f)$ and $g = (V_g, E_g, H_g)$ be two TTSP term graphs. A new TTSP term graph $f\{x := [g, [u, u']]\}$ is obtained by applying the binding $x := [g, [u, u']]$ to f in the following way. Let $e = [v, x, v']$ be a variable in f. Let g' be one copy of g and w, w' the vertices of g' corresponding to u, u' of g, respectively. For the variable $e = [v, x, v']$, we attach g' to f by removing the variable e from H_f and by identifying the vertices v, v' with the vertices w, w' of g', respectively. Now let $\theta = \{x_1 := [g_1, \sigma_1], \cdots, x_n := [g_n, \sigma_n]\}$ be a substitution. The TTSP term graph $f\theta$, called the *instance* of f by θ, is obtained by applying all the bindings $x_i := [g_i, \sigma_i]$ to f simultaneously. We remark that the source and the sink of f are the source and the sink of $f\theta$, respectively. For example, let g be a TTSP term graph in Fig. 1 and $\theta = \{x := [G_4, [w_1^4, w_2^4]], y := [G_5, [w_1^5, w_2^5]]\}$ a substitution, where G_4 and G_5 are TTSP graphs in Fig. 1. Then the instance $g\theta$ of the term graph g by θ is the TTSP graph G_1 in Fig. 1.

Definition 2: For a TTSP term graph $g \in \mathcal{TG}_{\mathcal{TTSP}}$, the *TTSP graph language* of g, denoted by $L(g)$, is defined as $\{G \in \mathcal{TTSP} \mid G \equiv g\theta \ \text{ for some substitution } \theta\}$.

3. Main Results

In this section, we formally define a membership problem and a minimal language problem for TTSP graph languages. Then, we summarize our results of this paper. In Sects. 4 and 5, we will discuss the membership problem and the minimal language problem for a TTSP graph language in detail.

For a class C, Angluin [1] and Shinohara [7] showed that if C has finite thickness, and the membership problem and the minimal language problem for C are solvable in polynomial time then C is polynomial time inductively inferable from positive data. In this paper, we consider the class $\mathcal{L}_{\mathcal{TTSP}} = \{L(g) \mid g \in \mathcal{TG}_{\mathcal{TTSP}}\}$ as a target of inductive inference.

It is easy to see that the following lemma holds, that is, for any nonempty finite set $S \subseteq \mathcal{TTSP}$, the cardinality of the set $\{L \in \mathcal{L}_{\mathcal{TTSP}} \mid S \subseteq L\}$ is finite.

Lemma 1: The class $\mathcal{L}_{\mathcal{TTSP}}$ has finite thickness.

Proof. Let S be a nonempty finite subset of $\mathcal{TTSP}$ and $G = (V_G, E_G)$ a TTSP graph in S. If $g = (V_g, E_g, H_g)$ is a TTSP term graph in $\mathcal{TG}_{\mathcal{TTSP}}$ such that $L(g)$ includes G, then $|V_g| \le |V_G|$ and $|E_g| + |H_g| \le |E_G|$. Moreover, the number of all edge labels in G is finite. Therefore $\mathcal{L}_{\mathcal{TTSP}}$ has finite thickness. □

In Sect. 4, by presenting a polynomial time matching algorithm for solving the membership problem for the set of languages of term trees each of whose internal vertices has ordered or unordered children, we show that the following membership problem for $\mathcal{L}_{\mathcal{TTSP}}$ is solvable in polynomial

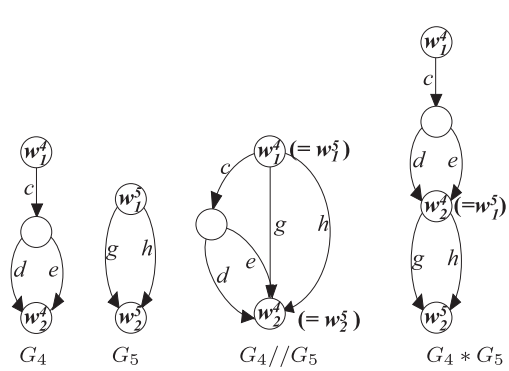

Fig. 3 TTSP graphs $G_4//G_5$ and $G_4 * G_5$, where G_4 and G_5 are given in Fig. 1.

time.

Membership Problem for $\mathcal{L}_{\mathcal{TTSP}}$.
Instance: A TTSP term graph $g \in \mathcal{TG}_{\mathcal{TTSP}}$ and a TTSP graph $G \in \mathcal{TTSP}$.
Question: Does $L(g)$ contain G?

A *minimally generalized TTSP term graph* explaining a given set of TTSP graphs $S \subseteq \mathcal{TTSP}$ is a TTSP term graph g such that $S \subseteq L(g)$ and there is no TTSP term graph g' satisfying that $S \subseteq L(g') \subsetneq L(g)$. In Sect. 5, we show that the following minimal language (MINL, for short) problem for $\mathcal{L}_{\mathcal{TTSP}}$ is solvable in polynomial time.

MINL Problem for $\mathcal{L}_{\mathcal{TTSP}}$.
Instance: A nonempty set of TTSP graphs $S \subseteq \mathcal{TTSP}$.
Question: Find a minimally generalized TTSP term graph $g \in \mathcal{TG}_{\mathcal{TTSP}}$ explaining S.

Therefore, we have the following main result.

Theorem 1: The class $\mathcal{L}_{\mathcal{TTSP}}$ is polynomial time inductively inferable from positive data.

4. An Efficient Matching Algorithm for TTSP Term Graphs

In this section, we give a polynomial time matching algorithm for the membership problem for $\mathcal{L}_{\mathcal{TTSP}}$ by presenting the matching algorithm for the membership problem for the set of languages of term trees each of whose internal vertices has ordered or unordered children. Firstly, based on notions of an ordered term tree and an unordered term tree presented in [5] and [9], we formally define a term tree each of whose internal vertices has ordered or unordered children and call such a term tree a *partially-ordered term tree*. Moreover, we define a membership problem for the set of the languages of partially-ordered term trees. Next, we give a polynomial time matching algorithm for the membership problem for the set of the languages of partially-ordered term trees. Finally, we give a polynomial time algorithm for the membership problem for $\mathcal{L}_{\mathcal{TTSP}}$ by reducing this problem to the membership problem for the set of the languages of a special kind of partially-ordered term trees.

4.1 Partially-Ordered Term Trees and Its Languages

In this paper, unless otherwise indicated, we call a rooted tree each of whose internal vertices has ordered or unordered children a *tree* simply. We call an internal vertex having ordered children (resp., unordered children) an *o-vertex* (resp., a *u-vertex*) simply.

Definition 3: Let $T = (V_T, E_T)$ be a tree where V_T and E_T are sets of vertices and edges, respectively. Let E_t and H_t be a partition of E_T (i.e., $E_t \cup H_t = E_T$ and $E_t \cap H_t = \emptyset$). Let $V_t = V_T$. A triplet $t = (V_t, E_t, H_t)$ is called a *partially-ordered term tree*. A partially-ordered term tree $t = (V_t, E_t, H_t)$ is *linear* if all variables in H_t have mutually distinct variable labels in $\mathcal{X}$.

In this paper, we deal with only linear partially-ordered term trees. Hence, unless otherwise indicated, we call a linear partially-ordered term tree a *term tree* simply. We denote by $\mathcal{TT}$ the set of all term trees. A term tree with no variable is called a *ground term tree*, which is a tree. We denote by $\mathcal{T}$ the set of all ground term trees. In the same way as a TTSP term graph, we assume that every edge and every variable have elements in Λ and $\mathcal{X}$ as labels, respectively, but every vertex has no label. Hence, we use the same notations of an edge and a variable as those of a TTSP term graph. For a term tree t and two vertices u, v of t, u is an ancestor of v and v is a descendant of u if there exists a path from u to v. For a tree or a term tree T, we call the maximal length of paths from the root of T to leaves the *height* of T.

For a term tree t and every internal vertex u in t having ordered children, all children of u have a total ordering on all children of u. The ordering on the children of u is denoted by $<_u^t$. Let $s = (V_s, E_s, H_s)$ and $t = (V_t, E_t, H_t)$ be two term trees. We say that s and t are *isomorphic*, denoted by $s \equiv t$, if there is a bijection φ from V_s to V_t such that (i) the root of s is mapped to the root of t by φ, (ii) u is an o-vertex of s if and only if $\varphi(u)$ is an o-vertex in t, (iii) $(u, a, v) \in E_s$ if and only if $(\varphi(u), a, \varphi(v)) \in E_t$, (iv) $[u, x, v] \in H_s$ if and only if $[\varphi(u), y, \varphi(v)] \in H_t$, for some x and y in $\mathcal{X}$, (v) for any o-vertex u in s which has more than one child, and for any two children u' and u'' of u, $u' <_u^s u''$ if and only if $\varphi(u') <_{\varphi(u)}^t \varphi(u'')$.

Let t be a term tree with at least two vertices and x a variable label in $\mathcal{X}$. Let $\sigma = [u, u']$ be a list of two vertices in t, where u is the root of t and u' is a leaf of t. The form $x := [t, \sigma]$ is called a *binding* for x. A *substitution* is a finite collection of bindings $\{x_1 := [t_1, \sigma_1], \cdots, x_n := [t_n, \sigma_n]\}$, where x_i's are mutually distinct variable labels in $\mathcal{X}$, t_i's are ground term trees, and σ_i's are lists of two vertices in t_i.

In the same way as a TTSP term graph, for a term tree t and a substitution θ, we define an *instance* of t by θ, denoted by $t\theta$, as a term tree obtained from t by applying θ to t. We define the root of the instance $t\theta$ of t by θ as the root of t. Further we have to give a new total ordering $<_v^{f\theta}$ on every vertex v of $f\theta$. These orderings are defined in a natural way.

Let $f = (V_f, E_f, H_f)$ be a term tree and $\theta = \{x_1 := [g_1, \sigma_1], \cdots, x_n := [g_n, \sigma_n]\}$ a substitution. Suppose that v is an o-vertex in $f\theta$ which has more than one child and let v' and v'' be two children of v of $f\theta$. There are five cases in which the ordering between v' and v'' have to be newly defined. (1) $v \in V_{f\theta} - V_f$: In this case, there is a term tree $g \in \{g_1, \cdots, g_n\}$ such that all of v, v', v'' are in V_g. Then $v' <_v^{f\theta} v''$ is defined if and only if $v' <_v^g v''$. If $v \in V_f$, we have the following four subcases. (2) $v' \in V_f$ and $v'' \in V_f$: $v' <_v^{f\theta} v''$ is defined if and only if $v' <_v^f v''$. (3) $v' \in V_f$ and there is a term tree $g \in \{g_1, \cdots, g_n\}$ such that $v'' \in V_g$: Let w be the child port of the variable for which g is substituted. We note that v is the parent port of the variable. Then $v' <_v^{f\theta} v''$ is defined if and only if $v' <_v^f w$. (4) There is a term tree $g \in \{g_1, \cdots, g_n\}$ such that both of v' and v'' are in V_g: Since v is identified with the root of g (say u), $v' <_v^{f\theta} v''$ is defined

if and only if $v' <_u^g v''$. (5) There are two distinct term trees $g, g' \in \{g_1, \cdots, g_n\}$ such that $v' \in V_g$ and $v'' \in V_{g'}$: Let w (resp. w') be the child port of the variable for which g (resp. g') is substituted. Then $v' <_v^{f\theta} v''$ is defined if and only if $w <_v^f w''$.

For example, let t be a term tree in Fig. 2 and $\theta = \{x := [T_1, [w_1^1, w_2^1]], y := [T_2, [w_1^2, w_2^2]]\}$ a substitution, where T_1 and T_2 are trees in Fig. 2. Then the instance $t\theta$ of the term tree t by θ is the tree T in Fig. 2.

Definition 4: For a term tree $t \in \mathcal{TT}$, the *term tree language* of t, denoted by $L(t)$, is defined as $\{s \in \mathcal{T} \mid s \equiv t\theta \text{ for some substitution } \theta\}$.

4.2 A Polynomial Time Algorithm for Solving the Membership Problem for Term Trees

In this section, by extending the polynomial time matching algorithm in [5] and [9], we give a polynomial time matching algorithm for solving the following membership problem for the class $\mathcal{L}_{\mathcal{TT}} = \{L(t) \subseteq \mathcal{T} \mid t \in \mathcal{TT}\}$.

Membership Problem for $\mathcal{L}_{\mathcal{TT}}$.
Instance: A term tree $t \in \mathcal{TT}$ and a tree $T \in \mathcal{T}$.
Question: Does $L(t)$ contain T ?

For a tree or a term tree t and its vertex u, $t[u]$ denotes the subtree consisting of u and all descendants of u in t. We note that u is the root of $t[u]$. Let $t = (V_t, E_t, H_t)$ be a term tree and T a tree. We assume that all vertices of a term tree t are associated with mutually distinct numbers, called *vertex identifiers*. We denote by $I(u)$ the vertex identifier of $u \in V_t$.

Definition 5: A correspondence set of a vertex v of T, denoted by $CS(v)$, is a subset of $\{I(u) \mid u \in V_t\} \cup \{(I(u)) \mid u$ is the child port of a variable of $t\}$ satisfying the following two conditions. (1) $I(u) \in CS(v)$ if and only if $L(t[u])$ contains $T[v]$, and (2) $(I(u)) \in CS(v)$ if and only if there exists a proper descendant v' of v such that $L(t[u])$ contains $T[v']$.

Below we call a correspondence set a C-set shortly. Let u be an internal vertex of t and $c_1, \ldots, c_m$ $(m \geq 1)$ all ordered (or unordered) children of u. The *C-set-attaching rule* of u is of the form $I(u) \overset{u}{\leftarrow}$ (or $\overset{o}{\leftarrow}, \leftarrow$)$\xi(c_1), \ldots, \xi(c_m)$, where $\xi(c_i) = (I(c_i))$ if c_i is the child port of a variable, $\xi(c_i) = I(c_i)$ otherwise. The C-set-attaching rule of t, denoted by $Rule(t)$, is defined as follows.

$Rule(t) =$
$\bigcup_{u \in V_t}(\{I(u) \overset{u}{\leftarrow} \xi(c_1), \ldots, \xi(c_m) \mid u$ is a u-vertex of $t\}$
$\cup\{I(u) \overset{o}{\leftarrow} \xi(c_1), \ldots, \xi(c_m) \mid u$ is an o-vertex of $t\}$
$\cup\{(I(u)) \leftarrow (I(u)) \mid u$ is the child port of a variable$\})$.

For example, for a TTSP term tree t given in Fig. 4, we give the C-set-attaching rule $Rule(t)$ of t in Fig. 4.

In Fig. 5, we present an algorithm TT-Matching for solving the membership problem for $\mathcal{L}_{\mathcal{TT}}$. In TT-Matching, by using CS-Attaching given in Fig. 5, we repeatedly attach a C-set to each vertex of a given tree T in the bottom-up manner, that is, from the leaves to the root of T. If the C-set of the root of T has the vertex identifier of the root of t, then we conclude that $L(t)$ contains T. For example, given a term tree t and a tree T in Fig. 4, TT-Matching constructs the C-set-attaching rule $Rule(t)$ of t in Fig. 4 and attaches C-sets in Fig. 4 to all vertices of T. In this example, TT-Matching returns "*yes*", because the C-set of the root A of T includes the vertex identifier 1 of the root of t.

Theorem 2: The membership problem for $\mathcal{L}_{\mathcal{TT}}$ is solvable in polynomial time.

Proof. Let v be a vertex of T. If v is a leaf of T, the C-set of v is correctly calculated at the first **foreach** loop of Algorithm TT-Matching(T, t). For an internal vertex v of T, let $CS(v)$ be a set lastly attached to v by Procedure CS-Attaching$(v, Rule(t))$. We assume that the C-sets of all proper descendants of v are correctly calculated by Procedure CS-Attaching. Then we have the following two claims.

Claim 1. $I(u) \in CS(v)$ if and only if $L(t[u])$ contains $T[v]$.
Claim 2. $(I(u)) \in CS(v)$ if and only if there exists a proper descendant v' of v such that $L(t[u])$ contains $T[v']$.
Proof of Claim 1. We suppose that u is an o-vertex and $I(u) \in CS(v)$. From the condition of the **if** statement at the line 5 of CS-Attaching, there is a rule $I(u) \overset{o}{\leftarrow} \xi(c_1), \ldots, \xi(c_m)$ $(m \geq 1)$ and there are exactly m ordered children of v which satisfy the condition of the **if** statement. Here we denote the m ordered children by $d_1, \ldots, d_m$. Since $CS(d_i)$ contains $I(c_i)$ or $(I(c_i))$, from the definition of the C-set, we can construct a substitution θ such that $t[u]\theta \equiv T[v]$. Then $L(t[u])$ contains $T[v]$. Conversely, if $L(t[u])$ contains $T[v]$, there is

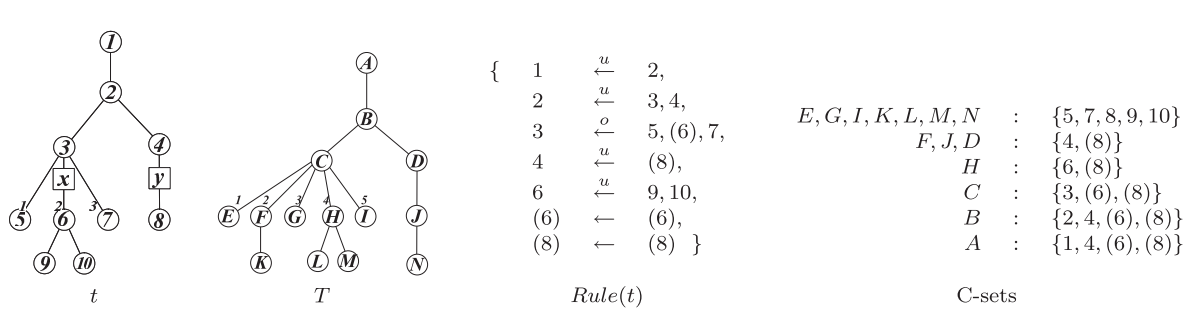

Fig. 4 A term tree t, the C-set-attaching rule $Rule(t)$ of t, a tree T and C-sets which are attached for vertices in T.

Algorithm TT-MATCHING(T, t);
input T: a tree, t: a term tree;
/* Let r_T and r_t be the roots of T and t, respectively; */
output "*yes*" or "*no*";
begin
Construct the C-set-attaching rule $Rule(t)$ of t;
foreach leaf v of T **do** $CS(v) := \{I(\ell) \mid \ell$ is a leaf of $t\}$;
while there exists a vertex v of T which is not attached any C-set but all of whose children are attached C-sets, respectively
do CS-ATTACHING$(v, Rule(t))$;
if $I(r_t) \in CS(r_T)$ **then output** "*yes*" **else output** "*no*"
end.

Procedure CS-ATTACHING$(v, Rule(t))$;
begin
1. $CS := \emptyset$;
2. Let $CS(c'_1), \cdots, CS(c'_{m'})$ be the C-Sets of all children $c'_1, \ldots, c'_{m'}$ $(m' \geq 1)$ of v in T, respectively;
3. **if** v is an o-vertex **then**
4. **foreach** $I(u) \overset{o}{\leftarrow} \xi(c_1), \cdots, \xi(c_m)$ in $Rule(t)$ **do begin**
5. **if** for the interval $[1, m']$, there are m intervals $[\ell_1, \ell_2 - 1], \ldots, [\ell_m, \ell_{m+1} - 1]$ satisfying the following condition, where $1 = \ell_1 < \ell_2 < \cdots < \ell_m < \ell_{m+1} - 1 = m'$:
For each i $(1 \leq i \leq m)$,
1. if $\xi(c_i) = I(c_i)$ then $\ell_i = \ell_{i+1} - 1$ and $I(c_i) \in CS(c'_{\ell_i})$,
2. if $\xi(c_i) = (I(c_i))$ then $CS(c'_{k_i})$ has $I(c_i)$ or $(I(c_i))$ for some k_i $(\ell_i \leq k_i \leq \ell_{i+1} - 1)$.
6. **then** $CS := CS \cup \{I(u)\}$
7. **end;**
8. **if** v is a u-vertex **then**
9. **foreach** $I(u) \overset{u}{\leftarrow} \xi(c_1), \cdots, \xi(c_m)$ in $Rule(t)$ **do begin**
10. Construct a bipartite graph $G = (\{I(c_1), \ldots, I(c_m)\}, \{CS(c'_1), \ldots, CS(c'_{m'})\}, E)$ where E is the set $\{\{I(c_i), CS(c'_j)\} \mid 1 \leq i \leq m, 1 \leq j \leq m', I(c_i) \in CS(c'_j)\} \cup \{\{I(c_i), CS(c'_j)\} \mid 1 \leq i \leq m, 1 \leq j \leq m', (I(c_i)) \in CS(c'_j)$ and $\xi(c_i) = (I(c_i))\}$;
11. **if** for any i $(1 \leq i \leq m)$, $\xi(c_i) = I(c_i)$ and there exists a perfect matching on G **then** $CS := CS \cup \{I(u)\}$;
12. **else if** there exists a matching of size m **then** $CS := CS \cup \{I(u)\}$
13. **end**;
14. **foreach** $(I(u)) \leftarrow (I(u))$ in $Rule(t)$ **do**
15. **if** there is a set in $CS(c'_1), \cdots, CS(c'_{m'})$ which has $I(u)$ or $(I(u))$ **then**
16. $CS := CS \cup \{(I(u))\}$;
17. Attach CS to v
end;

Fig. 5 Algorithm TT-MATCHING and Procedure CS-ATTACHING.

a substitution θ such that $t[u]\theta \equiv T[v]$. Let ψ be an isomorphism from $t[u]\theta$ to $T[v]$. Let $I(u) \overset{o}{\leftarrow} \xi(c_1), \ldots, \xi(c_m)$ be the C-set-attaching rule of u. There are exactly m ordered children of v, denoted by $d_1, \ldots, d_m$, such that for $i = 1, \ldots, m$, if $\xi(c_i) = I(c_i)$ then $\psi(c_i) = d_i$, and if $\xi(c_i) = (I(c_i))$ then $\psi(c_i)$ is a descendant of d_i. From the definition of the C-set, $CS(d_i)$ contains $I(c_i)$ or $(I(c_i))$ for each $i = 1, \ldots, m$. Therefore, $d_1, \ldots, d_m$ are ordered children which satisfy the condition of the line 5 of CS-ATTACHING. Then we have $I(u) \in CS(v)$. In the case that u is a u-vertex, we can prove this claim similarly. (*End of Proof of Claim 1*)

Proof of Claim 2. From the lines 15-17 of CS-ATTACHING, if

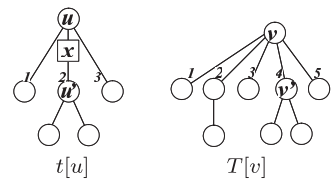

Fig. 6 A term tree $t[u]$ and a tree $T[v]$ is a subtree of t and a subtree of T in Fig. 4, respectively.

$(I(u)) \in CS(v)$ then u is the child port of some variable and there is a child of v whose C-set contains $I(u)$ or $(I(u))$. From the definition of the C-set, there exists a proper descendant v' of v such that $L(t[u])$ contains $T[v']$. Conversely, we suppose that there is a proper descendant v' of v such that $L(t[u])$ contains $T[v']$. If v' is a child of v, the $CS(v')$ contains $I(u)$. Then $(I(u)) \in CS(v)$. Otherwise, there is a child w of v such that v' is a proper descendant of w. From the definition of the C-set, $CS(w)$ has $(I(u))$. Then $(I(u)) \in CS(v)$. (*End of Proof of Claim 2*)

From these claims, Algorithm TT-MATCHING(T, t) correctly solves the membership problem for $\mathcal{L}_{\mathcal{TT}}$.

Finally, we show that TT-MATCHING can solve the membership problem for $\mathcal{L}_{\mathcal{TT}}$ in polynomial time as follows. Let N and n be the numbers of vertices of T and t, respectively. We can show that the time complexity of TT-MATCHING is $O(\sum_{v \in Q_T} \Phi(v, Rule(t)))$, where Q_T is the set of internal vertices in T and $\Phi(v, Rule(t))$ is the time complexity of the procedure CS-ATTACHING for v and $Rule(t)$.

In [3], Hopcroft and Karp presented an $O(\sqrt{|V|}|E|)$ time algorithm for finding a maximum cardinality matching for a given bipartite graph $G = (V, E)$. By using the algorithm RULE_MATCHING given in [9] and Hopcroft and Karp's algorithm for the line 5 and lines 11-12 of CS-ATTACHING, respectively, the time complexity $\Phi(v, Rule(t))$ of CS-ATTACHING is $\sum_{r \in Rule(t)}(\sqrt{D_v + d_r} \times D_v \times d_r)$, where D_v is the number of children of v and d_r is the number of elements in the righthand side of the rule r. Since $d_r \leq D_v$, $\sum_{r \in Rule(t)} d_r \leq 2n - 2$, $\sum_{v \in Q_T} D_v = N - 1$ and for each $v \in Q_T$, $D_v \leq D_{max}$ hold, we have $O(\sum_{v \in Q_T} \sum_{r \in Rule(t)} (\sqrt{D_v + d_r} \times D_v \times d_r) = O(\sqrt{D_{max}} \times N \times n)$, where $D_{max} = \max_{v \in Q_T} D_v$. Hence, this theorem holds. □

For example, let u and u' be the vertices 3 and 6 of t in Fig. 4, respectively. And let v and v' be the vertices C and H of T in Fig. 4, respectively. The term tree $t[u]$ and the tree $T[v]$ are described in Fig. 6. Then, we see that $L(t[u])$ contains $T[v]$ and $I(u) \in CS(v)$. Furthermore, $(I(u')) \in CS(v)$ and there exists a proper descendant v' of v such that $L(t[u'])$ contains $T[v']$.

4.3 A Polynomial Time Algorithm for Solving the Membership Problem for $\mathcal{L}_{\mathcal{TTSP}}$

In this section, we present a polynomial time algorithm for solving the membership problem for $\mathcal{L}_{\mathcal{TTSP}}$ by reducing this problem to the membership problem for $\mathcal{L}_{\mathcal{TT}}$. A tree

whose vertices have labels is called a *colored-tree*.

Definition 6: A *decomposition tree* of a TTSP term graph is recursively defined as follows.

(1) A colored-tree consisting of only one vertex having a label a in Λ is a decomposition tree of a TTSP term graph consisting of two vertices u, v and an edge (u, a, v).
(2) A colored-tree consisting of only one vertex having a label x in $\mathcal{X}$ is a decomposition tree of a TTSP term graph consisting of two vertices u, v and a variable $[u, x, v]$.
(3) Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be decomposition trees of TTSP term graphs g_1 and g_2, respectively, and r_1 and r_2 roots of T_1 and T_2, respectively. Let a be a label in Λ. Then, the following two colored-trees are decomposition trees.
   (a) A colored-tree $T = (V_T, E_T)$ having a u-vertex r_u as the root and having T_1 and T_2 as children of r_u is a decomposition tree of the TTSP term graph $g_1//g_2$. Namely, $V_T = V_1 \cup V_2 \cup \{r_u\}$, and $E_T = E_1 \cup E_2 \cup \{(r_u, a, r_1), (r_u, a, r_2)\}$.
   (b) A colored-tree $T = (V_T, E_T)$ having an o-vertex r_o as the root and having T_1 and T_2 as children of r_o with $r_1 <^T_{r_o} r_2$ is a decomposition tree of the TTSP term graph $g_1 * g_2$. Namely, $V_T = V_1 \cup V_2 \cup \{r_o\}$ and $E_T = E_1 \cup E_2 \cup \{(r_o, a, r_1), (r_o, a, r_2)\}$.

We remark that a decomposition tree is a tree whose internal vertices are o-vertices or u-vertices, all of whose leaves have labels in $\Lambda \cup \mathcal{X}$ and all of whose edges have the label a.

Let T be a decomposition tree having at least two vertices. We call an edge whose both endpoints are o-vertices (resp., u-vertices) an *o-edge* (resp., a *u-edge*). Let $e = (u, a, v)$ be an o-edge or a u-edge of T. A *contraction* of e is an operation of removing e from T, identifying u with v and, if u and v are o-vertices, updating a total ordering $<^{T'}_u$ on the o-vertex $u(= v)$ of the result tree T' as follows.

(1) For w and w' are children of u if $w <^T_u w'$ then $w <^{T'}_u w'$.
(2) For w and w' are children of v if $w <^T_v w'$ then $w <^{T'}_u w'$.
(3) For w and w' are children of u and v, respectively, if $w <^T_u v$ then $w <^{T'}_u w'$.
(4) For w and w' are children of u and v, respectively, if $v <^T_u w$ then $w' <^{T'}_u w$.

Let g be a TTSP term graph and T_g a decomposition tree of g. A *contraction tree* of T_g is the tree obtained from T_g by recursively applying contractions of o-edges and u-edges until there exists neither o-edges nor u-edges. For the TTSP term graph g in Fig. 1, for example, T_1 shown in Fig. 7 is a decomposition tree of g and T_2 shown in Fig. 7 is the contraction tree of T_1.

Let g be a TTSP term graph and $T = (V_T, E_T)$ the contraction tree of a decomposition tree of g and r the root of T. A *parse tree* of g is the term tree $t_g = (V_t, E_t, H_t)$ such that $V_t = V_T \cup \{v_0\}$, $E_t = \{(v_0, a, r)\} \cup \{(u, a, v) \mid (u, a, v) \in E_T, v$ is an internal vertex of $T\} \cup \{(u, b, v) \mid (u, a, v) \in E_T, v$ is a leaf labeled with $b \in \Lambda\}$ and $H_t = \{[u, x, v] \mid (u, a, v) \in E_T, v$ is a leaf labeled with $x \in \mathcal{X}\}$. For example, for a TTSP term graph g in Fig. 1, T_3 presented in Fig. 7 is the parse tree of g.

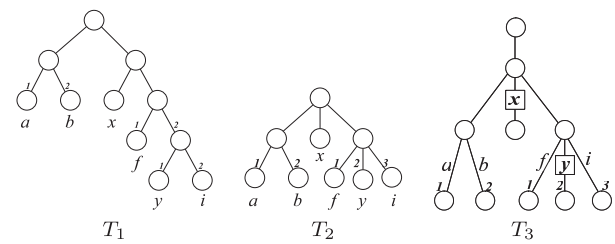

Fig. 7 A decomposition tree T_1 of the TTSP term graph g given in Fig. 1, the contraction tree T_2 of T_1 and the parse tree T_3 of g in Fig. 1.

Lemma 2: Let g_1 and g_2 be TTSP term graphs. Let t_1 and t_2 be the parse trees of g_1 and g_2, respectively. Then, $g_1 \equiv g_2$ if and only if $t_1 \equiv t_2$.

Proof. Let T_1 and T_2 be the contraction trees of g_1 and g_2, respectively. It is sufficient to show that $g_1 \equiv g_2$ if and only if $T_1 \equiv T_2$. Let $g_1 = (V_1, E_1, H_1)$. We show both the "if" part and "only if" part by inductions on $|E_1 \cup H_1|$. If $|E_1 \cup H_1| = 1$, it is obvious that $g_1 \equiv g_2$ if and only if $T_1 \equiv T_2$. Then we assume that $|E_1 \cup H_1| \geq 2$.
Only if part. We have in the two cases. (1) There exist k ($k \geq 2$) TTSP term graphs $g^1_1, \ldots, g^k_1$ such that $g_1 \equiv g^1_1// \cdots //g^k_1$ and none of g^i_1 ($1 \leq i \leq k$) can be obtained from two TTSP term graphs by a parallel operation. Since $g_1 \equiv g_2$, there exist k TTSP term graphs $g^1_2, \ldots, g^k_2$ such that $g^i_1 \equiv g^i_2$ for any i ($1 \leq i \leq k$), $g_2 \equiv g^1_2// \cdots //g^k_2$, and none of g^i_2 ($1 \leq i \leq k$) can be obtained from two TTSP term graphs by a parallel operation. For any i ($1 \leq i \leq k$), let T^i_1 and T^i_2 be the contraction trees of g^i_1 and g^i_2, respectively. T_1 is obtained from $T^1_1, \ldots, T^k_1$ and a new u-vertex by connecting the roots of $T^1_1, \ldots, T^k_1$ to the u-vertex. T_2 is also obtained from $T^1_2, \ldots, T^k_2$ and a new u-vertex by the same way. Therefore we have $T_1 \equiv T_2$. (2) There exist k ($k \geq 2$) TTSP term graphs $g^1_1, \ldots, g^k_1$ such that $g_1 \equiv g^1_1 * \cdots * g^k_1$ and none of g^i_1 ($1 \leq i \leq k$) can be obtained from two TTSP term graphs by a series operation. This case is shown in a similar way to the case (1).
If part. Let r_1 and r_2 be the roots of T_1 and T_2, respectively. Let $T^1_1, \ldots, T^k_1$ be the contraction trees whose roots are the children of r_1. Since $T_1 \equiv T_2$, r_2 has exactly k children, there are k contraction trees $T^1_2, \ldots, T^k_2$ whose roots are the children of r_2, and $T^i_1 \equiv T^i_2$ ($1 \leq i \leq k$). We have the following two cases. (a) r_1 is a u-vertex and (b) r_1 is an o-vertex. Here we show only the case (a). The case (b) can be shown in a similar way. Let $g^1_1, \ldots, g^k_1$ be the TTSP term graphs whose contraction trees are $T^1_1, \ldots, T^k_1$, respectively. Let $g^1_2, \ldots, g^k_2$ be the TTSP term graphs whose contraction trees are $T^1_2, \ldots, T^k_2$, respectively. By the induction hypothesis, $g^i_1 \equiv g^i_2$ ($1 \leq i \leq k$). Since both r_1 and r_2 are u-vertices, $g_1 \equiv g^1_1// \cdots //g^k_1$ and $g_2 \equiv g^1_2// \cdots //g^k_2$. Therefore we have $g_1 \equiv g_2$. □

Lemma 3: Let g be a TTSP term graph and G a TTSP graph. Let t_g and T_G be the parse trees of g and G, respectively. Then, $G \in L(g)$ if and only if $T_G \in L(t_g)$.

Proof. From the definition of parse trees, we have the following two claims.
Claim 1. Let $x := [p, [u_p, v_p]]$ be a binding for the variable label x of g and t_p the parse tree of p, where p is a TTSP term graph, and u_p and v_p are the source and the sink of p, respectively. Let t be the parse tree of $g\{x := [p, [u_p, v_p]]\}$. Then we have $t \equiv t_g\{x := [t_p, [r_p, \ell_p]\}$, where r_p and ℓ_p are the root and a leaf of t_p, respectively.
Claim 2. Let $x := [t', [r', \ell']]$ be a binding for the variable label x of t_g, where t' is a term tree, and r' and ℓ' are the root and a leaf of t', respectively. If $t_g\{x := [t', [r', \ell']]\}$ is the parse tree of a TTSP term graph, t' is also the parse tree of a TTSP term graph.

We assume $G \in L(g)$. Then there is a substitution $\theta = \{x_1 := [p_1, [u_{p_1}, v_{p_1}]], \cdots, x_n := [p_n, [u_{p_n}, v_{p_n}]]\}$ such that $G \equiv g\theta$, where $x_1, \ldots, x_n$ are mutually distinct variable labels, and for any i ($1 \le i \le n$), p_i is a TTSP term graph, and u_{p_i} and v_{p_i} are the source and the sink of p_i, respectively. For any i ($1 \le i \le n$), let t_{p_i} be the parse tree of p_i, and r_{p_i} and ℓ_{p_i} the root and a leaf of t_{p_i}, respectively. Let $\tau = \{x_1 := [t_{p_1}, [r_{p_1}, \ell_{p_1}]], \cdots, x_n := [t_{p_n}, [r_{p_n}, \ell_{p_n}]]\}$. From Claim 1, the parse tree of $g\theta$ is isomorphic to $t_g\tau$. Then, from Lemma 2, $T_G \equiv t_g\tau$. Therefore $T_G \in L(t_g)$ holds.

Conversely, we assume $T_G \in L(t_g)$. Then there is a substitution $\tau = \{x_1 := [t_1, [r_{t_1}, \ell_{t_1}]], \cdots, x_n := [t_n, [r_{t_n}, \ell_{t_n}]]\}$ such that $T_G \equiv t_g\tau$, where $x_1, \ldots, x_n$ are mutually distinct variable labels, and for any i ($1 \le i \le n$), t_i is a term tree, and r_{t_i} and ℓ_{t_i} are the root and a leaf of t_i, respectively. From Claim 2, there are TTSP graphs $p_1, \ldots, p_n$ such that each t_i is the parse tree of p_i ($1 \le i \le n$). Let $\theta = \{x_1 := [p_1, [u_{p_1}, v_{p_1}]], \cdots, x_n := [p_n, [u_{p_n}, v_{p_n}]]\}$, where for any i ($1 \le i \le n$), u_{p_i} and v_{p_i} are the source and the sink of p_i, respectively. From Claim 1, $t_g\tau$ is isomorphic to the parse tree of $g\theta$. Then, from Lemma 2, $G \equiv g\theta$. Therefore $G \in L(g)$ holds. □

In Fig. 8, we give a polynomial time algorithm TTSPTG-Matching which solves the membership problem for $\mathcal{L}_{\mathcal{TTSP}}$.

Theorem 3: The membership problem for $\mathcal{L}_{\mathcal{TTSP}}$ is solvable in polynomial time.

Proof. It is easy to see that TTSPTG-Matching certainly terminates. From Lemma 3, given a TTSP graph $G = (V_G, E_G)$ and a TTSP term graph $g = (V_g, E_g, H_g)$, TTSPTG-Matching correctly decides whether or not $L(g)$ contains G. By using a linear time algorithm presented by Valdes et al. [12], we can construct the parse trees of G and g in time proportional to $|V_G| + |E_G|$. Moreover, by using TT-Matching given in Fig. 5, Line 3 can be executed in $O(|E_G|^{1.5} \times |E_g \cup H_g|)$. Hence, for a given TTSP graph G and a given TTSP term graph g, the algorithm TTSPTG-Matching decides whether or not $L(g)$ contains G in $O(|E_G|^{1.5} \times |E_g \cup H_g|)$ time. □

Algorithm TTSPTG-Matching(G, g);
input G: a TTSP graph, g: a TTSP term graph;
output "*yes*" or "*no*";
begin
1. T:= Parse(G); // Construct the parse tree T of G
2. t:= Parse(g); // Construct the parse tree t of g
3. **output** TT-Matching(T, t)

end.

Procedure Parse(G);
input G: a TTSP term graph;
output T_G: the parse tree of G;
begin
  Construct a decomposition tree T of G;
  Construct the contraction tree T' of T;
  Construct the parse tree T_G of G from T';
  return T_G

end;

Fig. 8 Algorithm TTSPTG-Matching.

5. An Algorithm for Finding a Minimally Generalized TTSP Term Graph

Let Λ be a set of edge labels. In this section, we assume that $|\Lambda| = \infty$. Let g and f be TTSP term graphs. We denote $g \preceq f$ if there exists a substitution θ such that $g \equiv f\theta$. For any TTSP term graph g, we denote by $s(g)$ the TTSP term graph obtained from g by replacing each of all edges of g with a variable, i.e., for $g = (V_g, E_g, H_g)$, $s(g) = (V_g, \emptyset, H'_g)$, where $H'_g = H_g \cup \{[u, x_e, v] \mid e = (u, a, v) \in E_g$ and x_e is a new variable label only for $e\}$. For any two TTSP term graphs g and f, we write $g \approx f$ if $s(g) \equiv s(f)$. It is easy to see the following lemma since $|\Lambda| = \infty$.

Lemma 4: Let g and f be two TTSP term graphs in $\mathcal{TG}_{\mathcal{TTSP}}$. If $g \approx f$ and $L(g) \subseteq L(f)$ then $g \preceq f$.

The algorithm MINL-$\mathcal{TTSP}$ (Fig. 9) solves the MINL problem for $\mathcal{L}_{\mathcal{TTSP}}$. The procedure Variable-Extension (Fig. 9) extends a TTSP term graph g by adding variables as much as possible while $S \subseteq L(g)$ holds. Edge-Replacing (Fig. 9) tries to replace each variable in g with a labeled edge if possible. We use the following three substitutions in the algorithm. These substitutions are called *refinement operators*.

Par(h) : Replace $h = [u, x, v] \in H_g$ with $h' = [u, x', v]$ and $h'' = [u, x'', v]$, where x' and x'' are new variable labels in $\mathcal{X}$.
Ser(h) : Replace $h = [u, x, v] \in H_g$ with $h' = [u, x', w]$ and $h'' = [w, x'', v]$, where w is a new vertex and x' and x'' are new variable labels in $\mathcal{X}$.
Lab$(h)_\lambda$: Replace $h = [u, x, v] \in H_g$ with (u, λ, v), where $\lambda \in \Lambda$.

Lemma 5: Let $g \in \mathcal{TG}_{\mathcal{TTSP}}$ be the TTSP term graph just after the procedure Variable-Extension for an input S finishes. Let g' be another TTSP term graph. If $S \subseteq L(g') \subseteq L(g)$ then $g' \approx g$.

Algorithm MINL-$\mathcal{TTSP}(S)$;
input S: a set of TTSP graphs;
output a minimally generalized TTSP term graph g in $\mathcal{TG}_{\mathcal{TTSP}}$ for S;
begin
  $g := (\{u, v\}, \emptyset, \{[u, x, v]\})$,
  //u and v are new vertices and x is in $\mathcal{X}$;
  VARIABLE-EXTENSION(g, S); EDGE-REPLACING(g, S);
  output g
end.

Procedure VARIABLE-EXTENSION(g, S);
begin
  while there exist a variable h in g and an operator $\mathcal{R}$ ($\mathcal{R} \in \{Par, Ser\}$) s.t. $S \subseteq L(g\,\mathcal{R}(h))$ **do** $g := g\,\mathcal{R}(h)$
end;

Procedure EDGE-REPLACING(g, S);
begin
  Let Λ_S be the set of edge labels which appear in S;
  foreach variable h in g **do**
    foreach edge label $\lambda \in \Lambda_S$ **do begin**
      $g' := g\,Lab(h)_\lambda$;
      if $S \subseteq L(g')$ **then begin** $g := g'$; **break end**
    end
end;

Fig. 9 Algorithm MINL-$\mathcal{TTSP}$.

Proof. Let $T(g')$ and $T(g)$ be contraction trees of decomposition trees of g' and g, respectively. In the procedure VARIABLE-EXTENSION (Fig. 9), for each variable of g, refinement operators Par and Ser are executed as much as possible. From this algorithm, we can show that $T(g)$ must be equivalent to $T(g')$ if the labels of leaves are not considered. The statement follows from this fact. □

Lemma 6: Let $g \in \mathcal{TG}_{\mathcal{TTSP}}$ be the output of the algorithm MINL-$\mathcal{TTSP}$ for an input S. Let g' be a TTSP term graph satisfying that $S \subseteq L(g') \subseteq L(g)$. Then $g' \equiv g$.

Proof. From Lemmas 4 and 5, we have $g' \preceq g$. The procedure EDGE-REPLACING (Fig. 9) replaces all possible variables with labeled edges. Therefore from $S \subseteq L(g')$, $g' \equiv g$ holds. □

Theorem 4: The algorithm MINL-$\mathcal{TTSP}$ finds a minimally generalized TTSP term graph in $\mathcal{TG}_{\mathcal{TTSP}}$ for a given set of TTSP graphs in $\mathcal{TTSP}$ in polynomial time.

Proof. The correctness follows from Lemma 6. Let $S = \{G_1, \ldots, G_m\}$ be an input set of TTSP graphs, where $G_i = (V_i, E_i)$ $(1 \le i \le m)$. Let $N_{\min} = \min_{1 \le i \le m} |E_i|$ and $N_{\max} = \max_{1 \le i \le m} |E_i|$. Let $g = (V_g, E_g, H_g)$ be the TTSP term graph generated by the algorithm MINL-$\mathcal{TTSP}$ for S. It is easy to see that $|E_g \cup H_g| \le N_{\min}$. Therefore $O(N_{\min})$ refinement operators are totally executed in VARIABLE-EXTENSION. From Theorem 3, one inclusion test needs $\sum_{1 \le i \le m} O(|E_i|^{1.5} \times |E_g \cup H_g|) = O(mN_{\max}^{1.5} N_{\min})$ time. Since one inclusion test is executed every refinement operation, the procedure VARIABLE-EXTENSION needs $O(mN_{\max}^{1.5} N_{\min}^2)$ time. Let Λ_S be the set of edge labels which appear in S. Since EDGE-REPLACING tries to replace variables with labeled edges at most $|\Lambda_S| N_{\min}$ times, the procedure needs totally $|\Lambda_S| N_{\min} \times O(mN_{\max}^{1.5} N_{\min})$ time. Hence the total time for all executions in the algorithm MINL-$\mathcal{TTSP}$ is $O(|\Lambda_S| mN_{\max}^{1.5} N_{\min}^2)$, which is polynomial w.r.t. S. □

6. Conclusion

We have shown the polynomial time learnabilities of TTSP graph languages from positive data by giving a reduction to that of a special kind of term tree languages. Firstly, we have introduced a TTSP term graph as a graph pattern consisting of a TTSP graph structure and structured variables. Moreover, for a TTSP term graph g, we have defined a TTSP graph language $L(g)$ as the set of all TTSP term graphs obtained from g by substituting arbitrary TTSP graphs for all variables in g. Secondly, we have given a set $\mathcal{TT}$ of term trees such that there exists a bijection from the set $\mathcal{TG}_{\mathcal{TTSP}}$ of all TTSP term graphs to $\mathcal{TT}$, and have presented a polynomial time matching algorithm for solving the membership problem for $\mathcal{L}_{\mathcal{TTSP}} = \{L(g) \mid g \in \mathcal{TG}_{\mathcal{TTSP}}\}$ by giving a polynomial time matching algorithm for solving the membership problem for $\mathcal{L}_{\mathcal{TT}} = \{L(t) \mid t \in \mathcal{TT}\}$. Finally, we have presented a polynomial time algorithm for solving the minimal language problem for $\mathcal{L}_{\mathcal{TTSP}}$. By using the above polynomial time algorithms for $\mathcal{L}_{\mathcal{TTSP}}$, we have shown the polynomial time learnability of $\mathcal{L}_{\mathcal{TTSP}}$ from positive data.

Our results given in this paper lead us to study the learnability of languages over other classes of graphs such as series parallel graphs, outerplanar graphs, graphs of bounded treewidth (see [13]). As future works, we consider the learnability of languages on other classes of graph patterns. We also consider the learnability of the class of finite unions of TTSP graph languages from positive data. Furthermore, we consider the problem of deciding whether or not there is a minimally generalized TTSP term graph such that the number of variables is at most K for some integer K. Since the problem of finding a minimally generalized term tree such that the number of variables is at most K is NP-complete [8], we conjecture that the problem is NP-complete. Moreover, we consider applying our results in this paper to other fields such as data mining from graph structured data.

References

[1] D. Angluin, "Finding patterns common to a set of strings," J. Comput. Syst. Sci., vol.21, pp.46–62, 1980.
[2] D. Angluin, "Inductive inference of formal languages from positive data," Information and Control, vol.45, pp.117–135, 1980.
[3] J. Hopcroft and R. Karp, "An $n^{5/2}$ algorithm for maximum matching in bipartite graphs," SIAM J. Comput., vol.2, pp.225–231, 1973.
[4] S. Matsumoto, T. Shoudai, T. Uchida, T. Miyahara, and Y. Suzuki, "Learning of finite unions of tree patterns with internal structured variables from queries," IEICE Trans. Inf. & Syst., vol.E91-D, no.2, pp.222–230, Feb. 2008.
[5] T. Miyahara, T. Shoudai, T. Uchida, T. Kuboyama, K. Takahashi, and H. Ueda, "Discovering new knowledge from graph data using inductive logic programming," Proc. ILP-99, Springer-Verlag, LNAI 1634, pp.222–233, 1999.

[6] R. Okada, S. Matsumoto, T. Uchida, Y. Suzuki, and T. Shoudai, "Exact learning of finite unions of graph patterns from queries," Proc. ALT-2007, Springer-Verlag, LNAI 4754, pp.298–312, 2007.
[7] T. Shinohara, "Polynomial time inference of extended regular pattern languages," Springer-Verlag, LNCS 147, pp.115–127, 1982.
[8] T. Shoudai, T. Uchida, and T. Miyahara, "Polynomial time algorithms for finding unordered tree patterns with internal variables," Proc. FCT-2001, Springer-Verlag, LNCS 2138, pp.335–346, 2001.
[9] Y. Suzuki, R. Akanuma, T. Shoudai, T. Miyahara, and T. Uchida, "Polynomial time inductive inference of ordered tree patterns with internal structured variables from positive data," Proc. COLT-2002, Springer-Verlag, LNAI 2375, pp.169–184, 2002.
[10] K. Takamizawa, T. Nishizeki, and N. Saito, "Linear-time computability of combinatorial problems on series-parallel graphs," Journal of the Association for Computing Machinery, vol.29, no.3, pp.623–641, 1982.
[11] T. Uchida, T. Shoudai, and S. Miyano, "Parallel algorithms for refutation tree problem on formal graph systems," IEICE Trans. Inf. & Syst., vol.E78-D, no.2, pp.99–112, Feb. 1995.
[12] J. Valdes, R.E. Tarjan, and E.L. Lawler, "The recognition of series parallel digraphs," SIAM J. Comput., vol.11, pp.298–313, 1982.
[13] Jan van Leeuwen, ed., Handbook of theoretical computer science (vol.A): Algorithms and complexity, Elsevier and MIT Press, 1990.

Ryoji Takami received the B.S. degree in 2003, the M.S. degree in 2005 in Computer and Media Technologies from Hiroshima City University. Currentlly, he works at CSI Co. Ltd. His research interests include algorithmic learning theory and data mining.

Yusuke Suzuki received the B.S. degree in Physics, the M.S. and Dr. Sci. degrees in Informatics all from Kyushu University, in 2000, 2002 and 2007, respectively. He is currently a research associate of Graduate School of Information Sciences, Hiroshima City University, Hiroshima, Japan. His research interests include machine learning and data mining.

Tomoyuki Uchida received the B.S. degree in Mathematics, the M.S. and Dr. Sci. degrees in Information Systems all from Kyushu University, in 1989, 1991 and 1994, respectively. Currently, he is an associate professor of Graduate School of Information Sciences, Hiroshima City University. His research interests include data mining from semistructured data, algorithmic graph theory and algorithmic learning theory. He is a member of ACM.

Takayoshi Shoudai received the B.S. in 1986, the M.S. degrees in 1988 in Mathematics and the Dr. Sci. in 1993 in Information Science all from Kyushu University. Currently, he is an associate professor of Department of Informatics, Kyushu University. His research interests include algorithmic graph theory, algorithmic learning theory, and data mining from graph-structured data. He is a member of IPSJ and ACM.