数理解析研究所講究録
第1148巻 2000年 164-169

# Polynomial Time Matching Algorithms for Tree Structured Patterns

正代 隆義 (Takayoshi Shoudai)[1]　　宮原 哲浩 (Tetsuhiro Miyahara)[2]
内田 智之 (Tomoyuki Uchida)[2]

[1] Department of Informatics, Kyushu University 39, Kasuga 816-8580, Japan
shoudai@i.kyushu-u.ac.jp
[2] Faculty of Information Sciences, Hiroshima City University, Hiroshima 731-3194, Japan
{miyahara@its,uchida@cs}.hiroshima-cu.ac.jp

## 1 Introduction

Graph-structured data occurs in many domains, such as biomolecular database, chemical database, the World Wide Web, or semistructured data. Many researchers try to find hidden knowledge from structures of such data by using data mining techniques. The formalization of expressing graph-structured data is quite important for finding useful knowledge [10].

A term graph, which is one of expressions of graph-structured data, is a hypergraph whose hyperedges are regarded as variables. By expressing structures of data in database with term graphs, we can design tools for discovering hidden knowledge or background knowledge from graph-structured data. In Fig. 1, for example, we can obtain each tree $T_1$, $T_2$ and $T_3$ from the term tree $t$ by replacing hyperedges in $t$ with arbitrary trees. That is, the term tree $t$ shows common structures between them. The first-order language is much better suited for expressing background knowledge and graph structures [3]. Then, inductive logic programming (ILP) systems in knowledge discovery have been proposed [1, 2, 4]. In [8], we designed and implemented the knowledge discovery system KD-FGS for graph-structured data, which employs Formal Graph System (FGS,[11]) as a knowledge representation language and a refutably inductive inference as an ILP mechanism [9]. FGS is a kind of logic programming system which uses term graphs instead of terms in first-order logic. Therefore FGS can directly deal with graphs and is suited for expressing background knowledge obtained from graph-structured data. By using a term graph, we can design tools based on a graph pattern matching method for finding new knowledge represented by term graphs obtained from graph-structured data. Such tools are useful for finding association rules over term graphs, producing decision trees having term graphs as vertex labels, and finding the minimum term graph by using the minimum description length principle.

In this paper, we consider a matching problem for a term graph and a standard graph. Informally, the matching problem for a term graph $g$ and a graph $G$ is to decide whether or not there exists a graph $G'$ such that $G'$ is isomorphic to $G$ and $G'$ is obtained by replacing each variable in $g$ with an arbitrary graph. This problem is important for many knowledge discovery systems over term graphs for graph-structured data. Graphs have enough richness and flexibility to express unknown structures, but many elementary graph problems, e.g., subgraph isomorphism and largest common subgraph, are known to be NP-complete [5]. Due to this fact, it is difficult to solve the matching problem for a term graph in polynomial time. Then it is hard to design and implement a discovery system finding efficiently new knowledge from graph-structured data in practice. We consider interesting subclasses of term graphs, called regular term trees, such that their matching problems are solvable efficiently.

Let $\Sigma$ and $\Lambda$ be finite alphabets, and let $X$ be an alphabet. An element in $\Sigma$, $\Lambda$ and $X$ is called a *vertex label*, *edge label* and *variable label*, respectively. Assume that $(\Sigma\cup\Lambda)\cap X=\emptyset$. A *term graph* $g=(V,E,H)$ consists of a vertex set $V$, an edge set $E$ and a multi-set $H$. Each element in $H$ is a list of distinct vertices in $V$ and is called a *variable*. An item in a variable is called a *port*. And a term graph $g$ has a vertex labeling $\varphi_g : V \rightarrow \Sigma$, an edge labeling $\psi_g : E \rightarrow \Lambda$ and a variable labeling $\lambda_g : H \rightarrow X$. For a set or a list $S$, the number of elements in $S$ is denoted by $|S|$. The *dimension* of a variable $h$ is the number of vertices which are

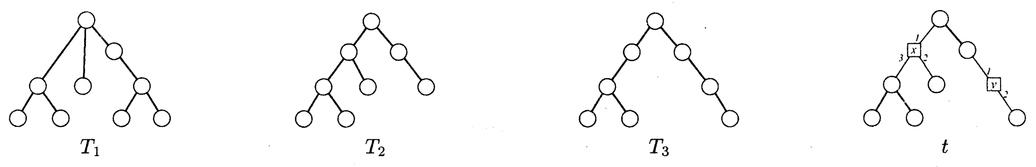


**Fig. 1.** A term tree $t$ as a tree-like structured pattern which matches trees $T_1$, $T_2$ and $T_3$.

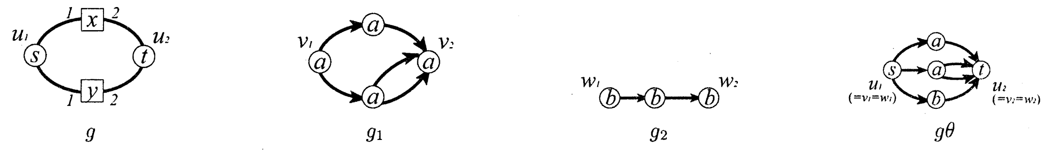


**Fig. 2.** A term graph $g = (V, E, H)$ is defined by $V = \{u_1, u_2\}$, $E = \emptyset$, $H = \{e_1 = (u_1, u_2), e_2 = (u_1, u_2)\}$, $\varphi_g(u_1) = s$, $\varphi_g(u_2) = t$, $\lambda_g(e_1) = x$, and $\lambda_g(e_2) = y$. $g\theta$ is obtained by applying a substitution $\theta = \{x := [g_1, (v_1, v_2)], y := [g_2, (w_1, w_2)]\}$ to $g$. A variable is represented by a box with lines to its elements and the order of its items is indicated by the numbers at these lines.

contained in $h$. The *dimension* of a term graph $g$ is the maximum dimension over all variables in $g$. The *degree* of a vertex $u$ in a term graph is the sum of the number of edges and variables containing $u$. The *degree* of a term graph $g$ is the maximum degree over all vertices in $g$. For example, a term graph $g = (V, E, H)$ is shown in Fig. 2.

In [7], for a regular term tree $t$ and a tree $T$ such that the dimension of each variable in $t$ is exactly 2 and the degree of $T$ is unbounded, we presented a polynomial time algorithm solving the matching problem for $t$ and $T$. In this paper, we show that, in general, the matching problem for a regular term tree $t$ and a tree $T$ is NP-complete even if the dimension of $t$ is only 4. But, if the dimension of $t$ is bounded by some constant greater than 1 and also the degree of $T$ is bounded by some constant, we can give a polynomial time algorithm solving the matching problem. These show that a term tree is a quite useful expression of knowledge obtained from tree-like structured data. Our algorithms lead us to develop new knowledge discovery tools employing term graphs directly which express knowledge obtained from tree-like structured data.

## 2 Matching Algorithms for Tree Structured Patterns

Let $g$ be a term graph and $\sigma$ a list of distinct vertices in $g$. We call the form $x := [g, \sigma]$ a *binding* for a variable label $x \in X$. Let $g_1, \ldots, g_n$ be term graphs. A *substitution* $\theta$ is a finite collection of bindings $\{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\}$, where $x_i$'s are mutually distinct variable labels in $X$ and each $g_i$ has no variable labeled with an element in $\{x_1, \ldots, x_n\}$. We obtain a new term graph $f$ by applying a substitution $\theta = \{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\}$ to a term graph $g = (V, E, H)$ in the following way. For each binding $x_i := [g_i, \sigma_i] \in \theta$ $(1 \le i \le n)$ in parallel, we attach $g_i$ to $g$ by removing all variables $t_1, \ldots, t_k$ labeled with $x_i$ from $H$, and by identifying the $m$-th vertex $t_j^m$ of $t_j$ and the $m$-th vertex $\sigma_i^m$ of $\sigma_i$ for each $1 \le j \le k$ and each $1 \le m \le |t_j| = |\sigma_i|$. We remark that the label of each vertex $t_j^m$ of $g$ is used for the resulting term graph which is denoted by $g\theta$. Namely, the label of $\sigma_i^m$ is ignored in $g\theta$.

A substitution $\theta = \{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\}$ is called a *tree substitution* if all of the $g_i$ are trees. A term graph $t$ is called a *term tree* if for any tree substitution $\theta$ which contains all variable labels in $t$, $t\theta$ is also a tree. When one of the vertices in $t$ is specified as the root of the term tree, $t$ is called a *term rooted tree*. A term tree $t$ is said to be *regular* if each variable label in $t$ occurs exactly once [6] (e.g. Fig. 3,4). We say that $T$ matches $t$ if there exists a tree substitution $\theta$ such that $t\theta$ and $T$ are isomorphic.

In this section, we consider the following problem for a regular term rooted tree with no label. For a regular term rooted tree with labels and a regular term unrooted tree with labels or without any label, we can also construct similar polynomial-time matching algorithms.

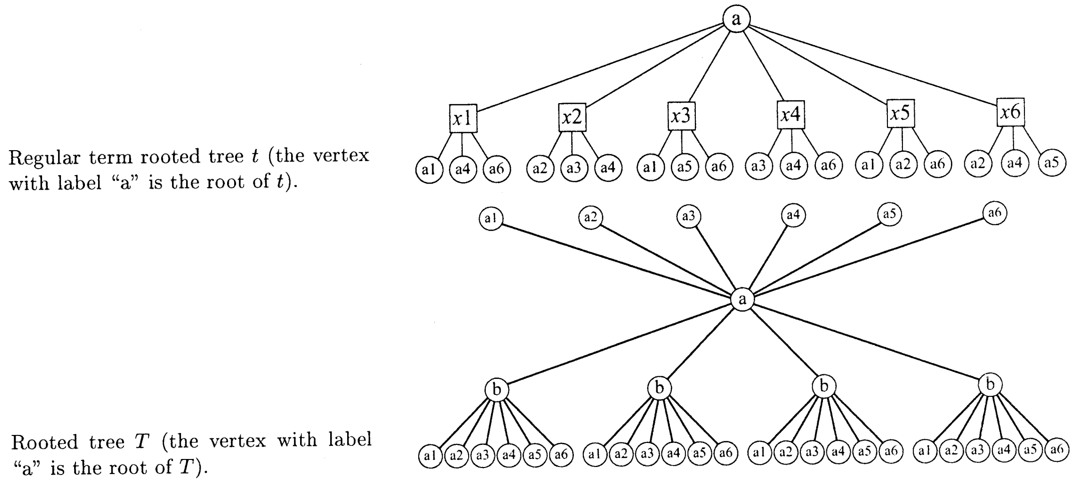

Regular term rooted tree $t$ (the vertex with label "a" is the root of $t$).
a
x1
x2
x3
x4
x5
x6
a1 a4 a6
a2 a3 a4
a1 a5 a6
a3 a4 a6
a1 a2 a6
a2 a4 a5
Rooted tree $T$ (the vertex with label "a" is the root of $T$).
a1 a2 a3 a4 a5 a6
a
b
b
b
b
a1 a2 a3 a4 a5 a6

**Fig. 3.** A transformation from an instance $(A, C)$ of X3C to an instance $(t, T)$ of REGULAR TERM ROOTED TREE MATCHING. $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}, C = \{c_1, c_2, c_3, c_4, c_5, c_6\}, c_1 = \{a_1, a_4, a_6\}, c_2 = \{a_2, a_3, a_4\}, c_3 = \{a_1, a_5, a_6\}, c_4 = \{a_3, a_4, a_6\}, c_5 = \{a_1, a_2, a_6\}, c_6 = \{a_2, a_4, a_5\}$.

**REGULAR TERM ROOTED TREE MATCHING**
**Instance**: A regular term rooted tree $t$ and a rooted tree $T$.
**Question**: Does $T$ match $t$ so that the root of $T$ corresponds to the root of $t$?

First we show the following theorem.

**Theorem 1.** REGULAR TERM ROOTED TREE MATCHING *is NP-complete even if the dimension of an input regular term rooted tree is only* 4.

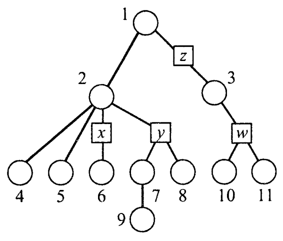
Membership in NP is obvious. We transform EXACT COVER BY 3-SETS (X3C) [5, page 221] to this problem (Fig. 3).

Second we explain the algorithm Matching (Fig. 5) which is a framework for deciding whether a rooted tree $T$ matches a regular term rooted tree $t$.

Let $t = (V_t, E_t, H_t)$ and $T = (V_T, E_T)$ be a regular term rooted tree with root $r_t$ and a rooted tree with root $r_T$, respectively. A vertex of degree one is called a *leaf* if it is not the root. A *path* from $v_1$ to $v_i$ is a sequence $v_1, v_2, \ldots, v_i$ of distinct vertices such that for $1 \leq j < i$, there exists an edge or a variable which includes $v_j$ and $v_{j+1}$. If there is an edge or a variable which includes $v$ and $v'$ such that $v'$ lies on the path from the root $r_t$ to $v$, then $v'$ is said to be the *father* of $v$ and $v$ is a *child* of $v'$. In particular for a variable $h$, $v$ is said to be a *child port* of $h$ if there is a vertex $v'$ such that both $v$ and $v'$ belong to $h$ and $v$ is a child of $v'$. A *descendant* of $v$ is any vertex on the path from $v$ to one of the leaves of the tree.

In Matching (Fig.5), a *label* for a vertex in $T$ is a set $\{v_1, \ldots, v_k, V_1, \ldots, V_\ell\}$ where $k \geq 0$, $\ell \geq 0$, $v_i$ is a vertex in $t$, and $V_j$ is a set of vertices in $t$. Let $L_1, \ldots, L_m$ be a collection of labels. For any $V' \subseteq V_t$, we say that $L_1, \ldots, L_m$ *covers* $V'$ if there exist distinct indices $k_1, \ldots, k_{m'}, \ell_1, \ldots, \ell_{m''}$ among $1, \ldots, m$ and also there exist $v'_i \in L_{k_i}$ and $V''_j \in L_{\ell_j}$ for each $1 \leq i \leq m'$ and $1 \leq j \leq m''$ such that $V' \subseteq \{v'_1, \ldots, v'_{m'}\} \cup V''_1 \cup \cdots \cup V''_{m''}$. In particular if there is no proper subcollection of $L_1, \ldots, L_m$ which covers $V'$ then we say that $L_1, \ldots, L_m$ *exactly covers* $V'$.

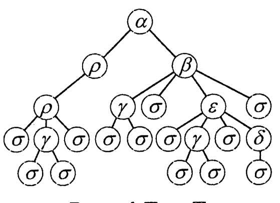
Let $W$ be a set of vertices in $t$. The *induced term tree* of $t$ by $W$ is a term tree $t[W] = (W', E_t[W'], H_t[W'])$ where $W' = \{v \in V_t \mid v$ is in $W$ or there is a vertex $v'$ in $W$ such that $v$ is a descendant of $v'.\}$, $E_t[W'] = \{\{u, v\} \in E_t \mid u \in W'$ and $v \in W'\}$ and $H_t[W'] = \{(v_1, \ldots, v_n) \mid v_i \in W'$ and $(v_1, \ldots, v_n)$ is the maximal sublist of some $h \in H_t$ with keeping the order of items in $h.\}$. For a single vertex $w \in V_t$, the induced term tree $t[w]$ of $t$ by $w$ is defined as $t[\{w\}]$. A *corresponding induced term tree of $t$ to $u \in V_T$* is an induced term tree by $W \subseteq V_t$ or $w \in V_t$ which



Term Rooted Tree $t$

$1 \Leftarrow 2, \{3\}$
$2 \Leftarrow 4, 5, \{6\}, \{7, 8\}$
$3 \Leftarrow \{10, 11\}$
$7 \Leftarrow 9$

Labeling Rules $R_1$



Rooted Tree $T$

$\sigma = \{4, 5, 6, 8, 9, 10, 11\}$
$\delta = \{7, \{6\}, \{8\}, \{10, 11\}\}$
$\gamma = \{3, \{6\}, \{7, 8\}, \{10, 11\}\}$
$\rho = \{\{3\}, \{6\}, \{7, 8\}, \{10, 11\}\}$
$\varepsilon = \{2, \{3\}, \{6\}, \{7, 8\}, \{10, 11\}\}$
$\beta = \{1, 2, \{3\}, \{6\}, \{7, 8\}, \{10, 11\}\}$
$\alpha = \{1, \{3\}, \{6\}, \{7, 8\}, \{10, 11\}\}$

The resulting labels by $R_1$

**Fig. 4.** An example: the labeling rule constructed from a regular term rooted tree $t = (\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}, \{\{1, 2\}, \{2, 4\}, \{2, 5\}, \{7, 9\}\}, \{(1, 3), (2, 6), (2, 7, 8), (3, 10, 11)\})$ with root 1 and the resulting labels of a rooted tree $T$ after the procedure Matching terminates.

```
procedure Matching(regular term rooted tree $t$ with root $r_t$, rooted tree $T$ with root $r_T$);
begin
    Construct the set of all labeling rules $R_{r_t}$;
    Label all leaves of $T$ with the set of all leaves of $t$;
    while there exists a vertex $v$ in $T$
                    such that $v$ is not labeled and all children of $v$ are labeled do
        Labeling($v, R_{r_t}$);
    if the label of $r_T$ includes $r_t$ then $T$ matches $t$ else $T$ does not match $t$
end.
```

**Fig. 5.** An algorithm for deciding whether or not a rooted tree $T$ matches a regular term rooted tree $t$.

matches $T[u]$, i.e., the subtree of $T$ with the root $u$. In particular if the induced term tree is induced by a single vertex $w$, the matching between $t[w]$ and $T[u]$ has a correspondence of $w$ to $u$.

First we construct the set of all labeling rules.

**Basic Labeling Rules**

Let $v$ be a vertex in $t$ which is not a leaf. Let $v_1, v_2, \ldots, v_k$ be all children of $v$ which are connected to $v$ with edges. Let $h_1, h_2, \ldots, h_\ell$ be all variables which include $v$, and for $i = 1, \ldots, \ell$, $V_i$ be the set of all children of $v$ which are connected to $v$ with the variable $h_i$. The labeling rule for $v$ is defined as follows. If there is no variable which includes $v$, then let the generating rule of $v$ be $v \leftarrow v_1, \ldots, v_k$, otherwise $v \Leftarrow v_1, \ldots, v_k, V_1, \ldots, V_\ell$.

Then we obtain the following theorem.

**Theorem 2.** *Let $t$ be a regular term rooted tree $t$ of dimension $p$ such that $t$ includes no variable of dimension* 1, *and let $T$ be a rooted tree of degree $d$.* REGULAR TERM ROOTED TREE MATCHING *is solvable in $O((n + N)^2 \mathrm{TMIS}(d^{O(p)}))$ time where $n$ and $N$ are the numbers of vertices in $t$ and $T$ respectively, and* TMIS$(s)$ *is the time needed to find the maximum independent set in a graph of size $s$.*

**Corollary 1.** *Let $t$ be a regular term rooted tree of dimension $p$ such that $t$ includes no variable of dimension* 1, *and let $T$ be a rooted tree of degree $d$.* REGULAR TERM ROOTED TREE MATCHING *for the inputs $t$ and $T$ is solvable in $O((n+N)^2)$ time where $n$ and $N$ are the numbers of vertices in $t$ and $T$ respectively.*

If the dimension of each variable in $t$ is exactly 2 and the degree of $T$ is unbounded, the procedure Matching by using the procedure (Fig.8) instead of the procedure (Fig.7) solves REGULAR TERM ROOTED TREE MATCHING in polynomial time.

**Theorem 3 (Miyahara, et al. [7]).** *If the dimension of each variable in $t$ is exactly* 2, *there exists a polynomial-time algorithm for solving* REGULAR TERM ROOTED TREE MATCHING.

Since the maximum graph matching for a bipartite graph $\mathcal{B} = (\mathcal{V}, \mathcal{V}', \mathcal{E})$ is found in $O(|\mathcal{E}|\sqrt{\max\{|\mathcal{V}|, |\mathcal{V}'|\}})$ time, by applying the procedure (Fig.8) to all labeling rules at the same

```
procedure Labeling(vertex $u \in V_T$, set of labeling rules $R_{r_t}$);
begin
  $L := \emptyset$;
  Let $m$ be the number of children of $u$ and $L_1, \ldots, L_m$ be the labels of the children;
  /* Step 1 */
  foreach $v \leftarrow v_1, \ldots, v_m$ in $R_{r_t}$ do
    if $L_1, \ldots, L_m$ exactly covers $\{v_1, \ldots, v_m\}$ then $L := L \cup \{v\}$;
  /* Step 2 */
  foreach $v \Leftarrow v_1, \ldots, v_k, V_1, \ldots, V_\ell$ in $R_{r_t}$ do
    if $L_1, \ldots, L_m$ covers $\{v_1, \ldots, v_k\} \cup V_1 \cup \cdots \cup V_\ell$ then $L := L \cup \{v\}$;
  /* Step 3 */
  foreach variable $h$ in $t$ do begin
    Let $V'$ be the set of all child ports of $h$;
    foreach $V'' \subseteq V'$ with $V'' \neq \emptyset$ do begin
      foreach $v \in V' - V''$ do
        if $v$ and $V''$ satisfy either
          (1) $v$ is a leaf and $V''$ is a maximal subset such that $L_1, \ldots, L_m$
            covers $V''$, or
          (2) $v$ is the head of a rule $v \Leftarrow v_1, \ldots, v_k, V_1, \ldots, V_\ell$ in $R_{r_t}$
            and $V''$ is a maximal subset such that $L_1, \ldots, L_m$ covers
            $\{v_1, \ldots, v_k\} \cup V_1 \cup \cdots \cup V_\ell \cup V''$.
        then $L := L \cup \{V'' \cup \{v\}\}$;
      if there is no vertex $v$ which satisfies either (1) or (2) and $V''$ is a
        maximal subset which is covered by $L_1, \ldots, L_m$
      then $L := L \cup \{V''\}$
    end
  end;
  Attach $L$ to $u$ as the label
end;
```

**Fig. 6.** Labeling: a procedure for labeling a vertex in $T$ with a set of vertices in $t$.

```
Input: labels $L_1, \ldots, L_m$, vertices $v_1, \ldots, v_k$, and sets of vertices $V_1, \ldots, V_\ell$;
Note that each $V_i$ $(i = 1, \ldots, \ell)$ is the set of all child ports of a certain variable and each label $L_j$ $(j = 1, \ldots, m)$ contains at most one subset of $V_i$. If the input term rooted tree $t$ is of bounded dimension and the input rooted tree $T$ is of bounded degree, the size of the graph $\mathcal{G}$ constructed below is bounded because both $k$ and $\ell$ are bounded by some constants and the size of each $L_i$ is also bounded.
begin
  Construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in the following way:
    $\mathcal{V}_i := \{(v_i, \{j\}) \mid v_i \in L_j \ (1 \leq j \leq m)\}$,
    Let $\mathcal{E}_i$ be the complete graph constructed by $\mathcal{V}_i$,
    /* In the following statement, because a subset of $V_i$ appears at most once in each $L_j$, it is easy to decide
      whether or not $L_{j_1}, \ldots, L_{j_{m'}}$ covers $V_i$. */
    $\mathcal{V}'_i := \{(V_i, \{j_1, \ldots, j_{m'}\}) \mid \{j_1, \ldots, j_{m'}\}$ is a subset of $\{1, \ldots, m\}$ such that $L_{j_1}, \ldots, L_{j_{m'}}$ covers $V_i$.$\}$.
    Let $\mathcal{E}'_i$ be the complete graph constructed by $\mathcal{V}'_i$,
    $\mathcal{V} := \bigcup_{i=1}^{k} \mathcal{V}_i \ \cup \ \bigcup_{i=1}^{\ell} \mathcal{V}'_i$,
    $\mathcal{E} := \{\{(X, Y), (X', Y')\} \mid (X, Y), (X', Y') \in \mathcal{V}, \ X \neq X'$ and $Y \cap Y' \neq \emptyset\} \cup \bigcup_{i=1}^{k} \mathcal{E}_i \cup \bigcup_{i=1}^{\ell} \mathcal{E}'_i$.
  if there is an independent set of size $k + \ell$ for the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ then
    $L_1, \ldots, L_m$ covers $\{v_1, \ldots, v_k\} \cup V_1 \cup \cdots \cup V_\ell$
end;
```

**Fig. 7.** A procedure for determining whether or not $L_1, \ldots, L_m$ covers $\{v_1, \ldots, v_k\} \cup V_1 \cup \cdots \cup V_\ell$ (Step 2 and Step 3 (2)).

```
**Input:** labels $L_1, \dots, L_m$, vertices $v_1, \dots, v_k$, and sets of vertices $\{v'_1\}, \dots, \{v'_\ell\}$;
**begin**
  Construct a bipartite graph $\mathcal{B} = (\mathcal{V}, \mathcal{V}', \mathcal{E})$ in the following way:
    $\mathcal{V} := \{v_1, \dots, v_k, v'_1, \dots, v'_\ell\}, \ \mathcal{V}' := \{1, \dots, m\}$,
    $\mathcal{E}_i := \{\{v_i, j\} \mid v_i \in L_j \ (1 \le j \le m)\}$                  $(i = 1, \dots, k)$,
    $\mathcal{E}'_i := \{\{v'_i, j\} \mid v'_i \in L_j$ or $\{v'_i\} \in L_j \ (1 \le j \le m)\}$   $(i = 1, \dots, \ell)$,
    $\mathcal{E} := \bigcup_{i=1}^{k} \mathcal{E}_i \cup \bigcup_{i=1}^{\ell} \mathcal{E}'_i$.
  **if** for the bipartite graph $(\mathcal{V}, \mathcal{V}', \mathcal{E})$, there exists a graph matching which
     contains all vertices in $\mathcal{V}$
  **then** $L_1, \dots, L_m$ covers $\{v_1, \dots, v_k\} \cup \{v'_1\} \cup \dots \cup \{v'_\ell\}$
**end**;
```

**Fig. 8.** A procedure for determining whether or not $L_1, \dots, L_m$ covers $\{v_1, \dots, v_k\} \cup \{v'_1\} \cup \dots \cup \{v'_\ell\}$ (for an input regular term rooted tree such that the dimension of each variable is exactly 2).

time, the label of $u$ can be computed in $O(n^2 \cdot \deg(u)\sqrt{n \cdot \deg(u)})$ where $\deg(u)$ is the degree of $u$. Then the total time for Matching used in Theorem 3 is $O(\sum_{u \in V_T} n^2 \cdot \deg(u)\sqrt{n \cdot \deg(u)})$. This does not exceed $O((n+N)^4)$.

The complexity of REGULAR TERM ROOTED TREE PROBLEM is still open if the dimension of an input regular term rooted tree is 3.

## 3 Conclusions

We have given an algorithmic foundation of discovering knowledge from tree structured data. We have presented polynomial time matching algorithms for tree structured patterns. Computational experiments of comparing our matching algorithm and a naive matching algorithm have shown that our matching algorithm is efficient and useful. We will incorporate the matching algorithm in the KD-FGS system and other knowledge discovery systems from tree-like structured data.

## References

1. S. Džeroski. Inductive logic programming and knowledge discovery in databases. *Advances in Knowledges Discovery and Data Mining, MIT Press*, pages 118–152, 1996.
2. S. Džeroski, N. Jacobs, M. Molina, C. Moure, S. Muggleton, and W. V. Laer. Detecting traffic problems with ILP. *Proc. ILP-98, LNAI 1446*, pages 281–290, 1998.
3. L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3:7–36, 1999.
4. L. Dehaspe, H. Toivonen, and R. King. Finding frequent substructures in chemical compounds. *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining*, pages 30–36, 1998.
5. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.
6. S. Matsumoto, Y. Hayashi, and T. Shoudai. Polynomial time inductive inference of regular term tree languages from positive data. *Proc. ALT-97, LNAI 1316*, pages 212–227, 1997.
7. 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, LNAI 1634*, pages 222–233, 1999.
8. T. Miyahara, T. Uchida, T. Kuboyama, T. Yamamoto, K. Takahashi, and H. Ueda. KD-FGS: a knowledge discovery system from graph data using formal graph system. *Proc. PAKDD-99, LNAI 1574*, pages 438–442, 1999.
9. Y. Mukouchi and S. Arikawa. Towards a mathematical theory of machine discovery from facts. *Theoretical Computer Science*, 137:53–84, 1995.
10. H. Toivonen. On knowledge discovery in graph-structured data. *Proc. the PAKDD Workshop on Knowledge Discovery from Advanced Databases (KDAD-99)*, 1999.
11. T. Uchida, T. Shoudai, and S. Miyano. Parallel algorithm for refutation tree problem on formal graph systems. *IEICE Transactions on Information and Systems*, E78-D(2):99–112, 1995.