IPSJ Transactions on Mathematical Modeling and Its Applications Vol.10 No.2 59–69 (July 2017)

Regular Paper

Enumeration of Maximally Frequent Ordered Tree Patterns with Wildcards for Edge Labels

Tetsuhiro Miyahara[1,a)] Yusuke Suzuki[1,b)] Takayoshi Shoudai[2,c)]
Tomoyuki Uchida[1,d)] Tetsuji Kuboyama[3,e)]

Received: January 30, 2017, Revised: March 20, 2017,
Accepted: April 17, 2017

Abstract: We consider representing tree structured features of structured data which are represented by rooted trees with ordered children. As representations of tree structured features, we use ordered tree patterns, called ordered wildcard tree patterns, which have structures of rooted ordered trees, structured variables and wildcards for edge labels. A structured variable can be replaced with an arbitrary rooted ordered tree. First we show that it is hard to compute two types of optimum frequent ordered wildcard tree patterns. Then we present an algorithm for enumerating all maximally frequent ordered wildcard tree patterns. Finally we consider extended ordered wildcard tree patterns, called ordered tag tree patterns, which have structured variables, wildcards, tags and keywords, and present an algorithm for enumerating all maximally frequent ordered tag tree patterns.

Keywords: ordered tree pattern, enumeration algorithm, tree structured feature

1. Introduction

As the amount of tree structured data has increased, the modeling of tree structured features common to given tree structured data has been more and more important. So we investigate new models for representing tree structured features. In this paper, we consider models of tree structured features in two aspects, i.e., representing power of tree structured patterns and the desired properties that the tree structured patterns must satisfy.

Tree structured data which we consider in this paper are semistructured data whose structures are modeled by rooted trees with ordered children, based on Object Exchange Model (OEM, for short) [1]. Among tree structured data we consider are XML files, some biological data such as the secondary structure data of RNA or glycan data, and parse trees in natural language processing. For example, in **Fig. 1**, the rooted ordered tree T_1 represents the structure which the XML file *xml_sample* has.

As a model of tree structured features we propose *wildcard tree patterns*, which are ordered tree patterns with structured variables and wildcards, and match whole trees. A structured variable can be replaced with an arbitrary rooted ordered tree and a wildcard matches any edge label. Since a variable can be replaced with an arbitrary tree and a wildcard matches any edge label, overgeneralized patterns which satisfy the mere *frequency* and explain given data are meaningless. Then, in order to model tree structured features common to given tree structured data better it is necessary to find a wildcard tree pattern t which satisfies *maximal frequency*, in the sense that t can explain more data of given tree structured data than a user-specified threshold but any wildcard tree pattern more specific than t cannot. In this work, the maximal frequency of wildcard tree patterns is the desired property that the tree structured patterns must satisfy. That is, we need to find maximally frequent (or least generalized) wildcard tree patterns. For example, consider finding one of the least generalized wildcard tree patterns explaining at least two trees in $\{T_1, T_2, T_3\}$ where T_1,T_2 and T_3 are trees in Fig. 1. The wildcard tree pattern t_1 in Fig. 1 can explain all trees in $\{T_1, T_2, T_3\}$, that is trees T_1,T_2 and T_3 are obtained from t_1 by replacing the variable of t_1 with a tree. But t_1 can explain all trees, so t_1 is an overgeneralized and meaningless pattern. On the other hand, the wildcard tree pattern t_2 in Fig. 1 is one of the least generalized wildcard tree patterns explaining two trees T_1 and T_3 but not T_2. For example in Fig. 1, T_1 is obtained from t_2 by replacing the variable between vertices u_1 and u_3 with the tree g_1, and the variable between vertices u_4 and u_9 with the tree g_2, and by replacing wildcards with the corresponding edge labels.

In this paper, we consider three computational problems, **Maximally Frequent Ordered Wildcard Tree Pattern of Maximum Tree-size**, **Maximally Frequent Ordered Wildcard Tree Pattern of Minimum Variable-size**, and **All Maximally Frequent Ordered Wildcard Tree Patterns** over wildcard tree patterns. Maximally Frequent Ordered Wildcard Tree Pattern of Maximum Tree-size is the problem of finding the maximum wild-

1 Graduate School of Information Sciences, Hiroshima City University, Hiroshima 731–3194, Japan
2 Faculty of Contemporary Business, Kyushu International University, Kitakyushu, Fukuoka 805–8512, Japan
3 Computer Centre, Gakushuin University, Toshima, Tokyo 171–8588, Japan
a) miyares17@info.hiroshima-cu.ac.jp
b) y-suzuki@info.hiroshima-cu.ac.jp
c) shoudai@cb.kiu.ac.jp
d) uchida@info.hiroshima-cu.ac.jp
e) ori-mps17@tk.cc.gakushuin.ac.jp

IPSJ Transactions on Mathematical Modeling and Its Applications Vol.10 No.2 59–69 (July 2017)

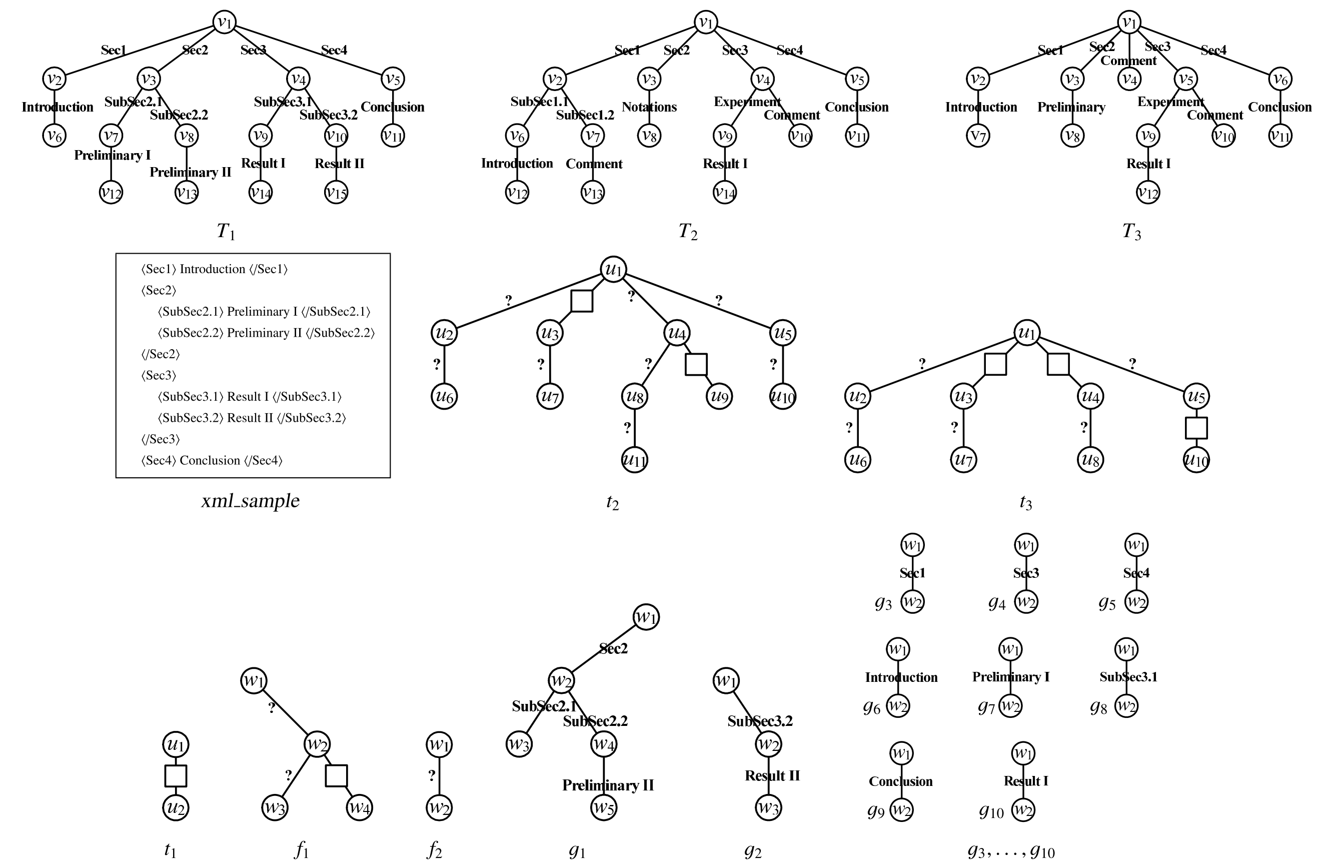


Fig. 1 An XML file *xml_sample* and a rooted ordered tree T_1 as its tree representation. g_1 and g_2 are trees. $g_3, \ldots, g_{10}$ are word trees. A variable is represented by a box with lines to its elements. A wildcard tree pattern t_1 explains trees T_1, T_2, and T_3. A wildcard tree pattern t_2 is one of the least generalized wildcard tree patterns which explain trees T_1 and T_3 but not T_2. The wildcard tree pattern t_2 is maximally σ-frequent w.r.t. $\mathcal{D} = \{T_1, T_2, T_3\}$, where $\sigma = 0.5$. A wildcard tree pattern t_3 explains trees T_1 and T_3 but not T_2. The wildcard tree pattern t_3 is σ-frequent but not maximally σ-frequent w.r.t. $\mathcal{D}$.

card tree pattern t with respect to the number of vertices such that t can explain more data of input data than a user-specified threshold and t is minimally generalized. This problem is based on the idea that the wildcard tree pattern, which has more vertices than any other wildcard tree patterns, gives more meaningful tree structured features to us. In a similar motivation, we consider the second problem Maximally Frequent Ordered Wildcard Tree Pattern of Minimum Variable-size, which is the problem of finding the minimum wildcard tree pattern t with respect to the number of variables such that t can explain more data of input data than a user-specified threshold and t is minimally generalized. Firstly, we show that Maximally Frequent Ordered Wildcard Tree Pattern of Maximum Tree-size and Maximally Frequent Ordered Wildcard Tree Pattern of Minimum Variable-size are NP-complete. This indicates that it is hard to find an optimum wildcard tree pattern representing given data. Next, we consider All Maximally Frequent Ordered Wildcard Tree Patterns, which is the problem of generating all maximally frequent wildcard tree patterns. This problem is based on the idea that meaningless wildcard tree patterns are excluded and all possible useful wildcard tree patterns are not missed. We present an algorithm for solving All Maximally Frequent Ordered Wildcard Tree Patterns, i.e., an algorithm for enumerating maximally frequent wildcard tree patterns, and show the correctness of the algorithm.

Since wildcard tree patterns consist of only structured variables and edges with wildcards for edge labels, maximally frequent wildcard tree patterns capture tree structured features with emphasized views on tree structures only. As an extended model, from wildcard tree patterns, of tree structured features, we propose *tag tree patterns*, which are ordered tree patterns with structured variables, wildcards, tags and keywords, and match whole trees. Since tag tree patterns contain tags and keywords, maximally frequent tag tree patterns, as a kind of structured keywords, capture tree structured features with emphasis on reflecting users views. So we propose two types of tree structured patterns, i.e., wildcard tree patterns and tag tree patterns in order to reflect two different views.

Finally, as an application of the algorithm for solving All Maximally Frequent Ordered Wildcard Tree Patterns, we present an algorithm for solving All Maximally Frequent Ordered Tag Tree Patterns, which is the problem of enumerating all maximally frequent tag tree patterns.

We discuss related work. As knowledge representations for tree structured data, a tree-expression pattern [16] and a regular path expression [5] were proposed. In Ref. [16], Wang and Liu presented the algorithm for finding maximally frequent tree-expression patterns from tree structured data. In Ref. [2], Asai et al. presented an efficient algorithm for finding frequent substructures from a large collection of tree structured data. In Ref. [5], Fernandez and Suciu presented the algorithm for finding optimal regular path expressions from tree structured data. Recent research on tree structure patterns are reported [3], [4], [7], [15]. Our wildcard tree patterns and tag tree patterns are different from the above mentioned representations, in that our tree patterns have structured variables which can be replaced with arbitrary trees, and match whole trees.

In our previous work [8], [10], [12], we considered maximally frequent tree patterns with unordered children, contractible variables, and height-constrained variables, all of which are different from wildcard tree patterns and tag tree patterns. In Ref. [13], we considered finding a minimally generalized tree pattern, that is, a least generalized tree pattern of frequency 1.0, from tree structured data with many edge labels or with no edge label. Finding a minimally generalized tree pattern from tree structured data with no edge label has theoretical importance in learning theory. In this paper we focus on practical aspects of tree structured patterns, and consider finding tree structured features, which are represented by maximally frequent wildcard tree patterns, from tree structured data with many edge labels. In Ref. [14], we gave an efficient pattern matching algorithm for ordered term tree patterns, the extended algorithms of which we use in this paper for calculating the matching relation of wildcard tree patterns and trees, and the matching relation of tag tree patters and trees. The work [6] gave an algorithm for enumerating all maximal tree patterns, which are different tree patterns of frequency 1.0. This paper is a complete version of our previous results on tag tree patterns [9], and presents newly introduced wildcard tree patterns, full descriptions of improved algorithms and full proofs.

This paper is organized as follows. In Section 2, we introduce wildcard tree patterns as tree structured patterns. In Section 3, we show that Maximally Frequent Ordered Wildcard Tree Pattern of Maximum Tree-size and Maximally Frequent Ordered Wildcard Tree Pattern of Minimum Variable-size are NP-complete. In Section 4, we give an algorithm for solving All Maximally Frequent Ordered Wildcard Tree Patterns and show its correctness. In Section 5, as an application, we give an algorithm for solving All Maximally Frequent Ordered Tag Tree Patterns and show its correctness. In Section 6, we conclude this paper.

2. Preliminaries

2.1 Ordered Wildcard Tree Patterns as Tree Structured Patterns

We explain ordered wildcard tree patterns as tree structured patterns. Let Λ be a language which consists of infinitely or finitely many words. Let "?" be a special symbol, called a *wildcard*, such that "?" $\notin \Lambda$. Let $\Lambda_{\{?\}}$ be a proper subset of Λ. The set $\Lambda_{\{?\}}$ means the set of all words which represent contents and the set $\Lambda \setminus \Lambda_{\{?\}}$ means the set of all words which do not represent contents, for example, the set of all words containing escape sequences such as carriage returns and tab movements. In the case where finitely many escape sequences are used, there exists an algorithm for deciding whether or not any word in Λ is in $\Lambda_{\{?\}}$. The symbol "?" is a wildcard for any word in $\Lambda_{\{?\}}$. For a set S, the number of elements in S is denoted by $|S|$. In this paper, a *tree* means a rooted ordered tree with ordered children such that each edge is labeled with an element in Λ.

Definition 1 Let $T = (V_T, E_T)$ be a tree which has a set V_T of vertices and a set E_T of edges. Let E_g and H_g be a partition of E_T, i.e., $E_g \cup H_g = E_T$ and $E_g \cap H_g = \emptyset$. And let $V_g = V_T$. An *ordered wildcard tree pattern* (or simply called a *wildcard tree pattern*) is a triplet $g = (V_g, E_g, H_g)$ such that each element of E_g is labeled with the symbol "?". Each element in V_g, E_g and H_g is called a *vertex*, an *edge* and a *variable*, respectively.

For a wildcard tree pattern g and its vertices v_1 and v_i, a *path* from v_1 to v_i is a sequence $v_1, v_2, \ldots, v_i$ of distinct vertices of g such that for any j with $1 \leq j < i$, there exists an edge or a variable which consists of v_j and v_{j+1}. If there is an edge or a variable which consists of v and v' such that v lies on the path from the root to v', then v is said to be the *parent* of v' and v' is a *child* of v. We use a notation (v, v') (resp. $[v, v']$) to represent an edge (resp. a variable) such that v is the parent of v'. Then we call v the *parent port* of $[v, v']$ and v' the *child port* of $[v, v']$. A wildcard tree pattern g has a total ordering on all children of every internal vertex u. The ordering on the children of u is denoted by $<_u^g$.

Definition 2 $\mathcal{OT}$ denotes the set of all trees whose edge labels are in Λ. $\mathcal{OWTP}$ denotes the set of all wildcard tree patterns.

A tree T is a *word tree* if $|V_T| = 2$ and $|E_T| = 1$. For a word $w \in \Lambda$, $T(w)$ denotes the word tree whose edge is labeled with the word w. For a subset $\Lambda' \subsetneq \Lambda$, we define the set of word trees $\mathcal{WT}_{\Lambda'} = \bigcup_{w \in \Lambda'} \{T(w)\}$. Note that for any set $\Lambda' \subsetneq \Lambda$, $\mathcal{WT}_{\Lambda'} \subsetneq \mathcal{OT}$.

Let $f = (V_f, E_f, H_f)$ and $g = (V_g, E_g, H_g)$ (resp. $f = (V_f, E_f)$ and $g = (V_g, E_g)$) be two wildcard tree patterns (resp. two trees). We say that f and g are *isomorphic*, denoted by $f \cong g$, if there is a bijection, called an isomorphism, φ from V_f to V_g such that (1) the root of f is mapped to the root of g by φ, (2) $(u, v) \in E_f$ if and only if $(\varphi(u), \varphi(v)) \in E_g$ and the two edges have the same edge label, (3) $[u, v] \in H_f$ if and only if $[\varphi(u), \varphi(v)] \in H_g$, and (4) for any internal vertex u in f which has more than one child, and for any two children u' and u'' of u, $u' <_u^f u''$ if and only if $\varphi(u') <_{\varphi(u)}^g \varphi(u'')$.

Let g be a wildcard tree pattern or a tree with at least two vertices. Let $\sigma = [w_0, w_1]$ be a list of two distinct vertices in g where w_0 is the root of g and w_1 is a leaf of g. Let f be a wildcard tree pattern with at least two vertices and e a variable or an edge of f. The form $e := [g, \sigma]$ is called a *binding* for e. A new wildcard tree pattern or a new tree f' is obtained by apply the binding $e := [g, \sigma]$ for f in the following way. Let $e = [v_0, v_1]$ (resp. $e = (v_0, v_1)$) be a variable (resp. an edge) in f. Let g' be one copy of g and w'_0, w'_1 the vertices of g' corresponding to w_0, w_1 of g, respectively. For the variable or the edge e, we attach g' to f by removing e from $E_f \cup H_f$ and by identifying the vertices v_0, v_1 with the vertices w'_0, w'_1 of g', respectively. Further we define a new total ordering $<_u^{f'}$ on every vertex u of f' in a natural way. Suppose that u has more than one child and let u' and u'' be two children of u of f'. We have the following three cases. *Case 1*: If $u, u', u'' \in V_f$ and $u' <_u^f u''$, then $u' <_u^{f'} u''$. *Case 2*: If $u, u', u'' \in V_g$ and $u' <_u^g u''$, then $u' <_u^{f'} u''$. *Case 3*: If $u = v_0$, $u' \in V_g$, $u'' \in V_f$, and $v_1 <_u^f u''$ (resp. $u'' <_u^f v_1$), then $u' <_u^{f'} u''$ (resp. $u'' <_u^{f'} u'$). A *substitution* θ for f is a finite collection of

IPSJ Transactions on Mathematical Modeling and Its Applications Vol.10 No.2 59–69 (July 2017)

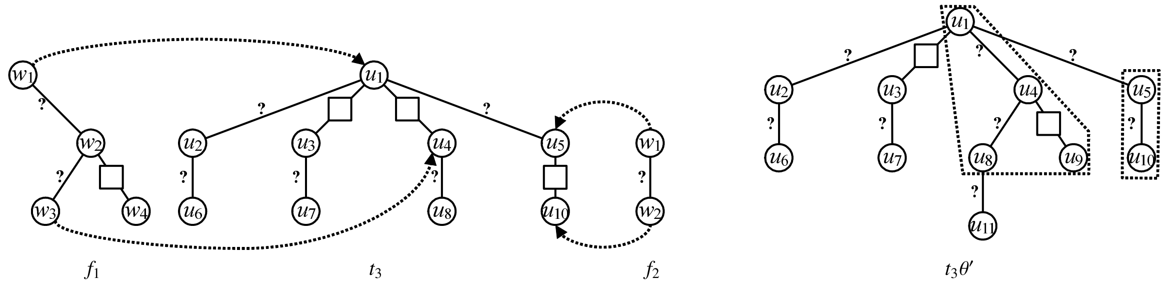


Fig. 2 Examples of $\mathcal{OWTP}$-bindings and $\mathcal{OWTP}$-substitution. Let t_3, f_1 and f_2 be wildcard tree patterns described in Fig. 1. The left figure displays the process of applying the $\mathcal{OWTP}$-bindings. The right figure displays the $\mathcal{OWTP}$-instance of t_3 by an $\mathcal{OWTP}$-substitution $\theta' = \{[u_1, u_4] := [f_1, [w_1, w_3]], [u_5, u_{10}] := [f_2, [w_1, w_2]]\}$.

bindings $\{e_1 := [g_1, \sigma_1], \ldots, e_n := [g_n, \sigma_n]\}$, where e_i's are mutually distinct variables or edges in f. The new wildcard tree pattern or the new tree $f\theta$, called the *instance* of f by θ, is obtained by applying the all bindings $e_i := [g_i, \sigma_i]$ to f simultaneously. We note that the root of $f\theta$ is the root of f.

For a variable e, a binding $e := [g, \sigma]$ is called an *$\mathcal{OWTP}$-binding* for e if $g \in \mathcal{OWTP}$. For a variable or an edge e, a binding $e := [g, \sigma]$ is called an *$\mathcal{OT}$-binding* for e if the following two conditions hold. (1) if e is an edge, then $g \in \mathcal{WT}_{\Lambda_{\{?\}}}$, (2) if e is a variable then $g \in \mathcal{OT}$. For a wildcard tree pattern f and a substitution $\theta = \{e_1 := [g_1, \sigma_1], \ldots, e_n := [g_n, \sigma_n]\}$ for f, θ is called an $\mathcal{OWTP}$-substitution for f if all bindings in θ are $\mathcal{OWTP}$-bindings, and θ is called an $\mathcal{OT}$-substitution for f if the following two conditions hold. (1) $\{e_1, \ldots, e_n\} = E_f \cup H_f$, (2) all bindings in θ are $\mathcal{OT}$-bindings. For an $\mathcal{OWTP}$-substitution θ, the new wildcard tree pattern $f\theta$ is called the *$\mathcal{OWTP}$-instance* of f by θ. For an $\mathcal{OT}$-substitution θ, the new tree $f\theta$ is called the *$\mathcal{OT}$-instance* of f by θ.

Example 1 Let t_2 and t_3 be two wildcard tree patterns described in Fig. 1. Let $[u_1, u_4] := [f_1, [w_1, w_3]]$ be an $\mathcal{OWTP}$-binding for the variable $[u_1, u_4]$ of t_3, and $[u_5, u_{10}] := [f_2, [w_1, w_2]]$ an $\mathcal{OWTP}$-binding for the variable $[u_5, u_{10}]$ of t_3, where f_1 and f_2 are wildcard tree patterns in Fig. 1. Let $\theta' = \{[u_1, u_4] := [f_1, [w_1, w_3]], [u_5, u_{10}] := [f_2, [w_1, w_2]]\}$ be an $\mathcal{OWTP}$-substitution for t_3. Then the $\mathcal{OWTP}$-instance $t_3\theta'$ of the wildcard tree pattern t_3 by θ' and the wildcard tree pattern t_2 are isomorphic. In **Fig. 2**, we describe the process of applying the above $\mathcal{OWTP}$-bindings in the $\mathcal{OWTP}$-substitution θ' for t_3 and the obtained wildcard tree pattern $t_3\theta'$ which is the $\mathcal{OWTP}$-instance of t_3 by θ'.
Let $\theta = \{[u_1, u_3] := [g_1, [w_1, w_3]], [u_4, u_9] := [g_2, [w_1, w_3]], (u_1, u_2) := [g_3, [w_1, w_2]], (u_1, u_4) := [g_4, [w_1, w_2]], (u_1, u_5) := [g_5, [w_1, w_2]], (u_2, u_6) := [g_6, [w_1, w_2]], (u_3, u_7) := [g_7, [w_1, w_2]], (u_4, u_8) := [g_8, [w_1, w_2]], (u_5, u_{10}) := [g_9, [w_1, w_2]], (u_8, u_{11}) := [g_{10}, [w_1, w_2]]\}$ be an $\mathcal{OT}$-substitution for t_2, where g_1, g_2 are trees and $g_3, \ldots, g_{10}$ are word trees in Fig. 1. Then the $\mathcal{OT}$-instance $t_2\theta$ of the wildcard tree pattern t_2 by θ and a tree T_1 in Fig. 1 are isomorphic.

A wildcard tree pattern t *matches* a tree T if there exists an $\mathcal{OT}$-substitution θ such that $t\theta \cong T$.

Definition 3 The *language* $L_\Lambda(t)$ of a wildcard tree pattern t is $\{s \in \mathcal{OT} \mid s \cong t\theta \text{ for an } \mathcal{OT}\text{-substitution } \theta\}$.

Let $\mathcal{D} = \{T_1, T_2, \ldots, T_m\} \subseteq \mathcal{OT}$ be a set of trees. The *matching count* of a wildcard tree pattern $\pi \in \mathcal{OWTP}$ w.r.t. $\mathcal{D}$, denoted by $match_{\mathcal{D}}(\pi)$, is the number of trees $T_i \in \mathcal{D}$ $(1 \le i \le m)$ such that π matches T_i. Then the *frequency* of π w.r.t. $\mathcal{D}$ is defined by $supp_{\mathcal{D}}(\pi) = match_{\mathcal{D}}(\pi)/m$. Let σ be a real number where $0 < \sigma \le 1$. A wildcard tree pattern π is *σ-frequent* w.r.t. $\mathcal{D}$ if $supp_{\mathcal{D}}(\pi) \ge \sigma$. A wildcard tree pattern π in $\mathcal{OWTP}$ is *maximally σ-frequent* w.r.t. $\mathcal{D}$ if (1) π is σ-frequent w.r.t. $\mathcal{D}$, and (2) if $L_\Lambda(\pi') \subsetneq L_\Lambda(\pi)$ then π' is not σ-frequent w.r.t. $\mathcal{D}$ for any wildcard tree pattern π' in $\mathcal{OWTP}$.

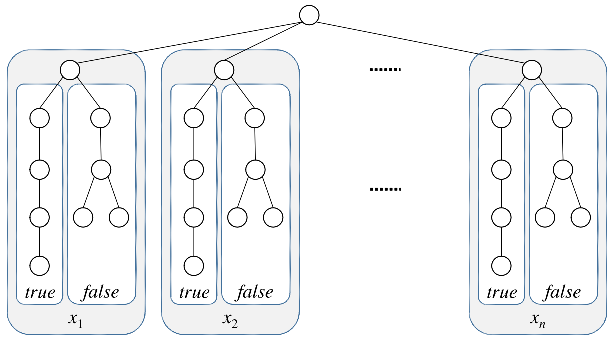


Fig. 3 Tree P_0.

3. Hardness Results of Finding an Optimum Frequent Wildcard Tree Pattern

In this section, we give hardness results of computing an optimum wildcard tree pattern. First we show that it is hard to compute a maximally frequent wildcard tree pattern of maximum tree-size w.r.t. a set of trees. The formal definition of the problem is as follows.

Maximally Frequent Ordered Wildcard Tree Pattern of Maximum Tree-size
Instance: A set of trees $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$, a real number σ $(0 < \sigma \le 1)$ and a positive integer K.
Question: Is there a maximally σ-frequent wildcard tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|V| \ge K$?

Theorem 1 **Maximally Frequent Ordered Wildcard Tree Pattern of Maximum Tree-size** is NP-complete.

Proof. Membership in NP is obvious. We transform 3-SAT to this problem. Let $U = \{x_1, \ldots, x_n\}$ be a set of variables and $C = \{c_1, \ldots, c_m\}$ a collection of clauses over U with $|c_j| = 3$ for any j $(1 \le j \le m)$. For a tree T and a vertex u of T, we denote the subtree consisting of u and the descendants of u by $T[u]$. Let P_0 be the tree which is described in **Fig. 3**. The root of P_0 has n chil-

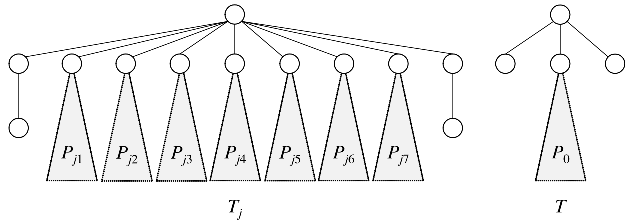


Fig. 4 Trees T_j and T.

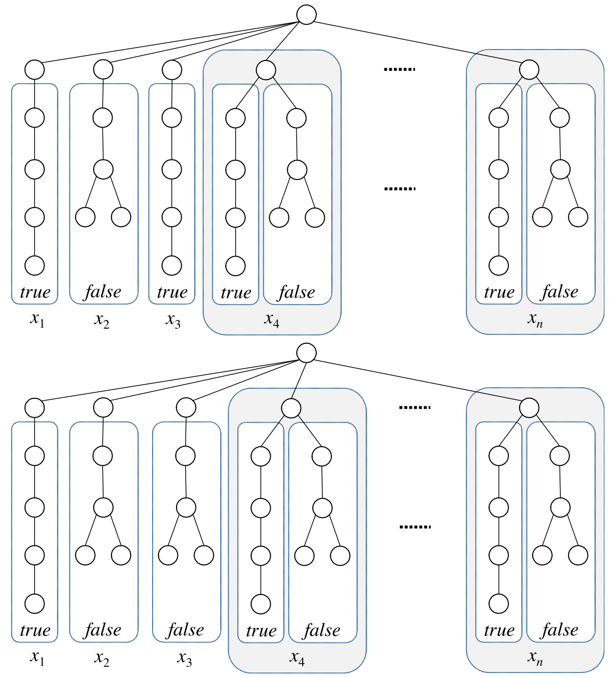


Fig. 5 Two examples of the truth assignments: The upper tree represents $(x_1, x_2, x_3) = (\mathit{true}, \mathit{false}, \mathit{true})$ and the lower shows $(x_1, x_2, x_3) = (\mathit{true}, \mathit{false}, \mathit{false})$.

dren. Let $v_1, v_2, \ldots, v_n$ be the n children. For each i $(1 \le i \le n)$, $P_0[v_i]$ corresponds to the truth assignment to x_i.

We construct trees $T_1, \ldots, T_m$ from the tree P_0 and $c_1, \ldots, c_m$ in the following way. T_j $(1 \le j \le m)$ is described in **Fig. 4**. The root of T_j has 9 children. Let $v_{j0}, v_{j1}, \ldots, v_{j8}$ be the 9 children. The inner 7 subtrees $T_j[v_{j1}], \ldots, T_j[v_{j7}]$ correspond to the truth assignments that satisfy c_j. Each $T_j[v_{ji}]$ $(1 \le i \le 7)$ is constructed as follows. Let $c_j = \{\ell_{j1}, \ell_{j2}, \ell_{j3}\}$ where $\ell_{jk} = x_{n_{jk}}$ or $\overline{x_{n_{jk}}}$ $(1 \le k \le 3,\ 1 \le n_{jk} \le n)$. The 7 truth assignments to $(x_{n_{j1}}, x_{n_{j2}}, x_{n_{j3}})$ make c_j *true*. For the ith truth assignment $(1 \le i \le 7)$ and all $1 \le n_{j1}, n_{j2}, n_{j3} \le n$, P_{ji} is obtained from P_0 by removing the right (resp. left) subtree rooted at $v_{n_{jk}}$ of P_0 if $x_{n_{jk}}$ is *true* (resp. *false*). This resulting tree P_{ji} becomes $T_j[v_{ji}]$. For example, the upper tree of **Fig. 5** represents a truth assignment $(x_1, x_2, x_3) = (\mathit{true}, \mathit{false}, \mathit{true})$.

Lastly let T be the special tree (Fig. 4) which is constructed from P_0. Let $\mathcal{D} = \{T_1, \ldots, T_m, T\}$, $\sigma = 1$, and $K = 5n + 4$. Then we can show the following two facts.

(1) Let π be a maximally σ-frequent wildcard tree pattern w.r.t. $\mathcal{D}$. Then the root of π has just three children and the second child of the three children has just n children.

(2) Let $G_1, G_2, G_3, g_1, g_2, g_3$ be trees and wildcard tree patterns described in **Fig. 6**, respectively. Then g_1 is maximally σ-frequent w.r.t. $\{G_1, G_2, G_3\}$, g_2 is maximally σ-frequent w.r.t. $\{G_1, G_3\}$, and g_3 is maximally σ-frequent w.r.t. $\{G_2, G_3\}$.

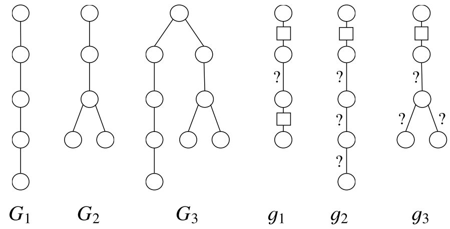


Fig. 6 Trees G_1, G_2, G_3 and wildcard tree patterns g_1, g_2, g_3.

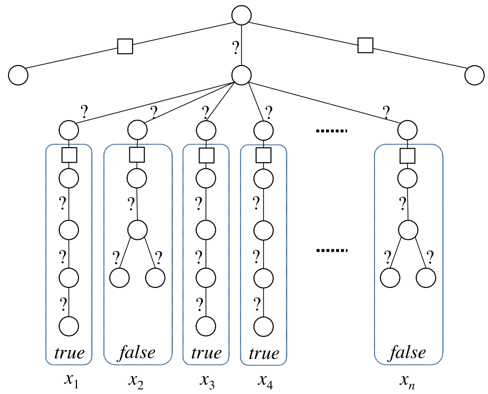


Fig. 7 An example of the wildcard tree patterns π such that π is $\sigma(=1)$-frequent w.r.t. $\mathcal{D}$.

From these two facts, if 3-SAT has a truth assignment which satisfies all clauses in C, there is a σ-frequent wildcard tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|V| = 5n + 4 = K$ (**Fig. 7**). Conversely, if there is a maximally σ-frequent wildcard tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|V| = 5n + 4$, the numbers of the children of the vertices of depth 5 show one of the truth assignment which satisfies C. □

Second we show that it is hard to compute a maximally frequent wildcard tree pattern of minimum variable-size w.r.t. a set of trees. The formal definition of the problem is as follows.

Maximally Frequent Ordered Wildcard Tree Pattern of Minimum Variable-size
Instance: A set of trees $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$, a real number σ $(0 < \sigma \le 1)$ and a positive integer K.
Question: Is there a maximally σ-frequent wildcard tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|H| \le K$?

Theorem 2 **Maximally Frequent Ordered Wildcard Tree Pattern of Minimum Variable-size** is NP-complete.

Proof. Membership in NP is obvious. The reduction is the same as the one in Theorem 1 but $K = n + 2$. □

4. Enumeration of Maximally Frequent Wildcard Tree Patterns

4.1 Enumeration Algorithm

In this section, we consider the following problem.

All Maximally Frequent Ordered Wildcard Tree Patterns (MFOWTP)
Input: A set of trees $\mathcal{D} \subsetneq \mathcal{OT}$, a real number σ $(0 < \sigma \le 1)$.
Assumption: (1) $\Lambda_{\{?\}} \subsetneq \Lambda$, and (2) there exists an algorithm for deciding whether or not any word in Λ is in $\Lambda_{\{?\}}$.
Problem: Enumerate all maximally σ-frequent wildcard tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OWTP}$.

Algorithm 1 GEN-MFOWTP

Input: A set $\mathcal{D} \subseteq \mathcal{OT}$ of trees and a real number σ ($0 < \sigma \leq 1$);

Output: The set $\Pi(\sigma)$ of all maximally σ-frequent wildcard tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OWTP}$;

/* Step1 Enumerate all σ-frequent variable-only tree patterns */
$\Pi_1(\sigma)$:=ENUMFREQTP($\mathcal{D}, \sigma$) (Procedure 2)
/* Step2 Enumerate all σ-frequent wildcard tree patterns */
$\Pi_2(\sigma)$:=REPLACEEDGE($\mathcal{D}, \sigma, \Pi_1(\sigma)$) (Procedure 4)
/* Step3 Maximality test */
$\Pi(\sigma)$:=TESTMAXIMALITY($\mathcal{D}, \sigma, \Pi_2(\sigma)$) (Procedure 6)
return $\Pi(\sigma)$

Procedure 2 ENUMFREQTP

Input: A set $\mathcal{D} \subseteq \mathcal{OT}$ of trees and a real number σ ($0 < \sigma \leq 1$);

Output: A set Π_{out} of variable-only tree patterns;

$\pi := (\{u, v\}, \emptyset, \{[u, v]\})$
Π_{out} :=ENUMFREQTPSUB($\mathcal{D}, \sigma, \pi$) (Procedure 3)
return Π_{out}

Procedure 3 ENUMFREQTPSUB

Input: A set $\mathcal{D} \subseteq \mathcal{OT}$ of trees, a real number σ ($0 < \sigma \leq 1$), and a variable-only tree pattern π;

Output: A set Π_{out} of variable-only tree patterns;

if π is not σ-frequent w.r.t. $\mathcal{D}$ **then**
  return $\emptyset$
end if
$\Pi_{out} := \{\pi\}$
for each child tree pattern π' of π **do**
  $\Pi_{out} := \Pi_{out} \cup$ENUMFREQTPSUB($\mathcal{D}, \sigma, \pi'$)
end for
return Π_{out}

Procedure 4 REPLACEEDGE

Input: A set $\mathcal{D} \subseteq \mathcal{OT}$ of trees, a real number σ ($0 < \sigma \leq 1$), and a set Π_{in} of variable-only tree patterns;

Output: A set Π_{out} of wildcard tree patterns;

$\Pi_{out} := \Pi_{in}$
for each wildcard tree pattern $\pi \in \Pi_{in}$ **do**
  $p := 1$
  /* p is an index of variables and edges of π in the DFS order */
  $\Pi_{out} := \Pi_{out} \cup$ REPLACEEDGESUB($\mathcal{D}, \sigma, \pi, p$) (Procedure 5)
end for
return Π_{out}

Procedure 5 REPLACEEDGESUB

Input: A set $\mathcal{D} \subseteq \mathcal{OT}$ of trees, a real number σ ($0 < \sigma \leq 1$), a wildcard tree pattern π and a positive integer p;

Output: A set Π_{out} of wildcard tree patterns;

if $p > |E_\pi \cup H_\pi|$ **then**
  return $\emptyset$
end if
$\Pi_{out} := \emptyset$
Let T_D be the wildcard tree pattern in **Fig. 9**.
Let h be the p-th variable in the DFS order of all edges and variables of π.
$\pi_? := \pi\{h := [T_D, [R_D, L_D]]\}$
if $\pi_?$ is σ-frequent w.r.t. $\mathcal{D}$ **then**
  $\Pi_{out} := \{\pi_?\}$
end if
$\Pi_{tmp} := \Pi_{out} \cup \{\pi\}$
for each wildcard tree pattern $\pi' \in \Pi_{tmp}$ **do**
  $\Pi_{out} := \Pi_{out} \cup$REPLACEEDGESUB($\mathcal{D}, \sigma, \pi', p+1$)
end for
return Π_{out}

Procedure 6 TESTMAXIMALITY

Input: A set $\mathcal{D} \subseteq \mathcal{OT}$ of trees, a real number σ ($0 < \sigma \leq 1$), and a set Π_{in} of wildcard tree patterns;

Output: A set Π_{out} of wildcard tree patterns;

$\Pi_{out} := \Pi_{in}$
Let T_A, T_B, T_C and T_D be the wildcard tree patterns in Fig. 9.
for each wildcard tree pattern $\pi \in \Pi_{out}$ **do**
  for each variable h in π **do**
    if there exists an $X \in \{A, B, C, D\}$ such that $\pi\{h := [T_X, [R_X, L_X]]\}$ is σ-frequent w.r.t. $\mathcal{D}$ **then**
      $\Pi_{out} := \Pi_{out} \setminus \{\pi\}$
    end if
  end for
end for
return Π_{out}

We give an algorithm GEN-MFOWTP (Algorithm 1) which generates all maximally σ-frequent wildcard tree patterns. Let $\mathcal{D} \subseteq \mathcal{OT}$ be a finite set of trees. In the algorithm GEN-MFOWTP, we can decide whether or not a candidate wildcard tree pattern is σ-frequent w.r.t. $\mathcal{D}$, by using a matching algorithm which decides whether or not a wildcard tree pattern matches a tree. This matching algorithm is an extended version of the efficient pattern matching algorithm [14] for an ordered term tree pattern and a tree, where an ordered term tree pattern is an ordered tree pattern having structured variables and labeled edges. In the matching algorithm for a wildcard tree pattern π and a tree T, we can decide whether or not π matches T by checking whether $\Lambda_{\{?\}}$ contains the label of the edge of T corresponding to an edge of π by using the algorithm in Assumption (2) of **MFOWTP**, instead of checking whether the two labels of corresponding edges of π' and T are the same in the matching algorithm for an ordered term tree pattern π' and a tree T. A *variable-only tree pattern* is a wildcard tree pattern consisting of only vertices and variables. We regard a variable-only tree pattern as a tree with the same tree structure. Asai et al. [2] presented a *rightmost expansion technique* over trees and an algorithm for enumerating all trees using the rightmost expansion technique, also developed in Refs. [11], [17]. In the following procedure ENUMFREQTP, we use this algorithm in order to enumerate all variable-only tree patterns by regarding a variable as an edge. For two variable-only tree patterns π and π', if π' is obtained from π by applying the rightmost expansion technique, then π' is called a *child tree pattern* of π and π is called the *parent tree pattern* of π'. An *enumeration tree over the set of all variable-only tree patterns* is a tree $\mathcal{T}_{enum}$ defined as follows. Each node of $\mathcal{T}_{enum}$ is a variable-only tree pattern. Let π and π' be two variable-only tree patterns which are nodes in $\mathcal{T}_{enum}$. Then there exists an edge from π to π' if and only if π' is a child tree pattern of π. The enumeration tree over the set of all variable-only tree patterns is illustrated in **Fig. 8**. By using the same parent-child relation as in Ref. [2], we can enumerate without any duplicate all variable-only tree patterns in a way of

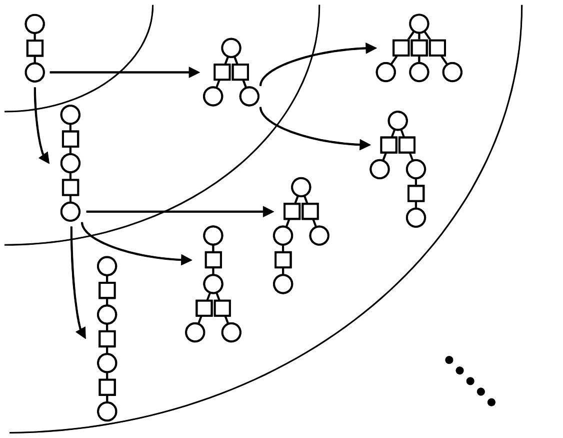


Fig. 8 The enumeration tree over the set of all variable-only tree patterns.

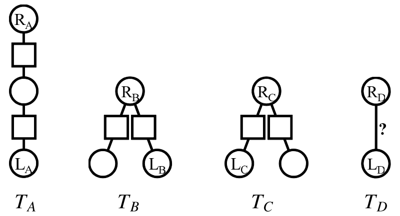


Fig. 9 Wildcard tree patterns T_X ($X \in \{A, B, C, D\}$).

depth first search from general to specific and backtracking. Although the semantics of matching of tree structured patterns and tree structured data is different from that in Ref. [2], a parent tree pattern π is more general than its child tree patterns π', that is $L_\Lambda(\pi') \subsetneq L_\Lambda(\pi)$, in generating process of variable-only tree patterns.

4.2 Correctness of the Enumeration Algorithm for Wildcard Tree Patterns

In this section, we show the correctness of Algorithm Gen-MFOWTP. Let Λ be a language which consists of infinitely or finitely many words, and $\Lambda_{\{?\}}$ a proper subset of Λ. For a wildcard tree pattern π, $V(\pi)$, $E(\pi)$, and $H(\pi)$ denote the vertex set, the edge set, and the variable set of π, respectively. For a tree T, $V(T)$ and $E(T)$ denote the vertex set and the edge set of T.

Lemma 1 Let π and π' be wildcard tree patterns. Then $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$ if and only if there is an $\mathcal{OWTP}$-substitution θ such that $\pi' \cong \pi\theta$.

Proof. (*If part*) Let T be a tree. If $T \in L_\Lambda(\pi')$, there is an $\mathcal{OT}$-substitution θ' such that $T \cong \pi'\theta'$. Since $\pi' \cong \pi\theta$, there is an isomorphism $\varphi : V(\pi') \to V(\pi\theta)$. Let θ'_φ be the $\mathcal{OT}$-substitution constructed from θ' by replacing all $\mathcal{OT}$-bindings $[u, v] := [g, [u', v']]$ and $(u, v) := [g, [u', v']]$ in θ' with $[\varphi(u), \varphi(v)] := [g, [u', v']]$ and $(\varphi(u), \varphi(v)) := [g, [u', v']]$, respectively. Since $T \cong \pi'\theta'$ and $\pi' \cong \pi\theta$, we have $T \cong (\pi\theta)\theta'_\varphi$. Let $\theta'' = \{[u, v] := [g\theta'_\varphi, [u', v']] \mid [u, v] := [g, [u', v']] \in \theta\} \cup \{(u, v) := [g\theta'_\varphi, [u', v']] \mid (u, v) := [g, [u', v']] \in \theta\} \cup \theta'_\varphi$. We see that $T \cong \pi\theta''$. Therefore $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$ holds.

(*Only-if part*) Let a be an edge label in $\Lambda \setminus \Lambda_{\{?\}}$ and b an edge label in $\Lambda_{\{?\}}$. Let $E(\pi') = \{e'_1, \ldots, e'_{m'}\}$ and $H(\pi') = \{h'_1, \ldots, h'_{\ell'}\}$. For each i ($1 \le i \le m'$), let $T'_i(b)$ be a copy of word tree $T(b)$ where $E(T'_i(b)) = \{(u'_i, v'_i)\}$, and for each i ($1 \le i \le \ell'$), let $T'_i(a)$ be a copy of word tree $T(a)$ where $E(T'_i(a)) = \{(u'_i, v'_i)\}$. Let θ_1 be an $\mathcal{OT}$-substitution defined as $\{e'_i := [T'_i(b), [u'_i, v'_i]] \mid 1 \le i \le m'\} \cup \{h'_i := [T'_i(a), [u'_i, v'_i]] \mid 1 \le i \le \ell'\}$. Since $\pi'\theta_1 \in L_\Lambda(\pi')$ and $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$, we have $\pi'\theta_1 \in L_\Lambda(\pi)$. Therefore there is an $\mathcal{OT}$-substitution θ_2 such that $\pi'\theta_1 \cong \pi\theta_2$. Let $E(\pi) = \{e_1, \ldots, e_m\}$ and $H(\pi) = \{h_1, \ldots, h_\ell\}$. We see that for all $e_i \in E(\pi)$ ($1 \le i \le m$), there are $\mathcal{OT}$-bindings $e_i := [T_i(b), [u_i, v_i]]$ in θ_2, where $T_i(b)$ is a copy of word tree $T(b)$ and $E(T_i(b)) = \{(u_i, v_i)\}$. For any $\mathcal{OT}$-binding $h_i := [t_i, [u_i, v_i]]$ ($1 \le i \le \ell$) where t_i is a tree with $u_i, v_i \in V(t_i)$, we make a new word tree pattern t'_i that is constructed from t_i by replacing all words a's and b's with variables and ?'s, respectively. Let θ be an $\mathcal{OWTP}$-substitution $\{h_i := [t'_i, [u_i, v_i]] \mid h_i := [t_i, [u_i, v_i]] \in \theta_2\}$. Then, we see that $\pi' \cong \pi\theta$ holds. □

Lemma 2 After the second step of Algorithm Gen-MFOWTP, $\Pi_2(\sigma)$ contains all σ-frequent wildcard tree patterns w.r.t. $\mathcal{D}$.

Proof. Let $K = \max\{|V(T)| \mid T \in \mathcal{D}\}$. At the first step, according to the enumeration tree $\mathcal{T}_{enum}$, Algorithm Gen-MFOWTP generates all σ-frequent variable-only tree patterns of tree-size at most K. Moreover, since the second step (Procedure ReplaceEdge) uses a brute-force method for replacing each variable of t in $\Pi_1(\sigma)$ with a wildcard edge, $\Pi_2(\sigma)$ contains all σ-frequent wildcard tree patterns of tree-size at most K. □

Theorem 3 Algorithm Gen-MFOWTP outputs the set of all maximally σ-frequent wildcard tree patterns w.r.t. $\mathcal{D}$.

Proof. The third step of Algorithm Gen-MFOWTP (Procedure TestMaximality) removes elements from $\Pi_2(\sigma)$ only. Therefore, from Lemma 2, any wildcard tree pattern in $\Pi(\sigma)$ is σ-frequent. Let π be a σ-frequent wildcard tree pattern in $\Pi(\sigma)$. We will prove that if there is a σ-frequent wildcard tree pattern π' w.r.t. $\mathcal{D}$ such that $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$, then $\pi \cong \pi'$ holds. Since $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$, from Lemma 1, there is an $\mathcal{OWTP}$-substitution θ such that $\pi' \cong \pi\theta$. Note that θ has only $\mathcal{OWTP}$-bindings for variables. We assume that there is an $\mathcal{OWTP}$-binding $h := [t, \sigma]$ in θ such that $|E(t)| + |H(t)| \ge 2$ or $|E(t)| \ge 1$. Since $L_\Lambda(\pi') = L_\Lambda(\pi\theta)$, if $|E(t)| + |H(t)| \ge 2$, $L_\Lambda(\pi') \subseteq L_\Lambda(\pi\{h := [T_X, [R_X, L_X]]\}) \subsetneq L_\Lambda(\pi)$ holds for some $X \in \{A, B, C\}$. If $|E(t)| = 1$ and $|H(t)| = 0$, $L_\Lambda(\pi') \subseteq L_\Lambda(\pi\{h := [T_D, [R_D, L_D]]\}) \subsetneq L_\Lambda(\pi)$ holds. This contradicts the fact that π is not removed from $\Pi(\sigma)$ in Procedure TestMaximality. Thus, $|E(t)| = 0$ and $|H(t)| = 1$ hold. Therefore, because the $\mathcal{OWTP}$-binding $h := [t, \sigma]$ is trivial, we can remove it from θ. In this way, we show that $\theta = \emptyset$ finally. Therefore, $\pi' \cong \pi$ holds. From this fact, we conclude that π is a maximally σ-frequent wildcard tree pattern w.r.t. $\mathcal{D}$. □

5. Application to Enumeration of Maximally Frequent Tree Patterns with Tags and Keywords

5.1 Enumeration of Maximally Frequent Tree Patterns with Tags and Keywords

Definition 4 (Ordered tag tree pattern) Let Λ_{Tag} be a language consisting of infinitely or finitely many words in Λ. Let Λ_{KW} be a language consisting of infinitely or finitely many words of the form "/k/" for words k in Λ, where we assume that "/" $\notin \Lambda$ holds. We call a word in Λ_{Tag} a *tag* and a word in

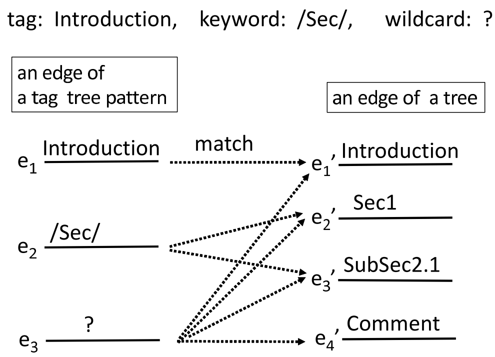


Fig. 10 The matching relation of an edge of a tag tree pattern and an edge of a tree.

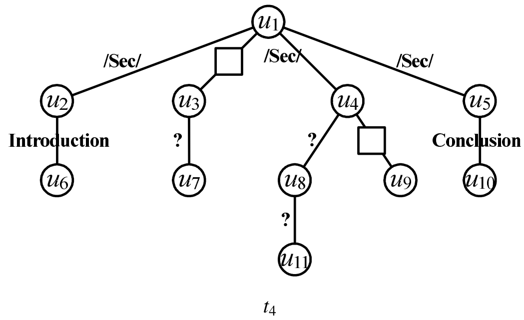


Fig. 11 A maximally σ-frequent tag tree pattern t_4 w.r.t. $\mathcal{D} = \{T_1, T_2, T_3\}$ given in Fig. 1, where $Tag = \{Introduction, Comment, Conclusion\}$, $KW = \{/Sec/, /SubSec/\}$ and $\sigma = 0.5$.

Λ_{KW} a *keyword*. For a keyword $/k/ \in \Lambda_{KW}$, we define the set $\Lambda_{\{/k/\}} = \{w \in \Lambda \mid k$ is a substring of $w\}$. Let $T = (V_T, E_T)$ be a tree which has a set V_T of vertices and a set E_T of edges. Let E_g and H_g be a partition of E_T, i.e., $E_g \cup H_g = E_T$ and $E_g \cap H_g = \emptyset$. And let $V_g = V_T$. An *ordered tag tree pattern* (or simply called a *tag tree pattern*) is a triplet $g = (V_g, E_g, H_g)$ such that each element in E_g is labeled with any of a tag, a keyword and the symbol "?". Each element in V_g, E_g and H_g is called a *vertex*, an *edge* and a *variable*, respectively.

Two tag tree patterns f and g are *isomorphic* if f and g are isomorphic as wildcard tree patterns (defined in Section 2) by regarding the symbol "?", tags and keywords as edge labels. A substitution for a tag tree pattern is an extended form of a substitution (defined in Section 2) for a wildcard tree pattern, where a binding $e := [g, \sigma]$ for an edge e labeled with a keyword $/k/$ can replace the edge e with any word tree $g \in \mathcal{WT}_{\Lambda_{\{/k/\}}}$, a binding $e := [g, \sigma]$ for an edge e labeled with the symbol "?" can replace the edge e with any word tree $g \in \mathcal{WT}_{\Lambda_{\{?\}}}$, and a binding $e := [g, \sigma]$ for a variable e can replace the variable e with any tag tree pattern or tree. A tag tree pattern t is said to *match* a tree T if there exists a substitution θ such that $T \cong t\theta$ holds. An edge e of a tag tree pattern is said to *match* an edge e' of a tree if there exists a substitution θ such that the edge label of e after the replacement by θ equals the edge label of e'. $\mathcal{OTTP}_{(\Lambda_{Tag}, \Lambda_{KW})}$ denotes the set of all tag tree patterns with tags in Λ_{Tag} and keywords in Λ_{KW}. For t in $\mathcal{OTTP}_{(\Lambda_{Tag}, \Lambda_{KW})}$, the *language* $L_\Lambda(t)$ is defined as {a tree T in $\mathcal{OT}$ | t matches T}.

Example 2 We explain the matching relation of an edge of a tag tree pattern and an edge of a tree in **Fig. 10**. Let "Introduction" be a tag, and "/Sec/" a keyword. We assume that "Introduction", "Sec1", "SubSec2.1" and "Comment" are all included in $\Lambda_{\{?\}}$. In a tag tree pattern, let us consider an edge e_1 with a label "Introduction", an edge e_2 with a label "/Sec/" and an edge e_3 with a label "?". In a tree, let us consider an edge e'_1 with a label "Introduction", an edge e'_2 with a label "Sec1", an edge e'_3 with a label "Sec2.1" and an edge e'_4 with a label "Comment". Then we have the following. e_1 matches e'_1. e_2 matches e'_2 and e'_3. e_3 matches e'_1, e'_2, e'_3 and e'_4.

Let $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$ $(m \geq 1)$ be a set of trees. The *matching count* of a tag tree pattern π w.r.t. $\mathcal{D}$, denoted by $match_{\mathcal{D}}(\pi)$, is the number of trees $T_i \in \mathcal{D}$ $(1 \leq i \leq m)$ such that π matches T_i. Then the *frequency* of π w.r.t. $\mathcal{D}$ is defined by $supp_{\mathcal{D}}(\pi) = match_{\mathcal{D}}(\pi)/m$. Let σ be a real number where $0 < \sigma \leq 1$. A tag tree pattern π is *σ-frequent* w.r.t. $\mathcal{D}$ if $supp_{\mathcal{D}}(\pi) \geq \sigma$. Let Tag be a finite subset of Λ_{Tag} and KW a finite subset of Λ_{KW}. Let $\Lambda(Tag, KW) = Tag \cup \bigcup_{/k/ \in KW} \Lambda_{\{/k/\}}$. We denote by $\mathcal{OTTP}(Tag, KW)$ the set of all tag tree patterns π with the tags of π in Tag and the keywords of π in KW. A tag tree pattern π in $\mathcal{OTTP}(Tag, KW)$ is *maximally σ-frequent* w.r.t. $\mathcal{D}$ if (1) π is σ-frequent, and (2) if $L_\Lambda(\pi') \subsetneq L_\Lambda(\pi)$ then π' is not σ-frequent for any tag tree pattern π' in $\mathcal{OTTP}(Tag, KW)$.

Example 3 Let t_4 be a tag tree pattern in $\mathcal{OTTP}(Tag, KW)$, which is described in **Fig. 11**, where we set $Tag = \{Introduction, Comment, Conclusion\}$ and $KW = \{/Sec/, /SubSec/\}$. The tag tree pattern t_4 is a maximally σ-frequent w.r.t. $\mathcal{D}$, where $\sigma = 0.5$, $\mathcal{D} = \{T_1, T_2, T_3\}$ given in Fig. 1. The tag tree pattern t_4 is more specific than the wildcard pattern t_2 in Fig. 1, that is, $L_\Lambda(t_4) \subsetneq L_\Lambda(t_2)$ holds.

All Maximally Frequent Ordered Tag Tree Patterns (MFOTTP)

Input: A set of trees $\mathcal{D} \subsetneq \mathcal{OT}$, a real number σ $(0 < \sigma \leq 1)$, a finite set Tag of tags, and a finite set KW of keywords.

Assumption: (1) $\Lambda(Tag, KW) \subsetneq \Lambda_{\{?\}} \subsetneq \Lambda$, (2) $Tag \cap \bigcup_{/k/ \in KW} \Lambda_{\{/k/\}} = \emptyset$, and (3) there exists an algorithm for deciding whether or not any word in Λ is in $\Lambda_{\{?\}}$.

Problem: Generate all maximally σ-frequent tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}(Tag, KW)$.

We give an algorithm Gen-MFOTTP (Algorithm 7) which generates all maximally σ-frequent tag tree patterns. Let $\mathcal{D} \subsetneq \mathcal{OT}$ be a finite set of trees. In the algorithm Gen-MFOTTP, we can decide whether or not a candidate tag tree pattern is σ-frequent w.r.t. $\mathcal{D}$, by using a matching algorithm which decides whether or not a tag tree pattern matches a tree. This matching algorithm is an extended version of the efficient pattern matching algorithm [14] for an ordered term tree pattern and a tree, and is similar to the matching algorithm for a wildcard tree pattern and a tree. In the matching algorithm for a tag tree pattern π and a tree T, we can decide whether or not π matches T by checking the following three cases (1)–(3), described in Fig. 10, of the matching relation of an edge e of π and its corresponding edge e' of T, instead of checking whether the two labels of corresponding edges of π' and T are the same in the matching algorithm [14] for an ordered term tree pattern π' and a tree T. (1) If e is labeled with a tag, then we check whether the two labels of e and e' are the same. (2) If e is labeled with a keyword $/k/$, then we check whether the label of e' is in $\Lambda_{\{/k/\}}$. (3) If e is labeled with the symbol "?", then we

Algorithm 7 GEN-MFOTTP

Input: A set $\mathcal{D} \subsetneq \mathcal{OT}$ of trees and a real number σ $(0 < \sigma \leq 1)$;
Output: The set $\Pi(\sigma)$ of all maximally σ-frequent tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}$;

```
/* Step1 Enumerate all σ-frequent variable-only tree patterns */
Π_1(σ) := EnumFreqTP(D, σ) (Procedure 2)
/* Step2 Enumerate all σ-frequent tag tree patterns */
Π_2(σ) := ReplaceEdge2(D, σ, Π_1(σ)) (Procedure 8)
/* Step3 Maximality test */
Π(σ) := TestMaximality2(D, σ, Π_2(σ)) (Procedure 10)
return Π(σ)
```

Procedure 8 REPLACEEDGE2

Input: A set $\mathcal{D} \subsetneq \mathcal{OT}$ of trees, a real number σ $(0 < \sigma \leq 1)$, and a set Π_{in} of variable-only tree patterns;
Output: A set Π_{out} of tag tree patterns;

```
Π_out := Π_in
for each tag tree pattern π ∈ Π_in do
    p := 1
    /* p is an index of variables and edges of π in the DFS order */
    Π_out := Π_out ∪ ReplaceEdgeSub2(D, σ, π, p) (Procedure 9)
end for
return Π_out
```

Procedure 9 REPLACEEDGESUB2

Input: A set $\mathcal{D} \subsetneq \mathcal{OT}$ of trees, a real number σ $(0 < \sigma \leq 1)$, a tag tree pattern π and a positive integer p;
Output: A set Π_{out} of tag tree patterns;

```
if p > |E_π ∪ H_π| then
    return ∅
end if
Π_out := ∅
Let T_D be the tag tree pattern in Fig. 12.
Let T_E(w) be the tag tree pattern in Fig. 12 for any keyword or tag w.
Let h be the p-th variable in the DFS order of all edges and variables of π.
π_? := π{h := [T_D, [R_D, L_D]]}
if π_? is σ-frequent w.r.t. D then
    Π_out := {π_?}
    for each keyword or tag w ∈ Tag ∪ KW do
        π_w := π{h := [T_E(w), [R_E, L_E]]}
        if π_w is σ-frequent w.r.t. D then
            Π_out := {π_w}
        end if
    end for
end if
Π_tmp := Π_out ∪ {π}
for each tag tree pattern π' ∈ Π_tmp do
    Π_out := Π_out ∪ ReplaceEdgeSub2(D, σ, π', p + 1)
end for
return Π_out
```

check whether the label of e' is in $\Lambda_{\{?\}}$ by using the algorithm in Assumption (3) of **MFOTTP**.

5.2 Correctness of the Enumeration Algorithm for Tag Tree Patterns

In this section, we show the correctness of Algorithm GEN-MFOTTP. For a tag tree pattern π, $V(\pi)$, $E(\pi)$, and $H(\pi)$ denote the vertex set, the edge set, and the variable set of π, respectively.

Procedure 10 TESTMAXIMALITY2

Input: A set $\mathcal{D} \subsetneq \mathcal{OT}$ of trees, a real number σ $(0 < \sigma \leq 1)$, and a set Π_{in} of tag tree patterns;
Output: A set Π_{out} of tag tree patterns;

```
Π_out := Π_in
Let T_A, T_B, T_C and T_D be the tag tree patterns in Fig. 12.
Let T_E(w) be the tag tree pattern in Fig. 12 for any keyword or tag w.
for each tag tree pattern π ∈ Π_out do
    for each variable h in π do
        if there exists an X ∈ {A, B, C, D} such that π{h := [T_X, [R_X, L_X]]}
        is σ-frequent w.r.t. D then
            Π_out := Π_out \ {π}
        end if
    end for
    for each edge e labeled with "?" in π do
        if there exists a keyword or a tag w ∈ Tag ∪ KW such that
        π{e := [T_E(w), [R_E, L_E]]} is σ-frequent w.r.t. D then
            Π_out := Π_out \ {π}
        end if
    end for
    for each edge e labeled with /k/ ∈ KW in π do
        if there exists a keyword /k'/ ∈ KW such that Λ_{/k'/} ⊋ Λ_{/k/} and
        π{e := [T_E(/k'/), [R_E, L_E]]} is σ-frequent w.r.t. D then
            Π_out := Π_out \ {π}
        end if
    end for
end for
return Π_out
```

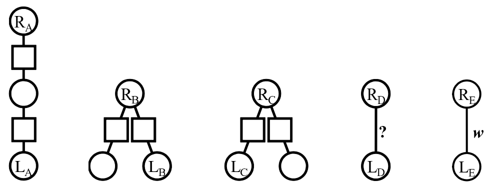


Fig. 12 Tag tree patterns T_X $(X \in \{A, B, C, D\})$ and a tag tree pattern $T_E(w)$.

Lemma 3 Let π and π' be tag tree patterns. Then $L_\Lambda(\pi') \subsetneq L_\Lambda(\pi)$ if and only if there is a substitution θ such that $\pi' \cong \pi\theta$.

Proof. (*If part*) Let T be a tree. If $T \in L_\Lambda(\pi')$, there is a substitution θ' such that $T \cong \pi'\theta'$. Since $\pi' \cong \pi\theta$, there is an isomorphism $\varphi : V(\pi') \to V(\pi\theta)$. Let θ'' be the substitution constructed from θ' and φ in the same way as the *if part* of Lemma 1. We see that $T \cong \pi\theta''$. Therefore $L_\Lambda(\pi') \subsetneq L_\Lambda(\pi)$ holds.

(*Only-if part*) Let a be an edge label in $\Lambda \setminus \Lambda_{\{?\}}$ and b an edge label in $\Lambda_{\{?\}} \setminus \Lambda(Tag, KW)$. We make a substitution θ_1 in the same way as the *only-if part* of Lemma 1. Let T be the tree obtained from $\pi'\theta_1$ by replacing all keyword edges labeled $/k/ \in KW$ with copies of word tree $T(k)$. Since $L_\Lambda(\pi') \subsetneq L_\Lambda(\pi)$, $T \in L_\Lambda(\pi)$ holds. Therefore, there is a substitution θ_2 such that $T \cong \pi\theta_2$. In a similar way to Lemma 1, we can construct the new substitution θ from θ_2 by replacing all edges labeled a with variables, all edges labeled b with wildcard edges, and all edges labeled k with keyword edges labeled $/k/$ for any $/k/ \in KW$. Finally, we see that $\pi' \cong \pi\theta$ holds. □

The following lemma can be proved in a similar way to Lemma 2.

Lemma 4 After the second step of Algorithm GEN-MFOTTP,

$\Pi_2(\sigma)$ is the set of all σ-frequent tag tree patterns w.r.t. $\mathcal{D}$.

Theorem 4 Algorithm Gen-MFOTTP outputs the set of all maximally σ-frequent tag tree patterns w.r.t. $\mathcal{D}$.

Proof. From Lemma 4, any tag tree pattern in $\Pi(\sigma)$ is σ-frequent. Let π be a σ-frequent tag tree pattern in $\Pi(\sigma)$. We will prove that if there is a σ-frequent tag tree pattern π' w.r.t. $\mathcal{D}$ such that $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$, then $\pi \cong \pi'$ holds. Since $L_\Lambda(\pi') \subseteq L_\Lambda(\pi)$, from Lemma 3, there is a substitution θ such that $\pi' \cong \pi\theta$. We assume that there is a binding $h := [t, \sigma]$ in θ such that $|E(t)| + |H(t)| \geq 2$ or $|E(t)| \geq 1$, where h is either a variable or an edge. Since $L_\Lambda(\pi') = L_\Lambda(\pi\theta)$, if $|E(t)| + |H(t)| \geq 2$, h is a variable and $L_\Lambda(\pi') \subseteq L_\Lambda(\pi\{h := [T_X, [R_X, L_X]]\}) \subsetneq L_\Lambda(\pi)$ holds for some $X \in \{A, B, C\}$. If $|E(t)| = 1$ and $|H(t)| = 0$, we have the following three cases: Let e is the unique edge in $E(t)$. (1) h is a variable and e is either a wildcard edge or an edge labeled with $w \in Tag \cup KW$, (2) h is a wildcard edge and e is labeled with $w \in Tag \cup KW$, and (3) h is an edge labeled with some $/k/ \in KW$ and e is labeled with keyword $/k'/ \in KW$ such that $\Lambda_{\{/k'/\}} \subsetneq \Lambda_{\{/k/\}}$. If either (1) or (2) holds, $L_\Lambda(\pi') \subseteq L_\Lambda(\pi\{h := [T_D, [R_D, L_D]]\}) \subsetneq L_\Lambda(\pi)$ holds, or there is a keyword or tag $w \in Tag \cup KW$ such that $L_\Lambda(\pi') \subseteq L_\Lambda(\pi\{h := [T_E(w), [R_E, L_E]]\}) \subsetneq L_\Lambda(\pi)$. This contradicts the fact that π is not removed from $\Pi(\sigma)$ at lines 4–14 in Procedure TestMaximality2. If the last case (3) holds, there is a keyword $/k'/ \in KW$ such that $L_\Lambda(\pi') \subseteq L_\Lambda(\pi\{h := [T_E(/k'/), [R_E, L_E]]\}) \subsetneq L_\Lambda(\pi)$. This contradicts the fact that π is not removed from $\Pi(\sigma)$ at lines 15–19 in Procedure TestMaximality2. Thus, $|E(t)| = 0$ and $|H(t)| = 1$ hold. If h is a variable, because the binding $h := [t, \sigma]$ is trivial, we can remove it from θ. If h is an edge, it contradicts the definition of the binding. In this way, we show that $\theta = \emptyset$ finally. Therefore, $\pi' \cong \pi$ holds. From this fact, we conclude that π is a maximally σ-frequent tag tree pattern w.r.t. $\mathcal{D}$. □

6. Conclusions

In this paper, we have considered the modeling of tree structured features of structured data which are represented by rooted trees with ordered children. As a model of tree structured features we have proposed wildcard tree patterns, which are ordered tree patterns with structured variables and wildcards, and match whole trees. A structured variable can be replaced with an arbitrary rooted ordered tree and a wildcard matches any edge label.

First we have shown that it is hard to compute a maximally frequent wildcard tree pattern of maximum-tree size and a maximally frequent wildcard tree pattern of minimum variable-size. Then we have presented an algorithm for enumerating all maximally frequent wildcard tree patterns. As an extended model, from wildcard tree patterns, of tree structured features, we have proposed tag tree patterns, which are ordered tree patterns with structured variables, wildcards, tags and keywords, and match whole trees. Finally, as an application of the former algorithm, we have presented an algorithm for enumerating all maximally frequent tag tree patterns.

Acknowledgments This work was partially supported by Grant-in-Aid for Scientific Research (C) (Grant Numbers 15K00312, 15K00313), Grant-in-Aid for Scientific Research (B) (Grant Number 26280087) from Japan Society for the Promotion of Science (JSPS), and Grant-in-Aid for Scientific Research on Innovative Areas (Grant Number 24106010) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan.

References

[1] Abiteboul, S., Buneman, P. and Suciu, D.: *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann (2000).

[2] Asai, T., Abe, K., Kawasoe, S., Sakamoto, H., Arimura, H. and Arikawa, S.: Efficient substructure discovery from large semi-structured data, *IEICE Trans. Inf. Syst.*, Vol.E87-D, No.12, pp.2754–2763 (2004).

[3] Chehreghani, M.H. and Bruynooghe, M.: Mining rooted ordered trees under subtree homeomorphism, *Data Mining and Knowledge Discovery*, Vol.30, No.5, pp.1249–1272 (2016).

[4] Doshi, M. and Roy, B.: Enhanced data processing using positive negative association mining on AJAX data, *Proc. 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications* (*CSCITA-2014*), pp.386–390 (2014).

[5] Fernandez, M. and Suciu, D.: Optimizing Regular Path Expressions Using Graph Schemas, *Proc. 14th International Conference on Data Engineering* (*ICDE-98*), pp.14–23, IEEE Computer Society (1998).

[6] Itokawa, Y., Uchida, T. and Sano, M.: An Algorithm for Enumerating All Maximal Tree Patterns Without Duplication Using Succinct Data Structure, *Proc. IMECS 2014*, pp.156–161 (2014).

[7] Jiang, C., Coenen, F. and Zito, M.: A survey of frequent subgraph mining algorithms, *The Knowledge Engineering Review*, Vol.28, No.01, pp.75–105 (2013).

[8] Miyahara, T., Shoudai, T., Uchida, T., Takahashi, K. and Ueda, H.: Discovery of Frequent Tree Structured Patterns in Semistructured Web Documents, *Proc. PAKDD-2001*, *LNAI 2035*, pp.47–52, Springer-Verlag (2001).

[9] Miyahara, T., Suzuki, Y., Shoudai, T., Uchida, T., Takahashi, K. and Ueda, H.: Discovery of Frequent Tag Tree Patterns in Semistructured Web Documents, *Proc. PAKDD-2002*, *LNAI 2336*, pp.341–355, Springer-Verlag (2002).

[10] Miyahara, T., Suzuki, Y., Shoudai, T., Uchida, T., Takahashi, K. and Ueda, H.: Discovery of Maximally Frequent Tag Tree Patterns with Contractible Variables from Semistructured Documents, *Proc. PAKDD-2004*, *LNAI 3056*, pp.133–134, Springer-Verlag (2004).

[11] Nakano, S.: Efficient generation of plane trees, *Information Processing Letters*, Vol.84, pp.167–172 (2002).

[12] Suzuki, Y., Miyahara, T., Shoudai, T., Uchida, T. and Nakamura, Y.: Discovery of Maximally Frequent Tag Tree Patterns with Height-Constrained Variables from Semistructured Web Documents, *Proc. International Workshop on Challenges in Web Information Retrieval and Integration* (*WIRI-2005*), pp.107–115 (2005).

[13] Suzuki, Y., Shoudai, T., Uchida, T. and Miyahara, T.: Ordered Term Tree Languages Which Are Polynomial Time Inductively Inferable from Positive Data, *Theoretical Computer Science*, Vol.350, No.1, pp.63–90 (2006).

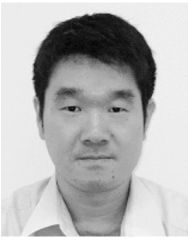
[14] Suzuki, Y., Shoudai, T., Uchida, T. and Miyahara, T.: An Efficient Pattern Matching Algorithm for Ordered Term Tree Patterns, *IEICE Trans. Inf. Syst.*, Vol.E98-A, No.6, pp.1197–1211 (2015).

[15] Wang, J., Liu, Z., Li, W. and Li, X.: Research on a frequent maximal induced subtrees mining method based on the compression tree sequence, *Expert Systems with Applications*, Vol.42, No.1, pp.94–100 (2015).

[16] Wang, K. and Liu, H.: Discovering structural association of semistructured data, *IEEE Trans. Knowledge and Data Engineering*, Vol.12, No.3, pp.353–371 (2000).

[17] Zaki, M.: Efficiently mining frequent trees in a forest: Algorithms and applications, *IEEE Trans. Knowledge and Data Engineering*, Vol.17, No.8, pp.1021–1035 (2005).

Tetsuhiro Miyahara is an associate professor of Graduate School of Information Sciences, Hiroshima City University, Hiroshima, Japan. He received his B.S. degree in Mathematics, M.S. and Dr.Sci. degrees in Information Systems all from Kyushu University, Fukuoka, Japan in 1984, 1986 and 1996, respectively. His research interests include algorithmic learning theory, knowledge discovery and machine learning.



Yusuke Suzuki received his B.S. degree in Physics, 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.

Takayoshi Shoudai received his B.S. and M.S. degrees in Mathematics, and Dr.Sci. degree in Information Science all from Kyushu University, in 1986, 1988, and 1993, respectively. Currently, he is a professor of Faculty of Contemporary Business, Kyushu International University. His research interests include graph algorithms, computational learning theory, and data mining. He is a member of IPSJ, ACM, and JSIAM.

Tomoyuki Uchida received his B.S. degree in Mathematics, 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.

Tetsuji Kuboyama received his B.Eng. and M.Eng. degrees from Kyushu University in 1992 and 1994 respectively, and his Ph.D. degree in 2007 from the University of Tokyo. He is currently a professor of the Computer Centre of Gakushuin University. He worked for the Center for Collaborative Research of the University of Tokyo as a research associate. His current research interests include pattern matching, data mining, and machine learning.