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

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

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 the 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 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 [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 match 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 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 by replacing the variable of t with a tree. But t can explain all trees, so t is an overgeneralized and meaningless pattern. On the other hand, the wildcard tree pattern t' 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' 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, **Frequent Ordered Wildcard Tree Pattern of Maximum Tree-size**, **Frequent Ordered Wildcard Tree Pattern of Minimum Variable-size**, and **All Maximally Ordered Wildcard Tree Patterns** over wildcard tree patterns. Frequent Ordered Wildcard Tree Pattern of Maximum Tree-size is the problem of finding the maximum wildcard 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. This problem is based on the idea that the wildcard tree pattern, which has more vertices than any other

[1] Graduate School of Information Sciences, Hiroshima City University, Hiroshima 731-3194, Japan
[2] Department of International Social Studies, Kyushu International University, Kitakyushu 805-8512, Japan
[3] Computer Centre, Gakushuin University, Tokyo 171-8588, Japan
[a)] miyares17@info.hiroshima-cu.ac.jp
[b)] y-suzuki@info.hiroshima-cu.ac.jp
[c)] shoudai@isb.kiu.ac.jp
[d)] uchida@info.hiroshima-cu.ac.jp
[e)] ori-mps17@tk.cc.gakushuin.ac.jp

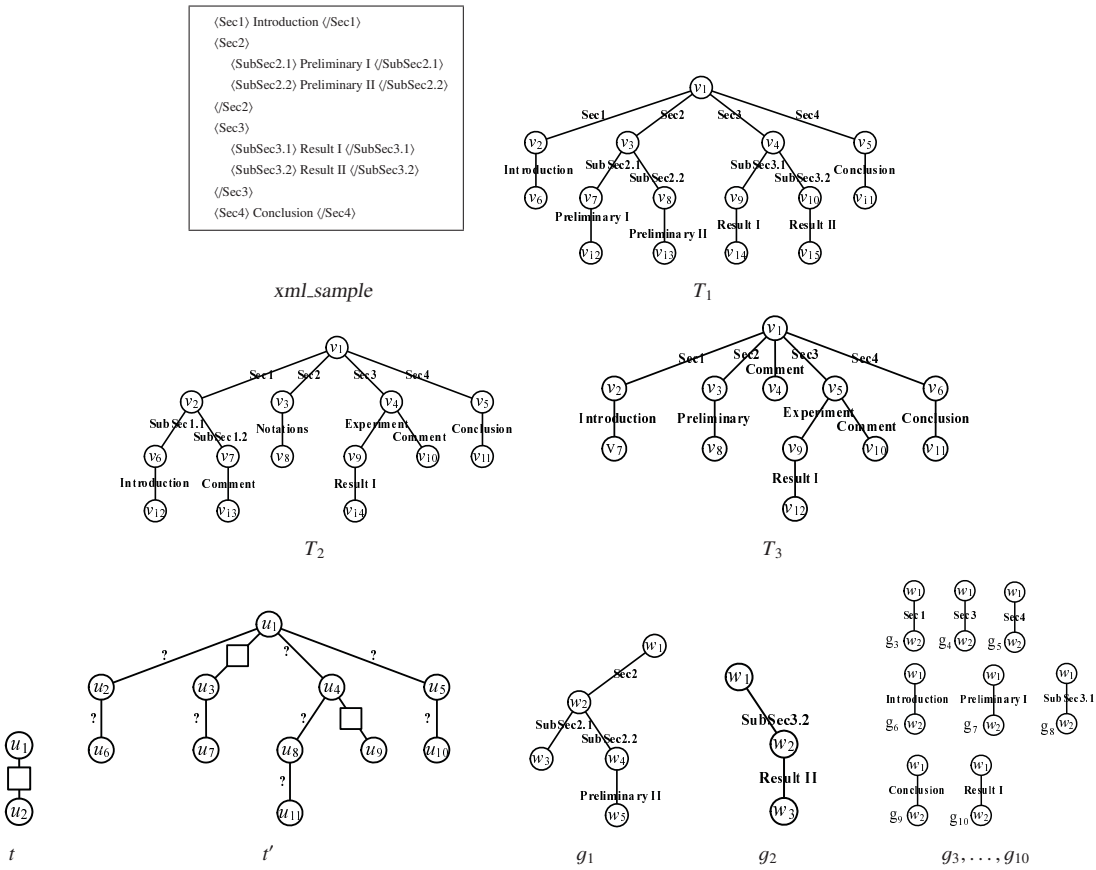

Fig. 1 An XML file *xml_sample* and a rooted ordered tree T_1 as its tree representation. g_1 and g_2 are trees. $g_3, \ldots, g_{10}$ are word trees. A variable is represented by a box with lines to its elements. A wildcard tree pattern t explains trees T_1, T_2, and T_3. A wildcard tree pattern t' 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' is a maximally σ-frequent w.r.t. $\mathcal{D} = \{T_1, T_2, T_3\}$, where $\sigma = 0.5$.

wildcard tree patterns, gives more meaningful tree structured features to us. In a similar motivation, we consider the second problem 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. Firstly, we show that Frequent Ordered Wildcard Tree Pattern of Maximum Tree-size and Frequent Ordered Wildcard Tree Pattern of Minimum Variable-size are NP-complete. This indicates that it is hard to find the optimum wildcard tree pattern representing given data. Next, we consider All Maximally 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 pattern are not missed. We present an algorithm for solving All Maximally Ordered Wildcard Tree Patterns, i.e., an algorithm for enumerating maximally frequent wildcard tree patterns. Finally, as an application of the algorithm for solving All Maximally Ordered Wildcard Tree Patterns, we consider an algorithm for solving All Maximally Frequent Ordered Tag Tree Patterns, which is the problem of enumerating all maximally frequent ordered tag tree patterns.

We discuss related work. Recent research on tree structure patterns are reported [3], [4], [6], [14]. Our ordered wildcard tree patterns and ordered 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 [7], [9], [11], we considered the concept of maximally frequent tree patterns with unordered children, contractible variables, height-constrained variables, all of which are different from wildcard tree patterns and ordered tag tree patterns. In [12], we considered finding a minimally generalized tree pattern, that is, a least generalized tree pattern with its frequency of 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 [13], 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 with trees, and the matching relation of tag tree patters with trees. The work [5] gave an algorithm for enumerating all maximal tree patterns, which are different tree patterns with the frequency of 1.0. This paper is a complete version of our previous results on ordered tag tree patterns [8].

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 *wild-card*, such that "?" $\notin \Lambda$. Let $\Lambda_{\{?\}}$ be a proper subset of Λ. 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 $<^g_u$.

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 $\Lambda' \subseteqq \Lambda$, we define the set of word trees $\mathcal{WT}_{\Lambda'} = \bigcup_{w \in \Lambda'} \{T(w)\}$. Note that for any set $\Lambda' \subseteqq \Lambda$, $\mathcal{WT}_{\Lambda'} \subsetneqq \mathcal{OT}$.

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 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 (iv) for any internal vertex u in f which has more than one child, and for any two children u' and u'' of u, $u' <^f_u u''$ if and only if $\varphi(u') <^g_{\varphi(u)} \varphi(u'')$.

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 $<^{f'}_u$ 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 $u' <^f_u u''$, then $u' <^{f'}_u u''$. *Case 2*: If $u, u', u'' \in V_g$ and $u' <^g_u u''$, then $u' <^{f'}_u u''$. *Case 3*: If $u = v_0$, $u' \in V_g$, $u'' \in V_f$, and $v_1 <^f_u u''$ (resp. $u'' <^f_u v_1$), then $u' <^{f'}_u u''$ (resp. $u'' <^{f'}_u u'$). A *substitution* θ for f is a finite collection of 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 form satisfies either of the following conditions. (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 and t' be two wildcard tree patterns described in Fig. 1. 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 a substitution for t', where g_1, g_2 are trees and $g_3, \ldots, g_{10}$ are word trees in Fig. 1. Then the $\mathcal{OT}$-instance $t'\theta$ of the wildcard tree pattern t' by θ and a tree T_1 are isomorphic in Fig. 1.

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 $\{s \in \mathcal{OT} \mid s \cong t\theta$ for an $\mathcal{OT}$-substitution $\theta\}$.

Let $\mathcal{D} = \{T_1, T_2, \ldots, T_m\} \subseteqq \mathcal{OT}$ 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 $T_i \in \mathcal{D}$ $(1 < i \leq m)$ 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 wildcard tree pattern π is σ**-frequent** w.r.t. $\mathcal{D}$ if $supp_{\mathcal{D}}(\pi) \geq \sigma$.

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

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

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 $(0 < \sigma \leq 1)$ and a positive integer K.
Question: Is there a σ-frequent wildcard tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|V| \geq K$?

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

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

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 $(0 < \sigma \leq 1)$ and a positive integer K.
Question: Is there a σ-frequent wildcard tree pattern $\pi = (V, E, H)$ w.r.t. $\mathcal{D}$ with $|H| \leq K$?

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

4. Enumeration of Maximally Frequent Wildcard Tree Patterns

4.1 Enumeration Algorithm

Let $\mathcal{D} = \{T_1, T_2, \ldots, T_m\} \subseteq \mathcal{OT}$ be a set of trees. 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 $\mathcal{OWTP}$.

All Maximally Frequent Ordered Wildcard Tree Patterns (MFOWTP)
Input: A set of trees $\mathcal{D} \subseteq \mathcal{OT}$, a real number σ $(0 < \sigma \leq 1)$.
Assumption: $\Lambda_{\{?\}} \subsetneq \Lambda$.
Problem: Enumerate all maximally σ-frequent wildcard tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OWTP}$.

We give an algorithm Gen-MFOWTP which generates all maximally σ-frequent wildcard tree patterns. Let $\mathcal{D} \in \mathcal{OT}$ be an input set of trees. 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 [10], [15]. 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 2. By using the same parent-child relation as in [2], we can enumerate without any duplicate all variable-only tree patterns in a way of 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 [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.

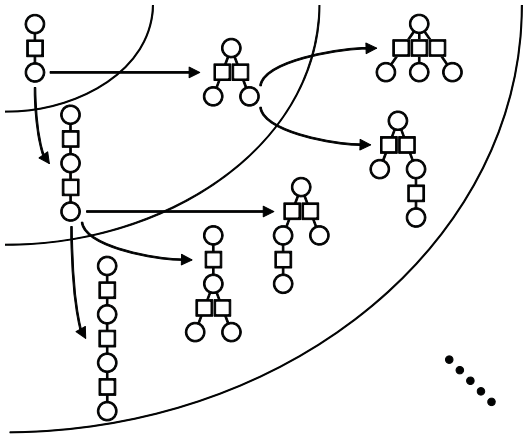

Fig. 2 The enumeration tree over the set of all variable-only tree patterns.

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 σ-frequent variable-only tree patterns */
1: $\Pi_1(\sigma)$:=EnumFreqTP($\mathcal{D}, \sigma$) (Procedure 2)
/* Step2 Enumerate all σ-frequent wildcard tree patterns */
2: $\Pi_2(\sigma)$:=ReplaceEdge($\mathcal{D}, \sigma, \Pi_1(\sigma)$) (Procedure 4)
/* Step3 Maximality test */
3: $\Pi(\sigma)$:=TestMaximality($\mathcal{D}, \sigma, \Pi_2(\sigma)$) (Procedure 6)
4: **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;

1: $\pi := (\{u, v\}, \emptyset, \{[u, v]\})$
2: Π_{out} :=EnumFreqTPSub($\mathcal{D}, \sigma, \pi$) (Procedure 3)
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;

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

We can prove the following theorem. Due to the space limit, we omit the proof.

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

Procedure 4 ReplaceEdge

Input: A set $\mathcal{D} \subseteqq \mathcal{OT}$ 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;

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

Procedure 5 ReplaceEdgeSub

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

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

1: **if** $p > |E_\pi \cup H_\pi|$ **then**
2: **return** $\emptyset$
3: **end if**
4: $\Pi_{out} := \emptyset$
5: Let T_D be the wildcard tree pattern in Fig.3.
6: Let h be the p-th variable in the DFS order of all edges and variables of π.
7: $\pi_? := \pi\{[h := [T_D, [R_D, L_D]]\}$
8: **if** $\pi_?$ is σ-frequent w.r.t. $\mathcal{D}$ **then**
9: $\Pi_{out} := \{\pi_?\}$
10: **end if**
11: $\Pi_{tmp} := \Pi_{out} \cup \{\pi\}$
12: **for** each wildcard tree pattern $\pi' \in \Pi_{tmp}$ **do**
13: $\Pi_{out} := \Pi_{out} \cup$ ReplaceEdgeSub$(\mathcal{D}, \sigma, \pi', p+1)$
14: **end for**
15: **return** Π_{out}

Procedure 6 TestMaximality

Input: A set $\mathcal{D} \subseteqq \mathcal{OT}$ 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;

1: $\Pi_{out} := \Pi_{in}$
2: Let T_A, T_B, T_C and T_D be the wildcard tree patterns in Fig.3.
3: **for** each wildcard tree pattern $\pi \in \Pi_{out}$ **do**
4: **for** each variable h in π **do**
5: **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**
6: $\Pi_{out} := \Pi_{out} \setminus \{\pi\}$
7: **end if**
8: **end for**
9: **end for**
10: **return** Π_{out}

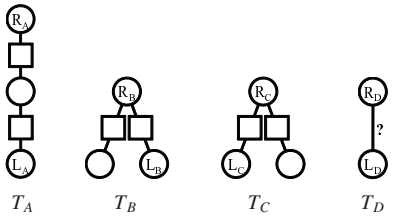

Fig. 3 Wildcard tree pattens T_X $(X \in \{A, B, C, D\})$.

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

Definition 4 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 Λ_{KW} a *keyword*. For a keyword $/k/ \in \Lambda_{KW}$, we define the set $\Lambda_{\{/k/\}} = \{w \in \Lambda \mid k \text{ is a substring of } w\}$. 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.

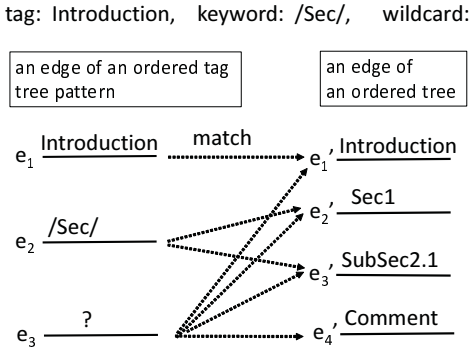

Fig. 4 The matching relation of an edge of a tag tree pattern and an edge of a tree.

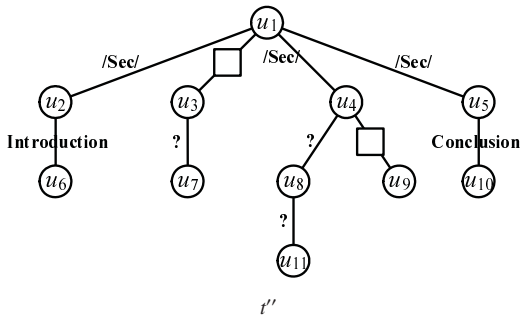

Fig. 5 A maximally σ-frequent tag tree pattern t'' w.r.t. $\mathcal{D} = \{T_1, T_2, T_3\}$ given in Fig.1, where $Tag = \{Introduction, Comment, Conclusion\}$, $KW = \{/Sec/, /SubSec/\}$ and $\sigma = 0.5$.

Two tag tree patterns f and g are *isomorphic* if f and g are isomorphic as wildcard tree patterns (defined in Sec. 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 Sec. 2) for a wildcard tree pattern, where a binding $e := [g, \sigma]$ for an edge e with a keyword $/k/$ can replace the edge e with any word tree $g \in \mathcal{WT}_{\Lambda_{\{/k/\}}}$, 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 $\mathcal{OT} \mid t$ 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. 4. Let "Introduction" be a tag, and "/Sec/" a keyword. We assume that "Intro-

duction",“Sec1”,“SubSec2.1” and “Comment” are all included in $\Lambda_{\{?\}}$. In a tag tree pattern, let 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 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, $\Lambda_{\mathcal{D}}$ the set of all edge labels of trees in $\mathcal{D}$. 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 $L_\Lambda(\pi') \subsetneqq L_\Lambda(\pi)$ then π' is not σ-frequent for any tag tree pattern π' in $\mathcal{OTTP}(Tag, KW)$.

Example 3 Let t'' be a tag tree pattern in $\mathcal{OTTP}(Tag, KW)$, which is described in Fig. 5, where we set $Tag = \{Introduction, Comment, Conclusion\}$ and $KW = \{/Sec/, /SubSec/\}$. The tag tree pattern t'' 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'' is more specific than the wildcard pattern t' in Fig. 1, that is, $L_\Lambda(t'') \subsetneqq L_\Lambda(t')$ holds.

All Maximally Frequent Ordered Tag Tree Patterns (MFOTTP)

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

Assumption: (1) $\Lambda_{\mathcal{D}} \subseteq \Lambda_{\{?\}} \subsetneqq \Lambda$, (2) $\Lambda(Tag, KW) \subsetneqq \Lambda_{\{?\}} \subsetneqq \Lambda$ and (3) $Tag \cap \bigcup_{/k/ \in KW} \Lambda_{\{/k/\}} = \emptyset$

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

We can give an algorithm which generates all maximally σ-frequent tag tree patterns, by extending the algorithm GenMFOWTM in Section 4.

6. Conclusions

We have proposed wildcard tree patterns, which are ordered tree patterns with structured variables and wildcards, and match whole trees. First we have shown that it is hard to compute the optimum frequent wildcard tree patterns, that is, the wildcard tree pattern of maximum-tree size and the wildcard tree pattern of maximum variable-size. Then we have presented an algorithm for enumerating all maximally frequent wildcard tree patterns. Finally, as an application, we have considered an algorithm for enumerating all maximally frequent ordered tag tree patterns.

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., Arimura, H., Sakamoto, H. and Arikawa, S.: Efficient substructure discovery from large semi-structured data, *Proc. 2nd SIAM Int. Conf. Data Mining (SDM-2002)*, pp. 158–174 (2002).

[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. of 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA-2014)*,, pp. 386–390 (2014).

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

[6] 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).

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

[8] 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, Springer-Verlag, LNAI 2336*, pp. 341–355 (2002).

[9] 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, Springer-Verlag, LNAI 3056*, pp. 133–134 (2004).

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

[11] 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. of International Workshop on Challenges in Web Information Retrieval and Integration (WIRI-2005)*, pp. 107–115 (2005).

[12] 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(1), pp. 63–90 (2006).

[13] 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(6), pp. 1197–1211 (2015).

[14] 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).

[15] Zaki, M.: Efficiently mining frequent trees in a forest, *Proc. SIGKDD-2002*, pp. 71–56 (2002).