IPSJ Transactions on Mathematical Modeling and Its Applications Vol.12 No.3 78–88 (Dec. 2019)

Regular Paper

Enumeration of Maximally Frequent Ordered Tree Patterns with Height-constrained Variables for Trees

Yusuke Suzuki[1,a) Tetsuhiro Miyahara[1,b)] Takayoshi Shoudai[2,c)]
Tomoyuki Uchida[1,d)] Satoshi Matsumoto[3,e)] Tetsuji Kuboyama[4,f)]

Received: October 29, 2018, Revised: January 8, 2019/July 11, 2019,
Accepted: August 30, 2019

Abstract: We propose height-constrained ordered tag tree patterns for representing characteristic tree structured features of structured data which are represented by rooted trees with ordered children. Height-constrained ordered tag tree patterns are ordered tree patterns having height-constrained structured variables, wildcards, tags and keywords as edge labels. For two positive integers i and j $(i \leq j)$, an (i, j)-height-constrained structured variable can be replaced with any rooted ordered tree whose trunk length is at least i and whose height is at most j. A height-constrained variable can represent the distance between subtree patterns connected by the variable as one of tree structured features. On the other hand, a variable without height constraint can be replaced with any rooted ordered tree. A variable without height constraint can represent the connectedness but not the distance between subtree patterns. In this sense, ordered tree patterns with height-constrained variables are more accurate tree structured features than ordered tree patterns having variables without height constraint. In this paper, first, we state that it is hard to compute an optimum frequent height-constrained ordered tag tree pattern. Then, we present an algorithm for enumerating all maximally frequent height-constrained ordered tag tree patterns. Finally, we report experimental results showing the effectiveness of the proposed model of characteristic tree structured features, maximally frequent height-constrained ordered tag tree patterns, compared with the previous model of maximally frequent ordered tag tree patterns without height constraint.

Keywords: ordered tree pattern, height-constrained variable, enumeration, tree structured feature, maximal frequency

1. Introduction

The modeling of characteristic tree structured features common to given tree structured data has been more and more important, as the amount of tree structured data has increased. In this paper, we present a new refined model for representing characteristic tree structured features and a discovery method of such characteristic features. We consider models of characteristic tree structured features in two aspects, i.e., expressive power of tree structured patterns and desired characteristics that the tree structured patterns must have.

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]. Many tree structured data such as glycan data with respect to a specific phenomenon and the XML format of DBLP database are known to have height constraint, i.e., the height of such tree structured data is partially bounded by a constant induced from the property of the data.

Therefore, in this paper, in order to formalize tree structured features concerning expressive power of tree structured patterns, we propose *height-constrained ordered tag tree patterns* (or simply *HC-tag tree patterns*), which are ordered tree patterns having height-constrained structured variables, wildcards, tags and keywords as edge labels. A height-constrained structured variable [12] can be replaced with an arbitrary rooted ordered tree satisfying height constraint, but a structured variable satisfying no height constraint [9] can be replaced with an arbitrary rooted ordered tree satisfying no height constraint. An HC-tag tree pattern or an ordered tag tree pattern without height constraint matches the whole structure of a tree, although many other work such as [2] uses subtree structures as tree structured features. For two positive integers i, j $(i \leq j)$, an (i, j)*-height-constrained variable* (or simply an *HC-variable*) can be replaced with any tree such that the length of the path (called the *trunk length* of the tree) corresponding to the variable (See Section 2) is at least i and the height of the tree is at most j. A height-constrained variable can represent the distance between subtree patterns connected by the variable as one of tree structured features. On the other hand, a variable without height constraint can be replaced with any rooted ordered tree. A variable without height constraint can represent

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 Faculty of Science, Tokai University, Hiratsuka, Kanagawa 259–1292, Japan
4 Computer Centre, Gakushuin University, Toshima, Tokyo 171–8588, Japan
a) y-suzuki@hiroshima-cu.ac.jp
b) miyares19@hiroshima-cu.ac.jp
c) shoudai@cb.kiu.ac.jp
d) uchida@hiroshima-cu.ac.jp
e) matsumoto@tsc.u-tokai.ac.jp
f) ori-mps19@tk.cc.gakushuin.ac.jp

IPSJ Transactions on Mathematical Modeling and Its Applications Vol.12 No.3 78–88 (Dec. 2019)

the connectedness but not the distance between subtree patterns. A wildcard for edge labels of an HC-tag tree pattern matches any edge label of a tree. A tag (resp. a keyword) as an edge label of an HC-tag tree pattern matches the same edge label as the tag (resp. an edge label containing the keyword as a substring) of a tree. An HC-tag tree pattern t is said to *match* a tree T if T can be obtained from t by replacing variables of t with appropriate trees satisfying the above height constraint and replacing wildcards and keywords with appropriate matched strings. Thus HC-tag tree patterns are more accurate tree structured features than ordered tag tree patterns without height constraint.

For example in **Fig. 1**, we consider an HC-tag tree pattern t and its corresponding ordered tag tree pattern t' without height constraint. The pattern t' is obtained from t by converting the HC-variables of t to the variables without height constraint. Figure 1 shows that the expressive power, i.e., the set of all matched trees, of t is strictly smaller than that of t'.

The maximal frequency of tree structured patterns is shown to be effective as the desired characteristic that the tree structured patterns must have [9]. In this paper, as a discovery method of a new refined model for representing characteristic tree structured features, we give a method for enumerating all maximally frequent HC-tag tree patterns. An HC-tag tree pattern t is said to be *σ-frequent* w.r.t. a set $\mathcal{D}$ of trees, if the ratio of the trees that are matched by t in $\mathcal{D}$ is larger than or equal to a user-specified threshold ratio σ (a real number σ with $0 < \sigma \leq 1$). An HC-tag tree pattern t is said to be *maximally σ-frequent* w.r.t. a set $\mathcal{D}$ of trees, if t is σ-frequent w.r.t. $\mathcal{D}$ and any HC-tag tree pattern more specific than t w.r.t. the substitution operation is not σ-frequent w.r.t. $\mathcal{D}$.

For example in **Fig. 2**, we consider finding one of the maximally 0.75-frequent HC-tag tree patterns that match at least three trees in $\{T_1, T_2, T_3, T_4\}$. The HC-tag tree pattern t_0 matches all trees in $\{T_1, T_2, T_3, T_4\}$. However t_0 matches all trees whose height is at most 6, so t_0 is an overgeneralized and meaningless pattern. In comparison, the HC-tag tree pattern t_1 is one of the maximally 0.75-frequent HC-tag tree patterns that match three trees T_1, T_2 and T_3 but not T_4. The ordered tag tree pattern t_2, which is an ordered tree pattern without height constraint considered in our previous work [9], is one of the maximally 0.75-frequent ordered tag tree patterns that match three trees T_1, T_2 and T_3 but not T_4. However the HC-tag tree pattern t_1 is far more specific than the ordered tag tree pattern t_2 without height constraint, as Fig. 2 shows that t_2 matches T_5 and t_1 does not match T_5.

In this paper, first we give a hardness result of computing an optimum HC-tag tree pattern. The problem, called **Maximally Frequent Height-Constrained Ordered Tag Tree Pattern of Maximum Tree-Size**, is to find a maximally frequent HC-tag tree pattern with maximum number of vertices. In Theorem 1, we state that this problem is NP-complete. This indicates that it is hard to find an optimum HC-tag tree pattern representing given data. Since meaningless tree patterns are excluded and all possible meaningful tree patterns are not missed, we consider the problem, called **All Maximally Frequent Height-Constrained Ordered Tag Tree Patterns** (**MFHCOTTP**), of enumerating all

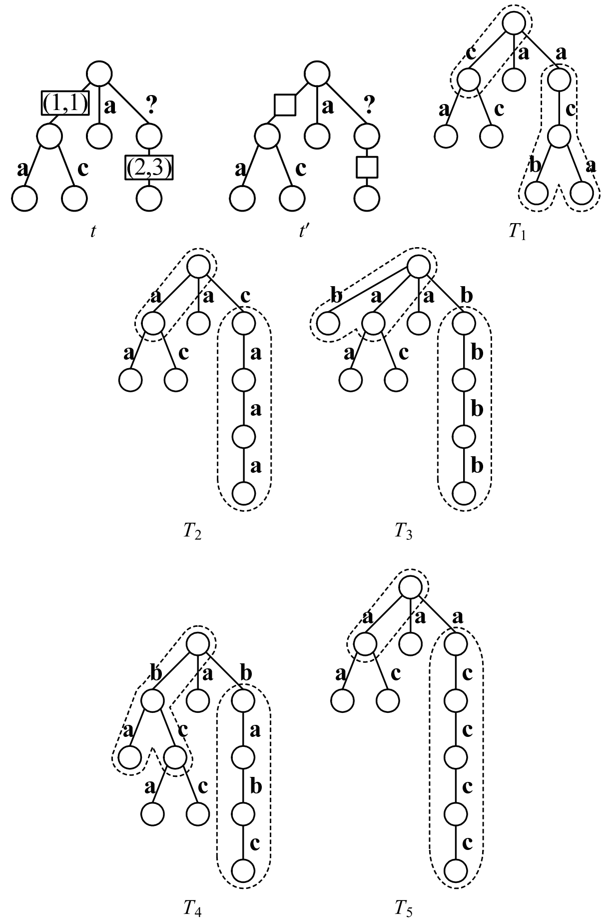


Fig. 1 An HC-tag tree pattern t and its corresponding ordered tag tree pattern t' without height constraint. Trees T_1, T_2, T_3, T_4 and T_5. A variable is represented by a box with lines to its elements. A box with a notation (i, j) inside shows an (i, j)-HC-variable. A box without a notation (i, j) inside shows a variable without height constraint. The label near an edge represents the edge label of the edge. The edge label "?" of t and t' is a wildcard for edge labels. The edge labels "a" and "c" are tags. Variables of tag tree patterns are replaced with subtrees enclosed by broken lines. The pattern t matches T_1, T_2 and T_3 but does not match T_4 and T_5. The pattern t' matches T_1, T_2, T_3, T_4 and T_5.

maximally frequent HC-tag tree patterns. We present an algorithm, called Gen-MFHCOTTP, for solving MFHCOTTP, i.e., an algorithm for enumerating maximally frequent HC-tag tree patterns, and show the correctness and the computational complexity of the algorithm.

Finally, we report comparative experimental results of the proposed algorithm Gen-MFHCOTTP that enumerates all maximally frequent HC-tag tree patterns and the previous algorithm Gen-MFOTTP [9] that enumerates all maximally frequent ordered tag tree patterns without height constraint. The experimental results show that maximally frequent HC-tag tree patterns have more characteristic tree structures than maximally frequent ordered tag tree patterns without height constraint.

We discuss related work. Our HC-tag tree patterns are different from tree structured patterns in related research [2], [3], [4], [5], [15], [16], in that our tree patterns have HC-variables that can be replaced with arbitrary trees having height constraint, and match the whole structure of a tree instead of a subtree structure. In our previous work [8], [10], we considered maximally frequent tree

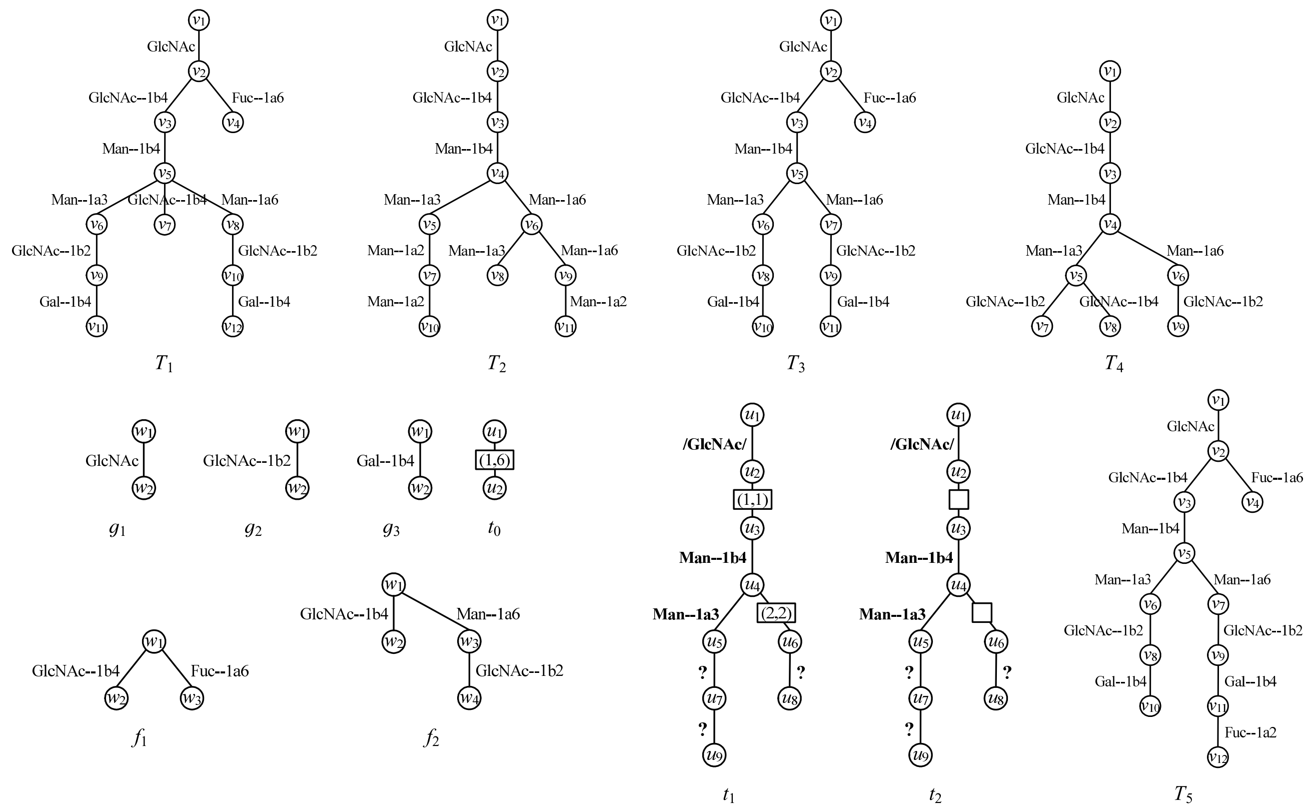


Fig. 2 Trees $T_1, \ldots, T_5, f_1, f_2, g_1, g_2$ and g_3. The HC-tag tree pattern t_0 matches trees T_1, T_2, T_3 and T_4. The HC-tag tree pattern t_1 is one of the maximally 0.75-frequent HC-tag tree patterns that match trees T_1,T_2,T_3 but not T_4. The HC-tag tree pattern t_1 is maximally 0.75-frequent w.r.t. $\mathcal{D} = \{T_1, T_2, T_3, T_4\}$. The ordered tag tree pattern t_2 without height constraint is treated in our previous work [9].

patterns with unordered children and contractible variables, all of which are different from HC-tag tree patterns. In Ref. [12], we considered finding a minimally generalized height-constrained ordered term tree pattern, i.e., a least generalized tree pattern of frequency 1.0, from tree structured data with at least two edge labels. In Ref. [12], also we gave an efficient pattern matching algorithm for height-constrained ordered term tree patterns, the extended algorithm of which we use in this paper for calculating the matching relation of HC-tag tree patterns and trees.

This paper is organized as follows. In Section 2, we introduce HC-tag tree patterns and give a hardness result of computing an optimum HC-tag tree pattern. In Section 3, we give an algorithm for enumerating all maximally frequent HC-tag tree patterns and show its correctness and computational complexity. In Section 4, we report experimental results showing the effectiveness of the proposed model of characteristic tree structured features, i.e., maximally frequent HC-tag tree patterns, compared with the previous model [9]. In Section 5, we conclude this paper. This paper is a complete version of our previous results on HC-tag tree patterns [13], and we present the full descriptions of an improved algorithm, full proofs and comparative experimental results showing the effectiveness of maximally frequent HC-tag tree patterns.

2. Height-Constrained Ordered Tag Tree Pattern

We explain height-constrained ordered tag tree patterns as tree structured patterns. Let Λ be a set of infinitely or finitely many words. In this paper, a *tree* means a rooted tree with ordered children such that each edge is labeled with an element in Λ. Let X be an infinite alphabet. We assume that $\Lambda \cap X = \emptyset$. For a set S, the number of elements in S is denoted by $|S|$.

Definition 1 (Wildcard, keyword and tag) Let "?" be a special symbol, called a *wildcard*, such that "?" $\notin \Lambda$ holds. Let $\Lambda_{\{?\}}$ be a subset of Λ. The symbol "?" is a wildcard for any word in $\Lambda_{\{?\}}$. Let Λ_{Tag} be a subset of Λ. Let Λ_{KW} be a set 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 that has a set V_T of vertices and a set E_T of edges. For a tree T, $V(T)$ and $E(T)$ denote the vertex set and the edge set of T, respectively. For a tree T and its vertices v and w, a *path* from v to w is a sequence $v = v_1, v_2, \ldots, v_n = w$ of distinct vertices of T such that for any k with $1 \leq k < n$, there exists an edge consisting of v_k and v_{k+1}. The integer $n-1$ is called the *length* of the path $v_1, v_2, \ldots, v_n$. If there is an edge consisting of u and u' such that u lies on the path from the root to u', then u is said to be the *parent* of u' and u' is a *child* of u. A notation (u, u') denotes the edge such that u is the parent of u'.

Definition 2 (Height-constrained variable label) Let $X^{\mathcal{H}}$ be an infinite subset of X. An element of $X^{\mathcal{H}}$ is called a *height-constrained variable label* (or simply an *HC-variable label*). For two positive integers i, j ($i \leq j$), let $X^{\mathcal{H}(i,j)}$ be an infinite subset of $X^{\mathcal{H}}$ such that $X^{\mathcal{H}} = \bigcup_{1\le i\le j} X^{\mathcal{H}(i,j)}$ and for $(i, j) \ne (i', j')$ $X^{\mathcal{H}(i,j)} \cap X^{\mathcal{H}(i',j')} = \emptyset$ hold.

Definition 3 (Height-constrained ordered tag tree pattern) Let $T = (V_T, E_T)$ be a tree which has a set V_T of vertices and a set E_T of edges with an edge labeling function $\mu_T : E_T \to \{\text{"?"}\} \cup \Lambda_{Tag} \cup \Lambda_{KW} \cup X^{\mathcal{H}}$. Let $E_t = \{e \in E_T \mid \mu_T(e) \in \{\text{"?"}\} \cup \Lambda_{Tag} \cup \Lambda_{KW}\}$ and $H_t = \{h \in E_T \mid \mu_T(h) \in X^{\mathcal{H}}\}$ be a partition of E_T, i.e., $E_t \cup H_t = E_T$ and $E_t \cap H_t = \emptyset$. And let $V_t = V_T$. A triplet $t = (V_t, E_t, H_t)$ is called a *height-constrained ordered tag tree pattern* (or simply an *HC-tag tree pattern*). The root of t is the root of T. The definitions of a path, a parent and a child of t are defined as those definitions of T. Hereafter, we call an element in E_t an edge and call an element in H_t a *height-constrained variable* (or simply an *HC-variable*).

For an HC-tag tree pattern t, $V(t)$, $E(t)$, and $H(t)$ denote the vertex set, the edge set, and the HC-variable set of t, respectively. For two positive integers i, j $(i \le j)$, an HC-variable h is called an (i, j)*-height-constrained variable* (or simply an (i, j)*-HC-variable*), if h is an HC-variable labeled with an element in $X^{\mathcal{H}(i,j)}$. The notation $h^{(i,j)}$ means that h is an (i, j)*-HC-variable*. We use a notation $[u, u']$ to represent an HC-variable such that u is the parent of u'. Let $H(v_1, v_n)$ be the set of all HC-variables in the path $v_1, v_2, \ldots, v_n$ of t and $E(v_1, v_n)$ the set of all edges in the path $v_1, v_2, \ldots, v_n$ of t. The *trunk length* of the path $v_1, v_2, \ldots, v_n$ is defined as the integer $\sum_{h^{(i,j)} \in H(v_1,v_n)} i + |E(v_1, v_n)|$. The *height* of t, denoted by *height*(t), is defined as the integer $\max\left\{ \sum_{h^{(i,j)} \in H(r,\ell)} j + |E(r, \ell)| \;\middle|\; r \text{ is the root and } \ell \text{ is a leaf} \right\}$. The *minimum size* of t, denoted by $Size_{\min}(t)$, is defined as the integer $\sum_{h^{(i,j)} \in H(t)} i + |E(t)|$. An HC-tag tree pattern t has a total ordering on all children of every internal vertex u. The ordering on the children of u is denoted by $<^t_u$. That is, for any two children u' and u'' of u, $u' <^t_u u''$ denotes that u' is a left sibling of u'' in t.

Definition 4 (Variable-chain) Let t be an HC-tag tree pattern. Let a sequence $u_0, u_1, \ldots, u_k$ be a path of t such that its length is more than one and for every $u_{\ell-1}$ and u_ℓ $(1 \le \ell \le k)$, $[u_{\ell-1}, u_\ell]^{(i_\ell, j_\ell)}$ is an (i_ℓ, j_ℓ)-HC-variable of t for certain integers i_ℓ and j_ℓ. Then, $u_0, u_1, \ldots, u_k$ is said to be a *variable-chain* of t if u_ℓ is the only child of $u_{\ell-1}$ for any $2 \le \ell \le k$. If g has no variable-chain, t is called *variable-chain free*.

Definition 5 (Word tree [9]) $\mathcal{OT}$ denotes the set of all trees whose edge labels are in Λ. 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 $\mathcal{WT}_{\Lambda'} = \bigcup_{w\in\Lambda'} \{T(w)\}$.

Definition 6 (Class of HC-tag tree patterns) $\mathcal{OTTP}^H$ denotes the set of all HC-tag tree patterns. $\mathcal{OTTP}^h$ denotes the set of all variable-chain free HC-tag tree patterns. For two positive integers p and q $(p \le q)$, $\mathcal{OTTP}^{h(p,q)}$ denotes the set of all variable-chain free HC-tag tree patterns whose HC-variable labels are in $\bigcup_{p\le i\le j\le q} X^{\mathcal{H}(i,j)}$. Note that for any two positive integers p and q, the relations $\mathcal{OTTP}^{h(p,q)} \subsetneq \mathcal{OTTP}^h \subsetneq \mathcal{OTTP}^H$ hold. Let Tag be a finite subset of Λ_{Tag} and KW a finite subset of Λ_{KW}. For a set C of HC-tag tree patterns, we denote by $C(Tag, KW)$ the set of HC-tag tree patterns $t \in C$ with the tags of t in Tag and the keywords of t in KW. In particular, if $Tag = \emptyset$ and $KW = \emptyset$, an HC-tag tree pattern in $C(Tag, KW)$ is called an *HC-wildcard tree pattern*. An HC-wildcard tree pattern is an extended model of a wildcard tree pattern [9] by introducing HC-variables. $\mathcal{OTTP}^h(\Lambda_{Tag}, \Lambda_{KW})$ denotes the set of all variable-chain free HC-tag tree patterns with tags in Λ_{Tag} and keywords in Λ_{KW}.

Let s be an HC-tag tree pattern or a tree with at least two vertices. Let $\tau = [\![w_0, w_1]\!]$ denote a list of two distinct vertices in s where w_0 is the root of s and w_1 is a leaf of s. The trunk length of $\tau = [\![w_0, w_1]\!]$ is regarded as the trunk length of the path $w_0, \ldots, w_1$. Let t be an HC-tag tree pattern with at least two vertices and e an HC-variable or an edge of t. The form $e := \langle s, \tau \rangle$ is called a *binding* for e if the following three conditions hold. (1) If e is an edge labeled with "?", then $s \in \mathcal{WT}_{\Lambda_{\{?\}}}$, (2) if e is an edge labeled with a keyword $/k/$ then $s \in \mathcal{WT}_{\Lambda_{\{/k/\}}}$ and (3) if e is an (i, j)-HC-variable $(1 \le i \le j)$ then (i) the trunk length of τ is at least i and (ii) the height of s is at most j. A new HC-tag tree pattern or a new tree t' is obtained by applying the binding $e := \langle s, \tau \rangle$ to t in the following way. Let $e = [v_0, v_1]$ (resp. $e = (v_0, v_1)$) be an HC-variable (resp. an edge) in t. Let s' be one copy of s and w'_0, w'_1 the vertices of s' corresponding to w_0, w_1 of s, respectively. For the HC-variable or the edge e, we attach s' to t by removing e from $E(t) \cup H(t)$ and by identifying the vertices v_0, v_1 with the vertices w'_0, w'_1 of s', respectively. Further we define a new total ordering $<^{t'}_u$ on every vertex u of t' in a natural way. Suppose that u has more than one child and let u' and u'' be two children of u of t'. We have the following three cases. *Case 1*: If $u, u', u'' \in V(t)$ and $u' <^t_u u''$, then $u' <^{t'}_u u''$. *Case 2*: If $u, u', u'' \in V(s)$ and $u' <^s_u u''$, then $u' <^{t'}_u u''$. *Case 3*: If $u = v_0$, $u' \in V(s)$, $u'' \in V(t)$, and $v_1 <^t_u u''$ (resp. $u'' <^t_u v_1$), then $u' <^{t'}_u u''$ (resp. $u'' <^{t'}_u u'$). Let $e_1, e_2, \ldots, e_n$ be mutually distinct HC-variables or edges in t. A *substitution* θ for t is a finite collection of bindings $\{e_1 := \langle s_1, \tau_1 \rangle, \ldots, e_n := \langle s_n, \tau_n \rangle\}$ if either following conditions are satisfied (1) all s_i $(1 \le i \le n)$ are HC-tag tree pattens or (2) all s_i $(1 \le i \le n)$ are trees and $\{e_1, \ldots, e_n\} = \{e \in E(t) \mid e$ is labeled with the wildcard or a keyword $\} \cup H(t)$. The new HC-tag tree pattern or the new tree $t\theta$, called the *instance* of t by θ, is obtained by applying the all bindings $e_i := \langle s_i, \tau_i \rangle$ to t simultaneously. We note that the root of $t\theta$ is the root of t.

Example 1 **Figure 3** shows examples of applying bindings. Let $e = [u_2, u_3]$ be the $(1, 6)$-HC variable in t_0. The HC-tag tree pattern t_1 is obtained from t_0 by applying the binding $e := \langle T_1, [\![w_1, w_3]\!] \rangle$ for e to t_0. The trunk length of $\tau_1 = [\![w_1, w_3]\!]$ is 2 and the height of T_1 is 2. The HC-tag tree pattern t_2 is obtained from t_0 by applying the binding $e := \langle T_2, [\![w_1, w_3]\!] \rangle$ for e to t_0. The trunk length of $\tau_2 = [\![w_1, w_3]\!]$ is 1 and the height of T_2 is 3.

Let t and s be two HC-tag tree patterns. We say that t and s are *isomorphic*, denoted by $t \cong s$, if there is a bijection φ from $V(t)$ to $V(s)$ such that (1) the root of t is mapped to the root of s by φ, (2) $(u, v) \in E(t)$ if and only if $(\varphi(u), \varphi(v)) \in E(s)$ and the two edges have the same edge label, (3) for any integers i, j $(1 \le i \le j)$, $[u, v]^{(i,j)} \in H(t)$ if and only if $[\varphi(u), \varphi(v)]^{(i,j)} \in H(s)$, and (4) for any internal vertex u in t which has more than one child, and for any two children u' and u'' of u, $u' <^t_u u''$ if and

IPSJ Transactions on Mathematical Modeling and Its Applications Vol.12 No.3 78–88 (Dec. 2019)

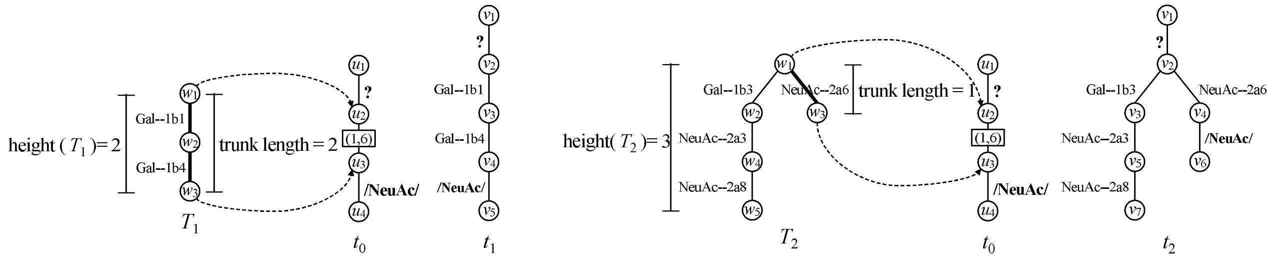


Fig. 3 Examples of applying bindings. HC-tag tree patterns t_0, t_1, t_2 and trees T_1, T_2.

only if $\varphi(u') <^s_{\varphi(u)} \varphi(u'')$. Such a bijection from $V(t)$ to $V(s)$ is called an *isomorphism* from t to s.

An HC-tag tree pattern t is said to *match* a tree T if there exists a substitution θ such that $T \cong t\theta$ holds.

Definition 7 (Language of HC-tag tree patterns) For an HC-tag tree pattern t in $\mathcal{OTTP}^H$, the *language* $L(t)$ is defined as $\{s \in \mathcal{OT} \mid s \cong t\theta$ for a substitution $\theta\}$.

Let $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$ be a nonempty finite set of trees. The *matching count* of an HC-tag tree pattern t w.r.t. $\mathcal{D}$, denoted by $match_{\mathcal{D}}(t)$, is the number of trees $T_i \in \mathcal{D}$ $(1 \le i \le m)$ such that t matches T_i. Then the *frequency* of t w.r.t. $\mathcal{D}$ is defined by $supp_{\mathcal{D}}(t) = match_{\mathcal{D}}(t)/m$. Let σ be a real number where $0 < \sigma \le 1$. An HC-tag tree pattern t is σ-*frequent* w.r.t. $\mathcal{D}$ if $supp_{\mathcal{D}}(t) \ge \sigma$ holds. Let Tag be a finite subset of Λ_{Tag} and KW a finite subset of Λ_{KW}.

Definition 8 (Maximally frequent HC-tag tree patterns) An HC-tag tree pattern t in $\mathcal{OTTP}^h(Tag, KW)$ is *maximally* σ-*frequent* w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^h(Tag, KW)$ if the following two conditions hold. (1) t is σ-frequent w.r.t. $\mathcal{D}$. (2) For any HC-tag tree pattern s in $\mathcal{OTTP}^h(Tag, KW)$ if $s \not\cong t$ and there is a substitution θ such that $s \cong t\theta$ holds, then s is not σ-frequent w.r.t. $\mathcal{D}$.

Example 2 Let $\mathcal{D} = \{T_1, T_2, T_3, T_4\}$ be the set of trees in Fig. 2. We set Tag = {"Gal–1b4","Man–1a3","Man–1a6","Man–1b4" } and KW = {"/GlcNAc/","/NeuAc/"}. The HC-tag tree pattern t_1 in Fig. 2 is a maximally σ-frequent w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^h(Tag, KW)$.

We give the hardness of computing an optimum HC-tag tree pattern. The formal definition of the problem is as follows.

Maximally Frequent Height-Constrained Ordered Tag Tree Pattern of Maximum Tree-Size

Instance: A nonempty finite set of trees $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$, a real number σ $(0 < \sigma \le 1)$, a finite set Tag of tags, a finite set KW of keywords and a positive integer K.

Question: Is there a maximally σ-frequent HC-tag tree pattern t w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^h(Tag, KW)$ with $|V(t)| \ge K$?

In Ref. [9], we showed that the problem of computing a maximally frequent tag tree pattern of maximum tree-size w.r.t. a nonempty finite set of trees is NP-complete. We can prove the following theorem in a similar way to the proof of Theorem 1 in Ref. [9].

Theorem 1 Maximally Frequent Height-Constrained Ordered Tag Tree Pattern of Maximum Tree-Size is NP-complete.

An ordered tag tree pattern (or simply called a tag tree pattern) is an ordered tree structured pattern having variables without height constraint [9]. We note that the set $\mathcal{OTTP}^h(\Lambda_{Tag}, \Lambda_{KW})$ of all variable-chain free HC-tag tree patterns is incomparable with the set $\mathcal{OTTP}_{(\Lambda_{Tag},\Lambda_{KW})}$ of all tag tree patterns [9]. Moreover, the class of all languages of HC-tag tree patterns in $\mathcal{OTTP}^h(\Lambda_{Tag}, \Lambda_{KW})$ is incomparable with the class of all languages of tag tree patterns in $\mathcal{OTTP}_{(\Lambda_{Tag},\Lambda_{KW})}$. From Theorem 1, we propose a new enumeration algorithm which outputs all maximally frequent HC-tag tree patterns.

3. Enumeration of Maximally Frequent Tag Tree Patterns with Height-constrained Variables

3.1 Enumeration Algorithm

In this section, we consider the following problem.

All Maximally Frequent Height-Constrained Ordered Tag Tree Patterns (MFHCOTTP)

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

Assumption: (1) $(Tag \cup \bigcup_{/k/\in KW} \Lambda_{\{/k/\}}) \subsetneqq \Lambda_{\{?\}} \subsetneqq \Lambda$, (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 HC-tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^h(Tag, KW)$.

Let $\mathcal{D} \subsetneqq \mathcal{OT}$ be a nonempty finite set of trees. In our previous work [9], we proposed the algorithm Gen-MFOTTP that enumerates all maximally σ-frequent tag tree patterns w.r.t. $\mathcal{D}$. By extending the previous algorithm Gen-MFOTTP, we propose an algorithm Gen-MFHCOTTP (Algorithm 1) that generates all maximally σ-frequent HC-tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^h(Tag, KW)$. In the algorithm Gen-MFHCOTTP, we decide whether or not an HC-tag tree pattern is σ-frequent w.r.t. $\mathcal{D}$, by using an extended version of polynomial time pattern matching algorithm [12] in a similar way to Ref. [9].

At first, Procedure EnumFreqTP is Procedure 2 of Gen-MFOTTP [9] that uses the tree enumeration technique [2], [11], [17]. Procedure ReplaceEdge2, which is Procedure 8 of Gen-MFOTTP, outputs the set $\Pi_2(\sigma)$ of all σ-frequent tag tree patterns w.r.t. $\mathcal{D}$.

Then, in Algorithm Gen-MFHCOTTP, we propose new procedures MergeVariable (Procedure 2), ConstrainVariable (Procedure 3) and ConstrainVariableSub (Procedure 4) to enumerate σ-frequent HC-tag tree patterns from $\Pi_2(\sigma)$ as follows. Let h_D be the maximum height of trees in $\mathcal{D}$. For a σ-frequent tag tree pattern $t \in \Pi_2(\sigma)$, Procedure MergeVariable makes the σ-frequent variable-chain free HC-tag tree pattern t^h from t by re-

Algorithm 1 GEN-MFHCOTTP

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

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

/* Step 1,2: Enumerate all σ-frequent tag tree patterns */
$\Pi_1(\sigma) :=$ ENUMFREQTP$(\mathcal{D}, \sigma)$ // See Procedure 2 in [9]
$\Pi_2(\sigma) :=$ REPLACEEDGE2$(\mathcal{D}, \sigma, Tag, KW, \Pi_1(\sigma))$ // See Procedure 8 in [9]
/* $\Pi_2(\sigma)$ is the set of all σ-frequent tag tree patterns without height constraint */
/* Step 3,4: Enumerate all σ-frequent HC-tag tree patterns */
$\Pi_3(\sigma) :=$ MERGEVARIABLE$(\mathcal{D}, \sigma, \Pi_2(\sigma))$ // See Procedure 2
$\Pi_4(\sigma) :=$ CONSTRAINVARIABLE$(\mathcal{D}, \sigma, \Pi_3(\sigma))$ // See Procedure 3
/* $\Pi_4(\sigma)$ is the set of all σ-frequent HC-tag tree patterns */
/* Step 5: Maximality test */
$\Pi(\sigma) :=$ TESTMAXIMALITY$(\mathcal{D}, Tag, KW, \sigma, \Pi_4(\sigma))$ // See Procedure 5
return $\Pi(\sigma)$

Procedure 2 MERGEVARIABLE

Input: A nonempty finite set $\mathcal{D} \subseteq \mathcal{OT}$ 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 HC-tag tree patterns;

$\Pi_{out} := \emptyset$
Let h_D be the maximum height of trees in $\mathcal{D}$
for each tag tree pattern $t \in \Pi_{in}$ **do**
  Let t^h be an HC-tag tree pattern obtained from t by replacing each variable of t with an $(1, h_D)$-HC-variable
  while t^h has a variable-chain **do**
    Let u_1, u_2, u_3 be vertices in t^h such that $[u_1, u_2]^{(i_1, h_D)}$ is an (i_1, h_D)-HC-variable, $[u_2, u_3]^{(i_2, h_D)}$ is an (i_2, h_D)-HC-variable and u_3 is the only child of u_2
    /* the path u_1, u_2, u_3 is a variable-chain */
    $t^h := (V(t^h) \setminus \{u_2\}, E(t), H(t^h) \cup \{[u_1, u_3]^{(i_1+i_2, h_D)}\} \setminus \{[u_1, u_2]^{(i_1, h_D)}, [u_2, u_3]^{(i_2, h_D)}\})$
  end while
  $\Pi_{out} := \Pi_{out} \cup \{t^h\}$
end for
return Π_{out}

Procedure 3 CONSTRAINVARIABLE

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

Output: A set Π_{out} of HC-tag tree patterns;

$\Pi_{out} := \Pi_{in}$
for each HC-tag tree pattern $t \in \Pi_{in}$ **do**
  $\Pi_{out} := \Pi_{out} \cup$ CONSTRAINVARIABLESUB$(\mathcal{D}, \sigma, t, 1)$ // See Procedure 4
end for
return Π_{out}

placing each variable of t with a $(1, h_D)$-HC-variable and merging consecutive HC-variables into one HC-variable. Procedure CONSTRAINVARIABLE outputs the set $\Pi_4(\sigma)$ of all σ-frequent HC-tag tree patterns in $\mathcal{OTTP}^{h(1,h_D)}(Tag, KW)$ by replacing each (i, j)-HC-variable with an (i, j')-HC-variable ($i \leq j' < j$).

Finally, Procedure TESTMAXIMALITY (Procedure 5), which is the extended version of Procedure TESTMAXIMALITY2 in Ref. [9], decides whether or not each HC-tag tree pattern in $\Pi_4(\sigma)$ is maximally σ-frequent w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^h(Tag, KW)$.

Procedure 4 CONSTRAINVARIABLESUB

Input: A nonempty finite set $\mathcal{D} \subseteq \mathcal{OT}$ of trees, a real number σ ($0 < \sigma \leq 1$), an HC-tag tree pattern t and a positive integer p;

Output: A set Π_{out} of HC-tag tree patterns;

if $p > |H(t)|$ **then**
  return $\emptyset$
end if
$\Pi_{out} := \emptyset$
Let $h^{(i,j)} = [u, v]^{(i,j)}$ be the p-th HC-variable of t in the DFS order
for $k := j - 1$ **downto** i **do**
  Let t' be an HC-tag tree pattern obtained from t by replacing an (i, j)-HC-variable $[u, v]^{(i,j)}$ of t with an (i, k)-HC-variable $[u, v]^{(i,k)}$
  if t' is σ-frequent w.r.t. $\mathcal{D}$ **then**
    $\Pi_{out} := \Pi_{out} \cup \{t'\}$
  end if
end for
$\Pi_{tmp} := \Pi_{out} \cup \{t\}$
for each HC-tag tree pattern $t' \in \Pi_{tmp}$ **do**
  $\Pi_{out} := \Pi_{out} \cup$ CONSTRAINVARIABLESUB$(\mathcal{D}, \sigma, t', p + 1)$
end for
return Π_{out}

Procedure 5 TESTMAXIMALITY

Input: A nonempty finite set $\mathcal{D} \subseteq \mathcal{OT}$ of trees, a real number σ ($0 < \sigma \leq 1$), a finite set Tag of tags, a finite set KW of keywords, and a set Π_{in} of HC-tag tree patterns;

Output: A set Π_{out} of HC-tag tree patterns;

$\Pi_{out} := \Pi_{in}$
for each HC-tag tree pattern $t \in \Pi_{out}$ **do**
  for each $(1, j)$-HC-variable $h^{(1,j)}$ in t **do**
    Let T_0("?") be the HC-tag tree pattern in **Fig. 4**
    if $t\{h^{(1,j)} := \langle T_0(\text{"?"}), [\![R_0, L_0]\!]\rangle\}$ is σ-frequent w.r.t. $\mathcal{D}$ **then**
      $\Pi_{out} := \Pi_{out} \setminus \{t\}$
    end if
  end for
  for each (i, j)-HC-variable $h^{(i,j)}$ in t **do**
    Let $T_1^{(i,j)}, \ldots, T_9^{(i,j)}$ be the HC-tag tree patterns in Fig. 4
    if there exists a positive integer $K \in \{1, \ldots, 9\}$ such that $t\{h^{(i,j)} := \langle T_K^{(i,j)}, [\![R_K, L_K]\!]\rangle\}$ is σ-frequent w.r.t. $\mathcal{D}$ **then**
      $\Pi_{out} := \Pi_{out} \setminus \{t\}$
    end if
  end for
  /* $T_0(w)$ is the HC-tag tree pattern in Fig. 4 for a tag or keyword w */
  for each edge e labeled with "?" in t **do**
    if there exists a tag or keyword $w \in Tag \cup KW$ such that $t\{e := \langle T_0(w), [\![R_0, L_0]\!]\rangle\}$ is σ-frequent w.r.t. $\mathcal{D}$ **then**
      $\Pi_{out} := \Pi_{out} \setminus \{t\}$
    end if
  end for
  for each edge e labeled with a keyword in t **do**
    Let $/k/ \in KW$ be the keyword of the edge e
    if there exists a keyword $/k'/ \in KW$ such that $\Lambda_{\{/k'/\}} \subsetneq \Lambda_{\{/k/\}}$ and $t\{e := \langle T_0(/k'/), [\![R_0, L_0]\!]\rangle\}$ is σ-frequent w.r.t. $\mathcal{D}$ **then**
      $\Pi_{out} := \Pi_{out} \setminus \{t\}$
    end if
  end for
end for
return Π_{out}

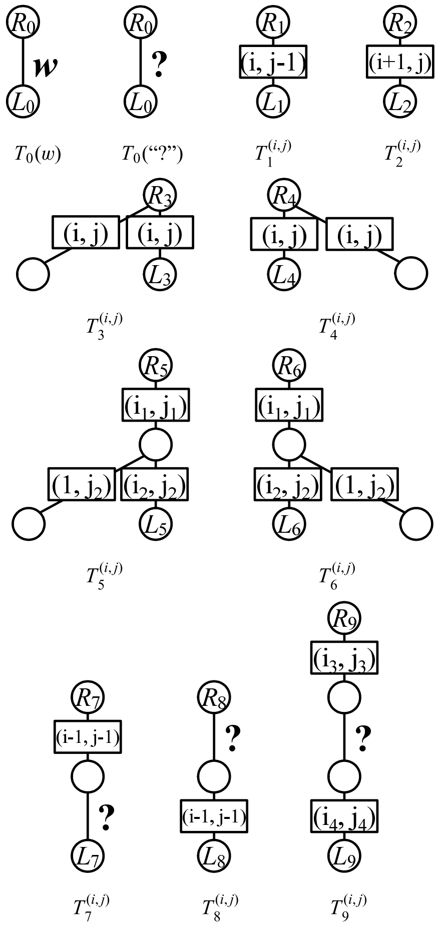


Fig. 4 HC-tag tree pattern $T_0(w)$ for any tag or keyword w. HC-tag tree patterns T_0("?") and $T_1^{(i,j)}, \ldots, T_9^{(i,j)}$. For HC-tag tree patterns $T_5^{(i,j)}$ and $T_6^{(i,j)}$, we assume that $i_1 + i_2 = i$ and $j_1 + j_2 = j$ hold. For the HC-tag tree pattern $T_9^{(i,j)}$, we assume that $i_3 + i_4 + 1 = i$ and $j_3 + j_4 + 1 = j$ hold.

3.2 Correctness of Enumeration Algorithm

In this section, we will prove the correctness of Algorithm Gen-MFHCOTTP in a similar way to the proofs of Lemmas 3, 4 and Theorem 4 in Ref. [9].

First, we give the following important fact of the languages of HC-tag tree patterns. For HC-tag tree patterns t and t', if there is a substitution θ such that $t' \cong t\theta$ holds then $L(t') \subseteq L(t)$ holds. However, for HC-tag tree patterns t and t', $L(t') \subseteq L(t)$ does not necessarily imply that there is a substitution θ such that $t' \cong t\theta$ holds. We give an example of this case in **Fig. 5**. Therefore, in Definition 8, the maximally σ-frequent HC-tag tree patterns is defined by the substitution operation of HC-tag tree patterns. Since the definition of a maximally σ-frequent HC-tag tree pattern is different from that of a maximally σ-frequent tag tree pattern [9], most of the proofs are different.

The following Lemma was proved in Ref. [9].

Lemma 1 (Lemma 4 in Ref. [9]) After the second step of Algorithm Gen-MFOTTP, the set $\Pi_2(\sigma)$ is the set of all σ-frequent tag tree patterns w.r.t. $\mathcal{D}$.

Therefore, after the second step of Algorithm Gen-MFHCOTTP, the set $\Pi_2(\sigma)$ is the set of all σ-frequent tag tree patterns w.r.t. $\mathcal{D}$.

Lemma 2 After the fourth step of Algorithm Gen-MFHCOTTP, the set $\Pi_4(\sigma)$ is the set of all σ-frequent HC-tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^{h(1,h_D)}(Tag, KW)$.

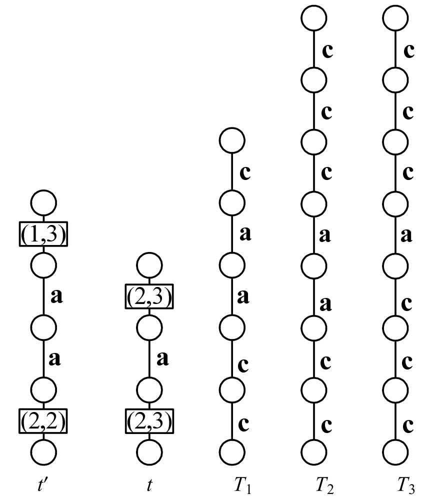


Fig. 5 HC-tag tree patterns t', t and trees T_1, T_2, T_3. We can show that $T_3 \in L(t)$ and $T_3 \notin L(t')$ hold. Thus, we can see that $\{T_1, T_2\} \subseteq L(t') \subsetneq L(t)$ holds, but there is no substitution θ such that $t' \cong t\theta$ holds.

Proof. Procedure MergeVariable makes a variable-chain free HC-tag tree pattern from each tag tree pattern in $\Pi_2(\sigma)$. We note that $|\Pi_2(\sigma)| = |\Pi_3(\sigma)|$ holds. Furthermore, Procedure ConstrainVariable also uses a brute-force method for replacing each (i, j)-HC-variable of t in $\Pi_3(\sigma)$ with an $(i, j-1)$-HC-variable $(1 \le i < j)$. Thus, $\Pi_4(\sigma)$ contains all σ-frequent HC-tag tree patterns in $\mathcal{OTTP}^{h(1,h_D)}(Tag, KW)$. □

Theorem 2 Algorithm Gen-MFHCOTTP outputs the set of all maximally σ-frequent HC-tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^{h}(Tag, KW)$.

Proof. The fifth step of Algorithm Gen-MFHCOTTP (Procedure TestMaximality) removes elements from $\Pi_4(\sigma)$. Therefore, from Lemma 2, all HC-tag tree patterns in $\Pi(\sigma)$ are σ-frequent. Let t be a σ-frequent HC-tag tree pattern in $\Pi(\sigma)$. We will prove that t is a maximally σ-frequent HC-tag tree pattern w.r.t. $\mathcal{D}$. That is, we will prove that for any substitution θ, if $t\theta$ is σ-frequent, then $t\theta \cong t$ holds. We consider each binding $h := \langle s, \tau \rangle$ in θ. According to the definition of bindings, we have the following two cases.

Case 1: h is an (i, j)-HC-variable. Note that $height(s) \le j$ and $Size_{\min}(s) \ge i$ hold. We show that $height(s) = j$, $Size_{\min}(s) = i$ and $|E(s)| = 0$ hold as follows.

- Suppose that $height(s) < j$ holds. Since there is a substitution θ' such that $t\{h := \langle s, \tau \rangle\} \cong t\{h := \langle T_1^{(i,j)}, [\![R_1, L_1]\!] \rangle\}\theta'$ holds, $t\{h := \langle T_1^{(i,j)}, [\![R_1, L_1]\!] \rangle\}$ is σ-frequent w.r.t. $\mathcal{D}$. This contradicts the fact that t is not removed from $\Pi(\sigma)$ in lines 9–14 in Procedure TestMaximality. Therefore, $height(s) = j$ holds.
- Suppose that $Size_{\min}(s) > i$ holds. Since there is a substitution θ' such that $t\{h := \langle s, \tau \rangle\} \cong t\{h := \langle T_K^{(i,j)}, [\![R_K, L_K]\!] \rangle\}\theta'$ holds for some $K \in \{2, 3, 4, 5, 6\}$, $t\{h := \langle T_K^{(i,j)}, [\![R_K, L_K]\!] \rangle\}$ is σ-frequent w.r.t. $\mathcal{D}$. This contradicts the fact that t is not removed from $\Pi(\sigma)$ in lines 9–14 in Procedure TestMaximality. Therefore, $Size_{\min}(s) = i$ holds.
- Suppose that $|E(s)| \ne 0$ holds. Since there is a substitution

θ' such that $t\{h := \langle s, \tau\rangle\} \cong t\{h := \langle T_0(\text{“?''}), [\![R_0, L_0]\!]\rangle\}\theta'$ or $t\{h := \langle s, \tau\rangle\} \cong t\{h := \langle T_K^{(i,j)}, [\![R_K, L_K]\!]\rangle\}\theta'$ hold for some $K \in \{7, 8, 9\}$, $t\{h := \langle T_0(\text{“?''}), [\![R_0, L_0]\!]\rangle\}$ or $t\{h := \langle T_K^{(i,j)}, [\![R_K, L_K]\!]\rangle\}$ is σ-frequent w.r.t. $\mathcal{D}$. This contradicts the fact that t is not removed from $\Pi(\sigma)$ in lines 3–14 in Procedure TestMaximality. Therefore, $|E(s)| = 0$ holds.

Thus, if h is an (i, j)-HC-variable then $height(s) = j$, $Size_{\min}(s) = i$ and $|E(s)| = 0$ hold. Since h is an (i, j)-HC-variable and $Size_{\min}(s) = i$, s consists of either one HC-variable or one variable-chain. However, since s is a variable-chain free HC-tag tree pattern, s is an HC-tag tree pattern consisting of only one (i, j)-HC-variable.

Case 2: h is an edge. Then s is a word tree. Let e be the unique edge of s.

- Suppose that h is labeled with the wildcard “?” and e is labeled with $w \in Tag \cup KW$. Since there is a substitution θ' such that $t\{h := \langle s, \tau\rangle\} \cong t\{h := \langle T_0(w), [\![R_0, L_0]\!]\rangle\}\theta'$ holds, there is a keyword or tag $w \in Tag \cup KW$ such that $t\{h := \langle T_0(w), [\![R_0, L_0]\!]\rangle\}$ is σ-frequent w.r.t. $\mathcal{D}$. This contradicts the fact that t is not removed from $\Pi(\sigma)$ in lines 15–19 in Procedure TestMaximality. Therefore, if h is an edge labeled with the wildcard then e is an edge labeled with the wildcard.
- Suppose that h is labeled with some $/k/ \in KW$ and e is labeled with a keyword $w \in KW$ such that $\Lambda_{\{w\}} \subsetneq \Lambda_{\{/k/\}}$ holds. Since there is a substitution θ' such that $t\{h := \langle s, \tau\rangle\} \cong t\{h := \langle T_0(w), [\![R_0, L_0]\!]\rangle\}\theta'$ holds. there is a keyword $/k'/ \in KW$ such that $t\{h := \langle T_0(w), [\![R_0, L_0]\!]\rangle\}$ is σ-frequent w.r.t. $\mathcal{D}$. This contradicts the fact that t is not removed from $\Pi(\sigma)$ in lines 20–25 in Procedure TestMaximality. Therefore, h and e have the same keyword.
- Suppose that h is labeled with some $/k/ \in KW$ and e is labeled with a tag $w \in Tag$ such that $w \in \Lambda_{\{/k/\}}$ holds. This contradicts the assumption of the problem **MFHCOTTP**.

If h is an edge then h and the unique edge of s have the same edge label.

From Cases 1 and 2, for each binding $h := \langle s, \tau\rangle$ in θ we see that $t\{h := \langle s, \tau\rangle\} \cong t$ holds. Thus, $t \cong t\theta$ holds. Therefore, we conclude that t is a maximally σ-frequent HC-tag tree pattern w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^h(Tag, KW)$. □

Next we discuss the complexity of the problem of enumerating all σ-frequent HC-tag tree patterns. An enumeration algorithm is *polynomial total time* if the time required to compute all solutions is bounded by a polynomial in the size of the input and the number of solutions [6].

Theorem 3 Algorithm Gen-MFHCOTTP computes the set $\Pi_4(\sigma)$ of all σ-frequent HC-tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^h(Tag, KW)$ in polynomial total time.

Proof. The problem of deciding whether an HC-tag tree pattern (resp. a tag tree pattern) is σ-frequent w.r.t. $\mathcal{D}$ is computable in polynomial time by using a polynomial time matching algorithm [12] (resp. [14]). A variable-only tree pattern is a tag tree pattern consisting of only vertices and variables [9]. Procedure EnumFreqTP outputs the set $\Pi_1(\sigma)$ of σ-frequent variable-only tree patterns in polynomial total time. Procedure ReplaceEdge2 makes a polynomial number of new candidate tag tree patterns by replacing each variable in an input tag tree pattern t with a labeled edge. That is, the number of new candidate tag tree patterns is $|H(t)| \times (|Tag \cup KW| + 1)$. Since $\Pi_1(\sigma) \subsetneqq \Pi_2(\sigma)$ holds, Procedure ReplaceEdge2 outputs the set Π_2 of all σ-frequent tag tree patterns in polynomial total time. Procedure MergeVariable makes a variable-chain free HC-tag tree pattern from each tag tree pattern t' in $\Pi_2(\sigma)$ in $O(|H(t')|)$ time. Note that $|\Pi_3(\sigma)| = |\Pi_2(\sigma)|$ holds. Procedure ConstrainVariable makes a polynomial number of new candidate HC-tag tree patterns by replacing each (i, j)-HC-variable of an input HC-tag tree pattern t'' with an $(i, j-1)$-HC-variable $(1 \leq i < j)$. That is, the number of new candidate HC-tag tree patterns is $|H(t'')| \times h_D$. Since $\Pi_3(\sigma) \subsetneqq \Pi_4(\sigma)$ holds, Algorithm Gen-MFHCOTTP computes the set $\Pi_4(\sigma)$ in polynomial total time. □

Unfortunately, Algorithm Gen-MFHCOTTP cannot output the set of all maximally σ-frequent HC-tag tree patterns w.r.t. $\mathcal{D}$ in $\mathcal{OTTP}^h(Tag, KW)$ in polynomial total time. The reason is shown with this counterexample. We assume that the set $\mathcal{D}_1$ consisting of only one tree T, $\sigma = 1.00$, $Tag = \emptyset$ and $KW = \emptyset$ are given as an input of Algorithm Gen-MFHCOTTP. The number of σ-frequent HC-tag tree patterns w.r.t. $\mathcal{D}_1$ is more than $2^{|E(T)|}$. However, the number of maximally σ-frequent HC-tag tree patterns w.r.t. $\mathcal{D}_1$ is just one. Thus, Algorithm Gen-MFHCOTTP cannot output in polynomial total time.

4. Experimental Results

In this section, we report experimental results of the proposed algorithm Gen-MFHCOTTP. We implemented Algorithm Gen-MFHCOTTP and our previous algorithm Gen-MFOTTP [9] in Java on a PC with 3.50 GHz processors, 32.0 GB of RAM on Windows 7 (64-bit). We compare experimental results of the two algorithms Gen-MFHCOTTP and Gen-MFOTTP. Algorithm Gen-MFHCOTTP uses the polynomial time pattern matching algorithm [12] as a subroutine.

In our experiments, we used glycan data extracted from the KEGG/GLYCAN database [7] as tree structured data. For example, the tree T in **Fig. 6** shows the tree structured data corresponding to the glycan data g.

Let $\mathcal{D}_{leu}$ be the glycan data related to leukemia and $\mathcal{D}_{non}$ the glycan data not related to leukemia. Then we have 177 trees in $\mathcal{D}_{leu}$ and 302 trees in $\mathcal{D}_{non}$. We set Tag = {“Gal–1b4”} and KW = {“/GlcNAc/”,“/NeuAc/”} as inputs of the algorithms, since the occurrences of these tags and keywords are higher than those of the other tags and keywords. We performed experiments in 20 experimental settings, by the two algorithms Gen-MFHCOTTP and Gen-MFOTTP, given two datasets $\mathcal{D}_{leu}$ and $\mathcal{D}_{non}$, for a threshold $\sigma \in \{1.00, 0.95, 0.90, 0.85, 0.80\}$. We have 10 runs for each experimental setting and measured the average run time of 10 runs. **Table 1** shows the experimental results of the proposed algorithm Gen-MFHCOTTP. We report the number of σ-frequent HC-tag tree patterns, the number of maximally σ-frequent HC-tag tree patterns and the average run time of Algorithm Gen-MFHCOTTP. **Table 2** shows the experimental results of the previous algorithm Gen-MFOTTP. We

Table 1 Experimental results of the proposed algorithm GEN-MFHCOTTP.

target dataset	$\mathcal{D}_{leu}$					$\mathcal{D}_{non}$				
threshold σ	1.00	0.95	0.90	0.85	0.80	1.00	0.95	0.90	0.85	0.80
number of σ-frequent HC-tag tree patterns	4	139	612	3387	30152	4	16	28	147	513
number of maximally σ-frequent HC-tag tree patterns	1	7	14	21	40	1	1	2	6	15
run time of Algorithm GEN-MFHCOTTP (ms)	21	56	204	1107	11001	15	14	48	107	386

Table 2 Experimental results of the previous algorithm GEN-MFOTTP.

target dataset	$\mathcal{D}_{leu}$					$\mathcal{D}_{non}$				
threshold σ	1.00	0.95	0.90	0.85	0.80	1.00	0.95	0.90	0.85	0.80
number of σ-frequent tag tree patterns	3	9	18	42	99	3	3	5	6	11
number of maximally σ-frequent tag tree patterns	1	2	4	6	8	1	1	1	2	2
run time of Algorithm GEN-MFOTTP (ms)	18	45	138	399	822	9	7	25	43	115

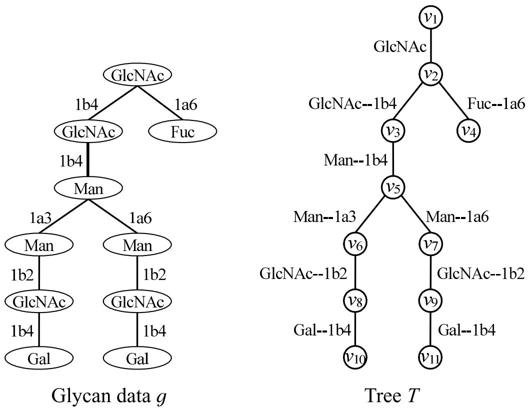


Fig. 6 Glycan data and corresponding tree structured data. We treat a tree corresponding to glycan data by regarding each vertex label in glycan data as the prefix of the edge label assigned to the edge adjacent to the vertex in the tree.

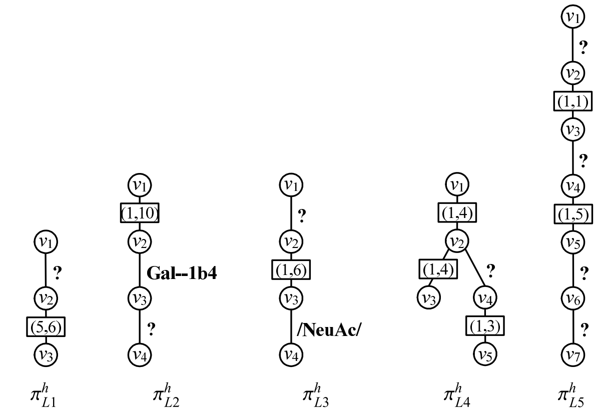


Fig. 7 Examples of maximally 0.80-frequent HC-tag tree patterns w.r.t. $\mathcal{D}_{leu}$ obtained by GEN-MFHCOTTP.

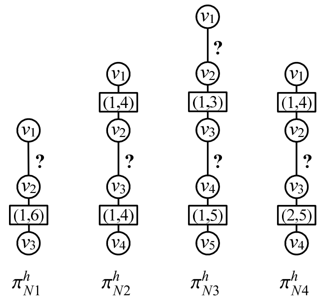


Fig. 8 Examples of maximally 0.80-frequent HC-tag tree patterns w.r.t. $\mathcal{D}_{non}$ obtained by GEN-MFHCOTTP.

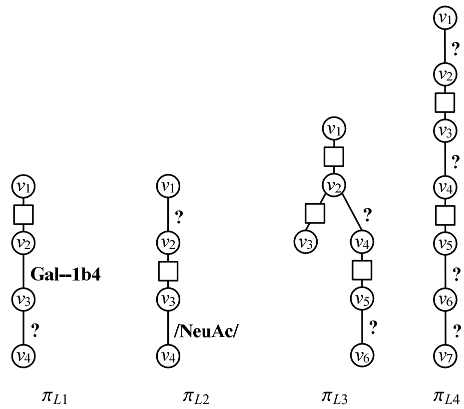


Fig. 9 Examples of maximally 0.80-frequent tag tree patterns w.r.t. $\mathcal{D}_{leu}$ obtained by GEN-MFOTTP.

also report the number of σ-frequent tag tree patterns, the number of maximally σ-frequent tag tree patterns and the average run time of Algorithm GEN-MFOTTP. **Figure 7** (resp. **Fig. 8**) shows examples of maximally 0.80-frequent HC-tag tree patterns w.r.t. $\mathcal{D}_{leu}$ (resp. $\mathcal{D}_{non}$) obtained by the proposed algorithm GEN-MFHCOTTP. Also, **Fig. 9** (resp. **Fig. 10**) shows examples of maximally 0.80-frequent tag tree patterns w.r.t. $\mathcal{D}_{leu}$ (resp. $\mathcal{D}_{non}$) obtained by the previous algorithm GEN-MFOTTP. As additional experiments, for Algorithm GEN-MFHCOTTP and the dataset $\mathcal{D}_{leu}$, we performed experiments in 21 experimental settings for a threshold $\sigma \in \{1.00, 0.99, 0.98, \ldots, 0.81, 0.80\}$. **Figure 11** shows the relationship between the average run time of Algorithm GEN-MFHCOTTP and the number of σ-frequent HC-tag tree patterns w.r.t. $\mathcal{D}_{leu}$.

Table 1 shows that as the threshold σ decreases, the number of σ-frequent HC-tag tree patterns and the average run time of Algorithm GEN-MFHCOTTP increase. Figure 11 shows that the average run time of Algorithm GEN-MFHCOTTP increases in proportion to the number of σ-frequent HC-tag tree patterns. The reason is that Algorithm GEN-MFHCOTTP computes all σ-frequent HC-tag tree patterns and checks whether or not each σ-frequent HC-tag tree pattern is maximally σ-frequent. From Table 1, for each threshold σ, the number of maximally σ-frequent HC-tag tree patterns is much smaller than that of σ-frequent HC-tag tree patterns. Therefore, Algorithm GEN-MFHCOTTP succeeds in reducing the number of candidate HC-tag tree patterns characteristic to $\mathcal{D}_{leu}$.

Tables 1 and 2 show that the run time of Algorithm GEN-MFHCOTTP is greater than that of Algorithm GEN-MFOTTP. This reason is as follows. Algorithm GEN-MFHCOTTP first computes all σ-frequent tag tree patterns by using procedures of

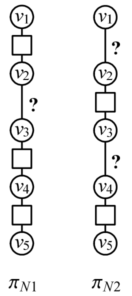


Fig. 10 Examples of maximally 0.80-frequent HC-tag tree patterns w.r.t. $\mathcal{D}_{non}$ obtained by Gen-MFOTTP.

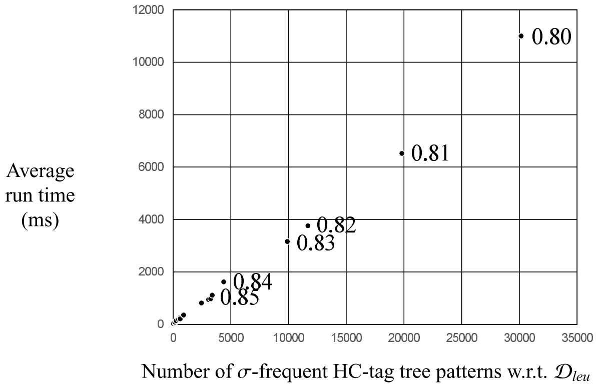


Fig. 11 Average run time of Algorithm Gen-MFHCOTTP. The value near each dot means threshold of each experimental setting.

Gen-MFOTTP. Algorithm Gen-MFHCOTTP computes all σ-frequent HC-tag tree patterns from all σ-frequent tag tree patterns. Therefore, for the same inputs, the run time of Algorithm Gen-MFHCOTTP is always greater than that of Gen-MFOTTP. In general, the number of maximally σ-frequent HC-tag tree patterns is larger than that of maximally σ-frequent tag tree patterns.

From the definition of bindings, the HC-tag tree pattern π^h_{L1} in Fig. 7 matches any tree such that the root of the tree has only one child and the height of the tree is at least 6 and at most 7. In comparison, the HC-tag tree pattern π^h_{N1} in Fig. 8 matches any tree such that the root of the tree has only one child and the height of the tree is at least 2 and at most 7. More than 80% of the trees in $\mathcal{D}_{leu}$ are of height 7. π^h_{L1} well represents structured features of the trees in $\mathcal{D}_{leu}$. Furthermore, the structural difference between π^h_{L1} and π^h_{N1} implies a difference between $\mathcal{D}_{leu}$ and $\mathcal{D}_{non}$. Knowledge on the height of the trees in $\mathcal{D}_{leu}$ and $\mathcal{D}_{non}$ cannot be obtained from the output tag tree patterns of Algorithm Gen-MFOTTP. Therefore, the results show the effectiveness of the proposed model of HC-tag tree patterns with HC-variables. The knowledge on the height of the trees is not always meaningful.

Furthermore, we can obtain knowledge on the distance between two edges. For example, from the HC-tag tree pattern π^h_{L3} in Fig. 7, the distance between the edges labeled with "?" and "/NeuAc/" is at most 6. Since this knowledge cannot be obtained from tag tree patterns without height constraint, HC-tag tree patterns are effective in representing tree structured features.

5. Conclusions

In this paper, we have presented a new refined model of characteristic tree structured features of structured data which are represented by rooted trees with ordered children, by extending our previous model of characteristic tree structured features, maximally frequent ordered tag tree patterns without height constraint [9]. As a new refined model of characteristic tree structured features, we have proposed height constrained ordered tag tree patterns, which are ordered tree patterns having height-constrained structured variables, wildcards, tags and keywords as edge labels.

First, we have stated that it is hard to compute a maximally frequent height-constrained ordered tag tree pattern of maximum tree-size. Then, we have presented an algorithm for enumerating all maximally frequent height-constrained ordered tag tree patterns. Finally, we have reported experimental results showing the effectiveness of the proposed model of characteristic tree structured features, maximally frequent height-constrained ordered tag tree patterns, compared with the previous model [9]. As future work, we will study more efficient algorithms for enumerating characteristic HC-tag tree patterns from the viewpoint of the theory of enumeration algorithms. Furthermore, we will develop a robust and scalable algorithm for enumerating characteristic tree patterns from large amount of tree structured data.

Acknowledgments We would like to thank the anonymous reviewers for their valuable and helpful comments on our manuscript. This work was partially supported by Grant-in-Aid for Scientific Research (C) (Grant Numbers 15K00312, 15K00313, 17K00321) from Japan Society for the Promotion of Science (JSPS).

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] Jiang, C., Coenen, F. and Zito, M.: A survey of frequent subgraph mining algorithms, *The Knowledge Engineering Review*, Vol.28, No.1, pp.75–105 (2013).

[6] Johnson, D., Yannakakis, M., Papadimitriou, C.: On generating all maximal independent sets, *Information Processing Letters*, Vol.27, pp.119–123 (1988).

[7] KEGG GLYCAN Database, available from ⟨https://www.genome.jp/kegg/glycan/⟩ (accessed 2018-11-16).

[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).

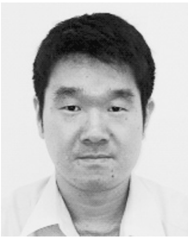
[9] Miyahara, T., Suzuki, Y., Shoudai, T., Uchida, T. and Kuboyama, T.: Enumeration of Maximally Frequent Ordered Tree Patterns with Wildcards for Edge Labels, *IPSJ Trans. Math. Model. Appl.* (*TOM*), Vol.10, No.2, pp.59–69 (2017).

[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] Shoudai, T., Aikoh, K., Suzuki, Y., Matsumoto, S., Miyahara, T. and Uchida, T.: Polynomial Time Inductive Inference of Languages of Ordered Term Tree Patterns with Height-Constrained Variables from Positive Data, *IEICE Trans. Fund.*, Vol.E100-A, No.3, pp.785–802 (2017).

[13] 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).
[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).



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

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, his 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.

Takayoshi Shoudai received his B.S. in 1986, his M.S. degree in 1988 in Mathematics and his Dr. Sci. in 1993 in Information Science all from Kyushu University. 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.

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

Satoshi Matsumoto is an associate professor of Department of Mathematical Sciences, Tokai University, Kanagawa, Japan. He received his B.S. degree in Mathematics, his M.S. and Dr. Sci. degrees in Information Systems all from Kyushu University, Fukuoka, Japan in 1993, 1995 and 1998, respectively. His research interests include algorithmic learning theory.

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.

