An Algorithm for Enumerating All Maximal Tree Patterns Without Duplication Using Succinct Data Structure

Yuko ITOKAWA, Tomoyuki UCHIDA and Motoki SANO

***Abstract*—In order to extract structured features from big tree-structured data, a fast and memory-efficient tree mining algorithm is needed. In this paper, we propose an efficient algorithm, given a set S of ordered trees as input, to enumerate all maximal ordered tree patterns explaining all ordered trees in S without duplication. The enumeration algorithm uses a depth-first unary degree sequence (DFUDS), which is a succinct data structure for an ordered tree, as a succinct data structure for ordered tree patterns that express structured features of a tree structure. We also implement the proposed algorithm on a computer and evaluate the algorithm by experiments. The results are reported and discussed.**

***Index Terms*—tree-mining algorithm, enumeration of tree patterns, succinct data structure, tree-structured data**

I. Introduction

Web documents consist of HTML/XML data and other tree-structured data, such as TEX sources, natural languages and so on, whose structure is not entirely clear is referred to as tree-structured data. Tree-structured data can be represented by an ordered tree. To extract useful information from tree-structured data, it is necessary to extract tree patterns that are common to tree-structured data. To design efficient tree mining tools for big tree-structured data, the following fast and memory-efficient algorithms are necessary to be efficient. One is a pattern matching algorithm for determining whether or not given tree-structured data has structured features represented by a given tree pattern. Another is an algorithm for extracting common structured features to given tree structured data. To reduce the memory required to store an ordered tree, succinct data structures for ordered trees have been proposed [1], [2], [3], [5], [7], [8], [10]. D. Benoit et al. proposed a depth-first unary degree sequence (**DFUDS**) representation [1] as a succinct data structure for an ordered tree. The DFUDS representation uses a string of parentheses constructed by a depth-first traversal of all nodes in which, if the index of a node is k, the k-th $\boxed{(}$ and its subsequent $\boxed{)}$ are output. By taking $\boxed{(}$ to be '0' and $\boxed{)}$ to be '1', the ordered tree representation can be handled as a bit string. As previous work [6], we proposed a term tree pattern as a tree pattern which can represent structured features common to tree-structured data. We also defined a DFUDS representation for a term tree pattern based on a DFUDS representation of an ordered tree [6]. In Fig. 1, we give term tree pattern t and s as examples. Moreover, in Fig. 2, we show the DFUDS representation of the term tree pattern t as an example.

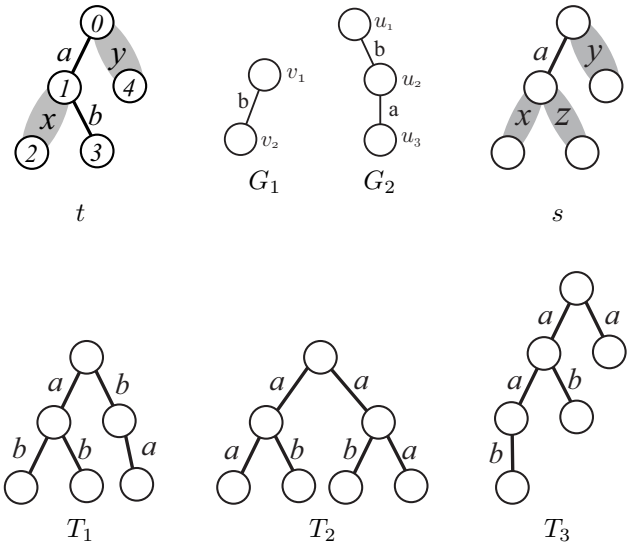

Fig. 1. Term tree patterns t, s, ordered trees G_1, G_2, T_1, T_2, T_3

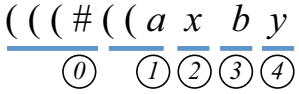

Fig. 2. DFUDS representation of the term tree pattern t given in Fig. 1. The indexes under DFUDS representation show node IDs.

If an ordered tree T is obtained from a term tree pattern t by substituting all structured variables in a given term tree pattern with arbitrary ordered trees, t is said to match with T. In [6], we proposed a fast pattern matching algorithm for determining whether or not a given ordered tree matches with a given term tree pattern, using DFUDS as a data structure. For example, in Fig. 1, we can see that the ordered tree T_1 matches the term tree pattern t, since T_1 can be obtained from t by substituting variables x and y with ordered trees G_1 and G_2, respectively. For a term tree pattern t, the set of all ordered trees which match with t is denoted by $L(t)$. When a set S of term tree patterns is given, if there exists no term tree pattern s such that $S \subseteq L(s) \subseteq L(t)$ holds, t is said to be maximal with respect to S. For example, in Fig. 1, the term tree pattern s isn't maximal with respect to the set $S = \{T_1, T_2, T_3\}$ of ordered trees T_1, T_2, T_3. On the other hand, the term tree pattern t is maximal with respect to S. The aim of this paper is to present fast and memory-efficient algorithms for enumerating all maximal term tree patterns which can represent all ordered trees in a given set of ordered trees. As related works, Miyahara et al.

Y. Itokawa is with Faculty of Psychological Science, Hiroshima International University. 555-36 Kurose-Gakuendai, Higashi-Hiroshima, Hiroshima Japan e-mail: y-itoka@he.hirokoku-u.ac.jp

T. Uchida and M. Sano are with Department of Intelligent Systems, Hiroshima City University. Hiroshima, Japan, email: uchida@hiroshima-cu.ac.jp

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[9] presented the algorithm GEN-MFOTTP for generating all maximal ordered tag tree patterns that can represent structured features common to a given set of ordered trees. A tag tree pattern is a variant of a term tree pattern treated in this paper. GEN-MFOTTP employs the pattern matching algorithm for tag tree patterns presented by Suzuki et al. [12]. We already showed in [6] that Itokawa's pattern matching algorithm is faster than Suzuki's algorithm. In this paper, we show that a proposed algorithm in this paper is faster than GEN-MFOTTP.

This paper is organized as follows. In section II-A, we introduce a term tree pattern which represents the structured features of ordered trees. In section II-B, we describe the DFUDS representation for ordered term tree patterns proposed by Itokawa et al. [6]. In section III, we formulate a problem for enumerating all maximal term tree patterns whose languages contain a given set of edge-labeled trees as a subset and propose an enumeration algorithm for solving it. In section IV, the proposed enumeration algorithm is implemented on a computer. The test results for the implementation are reported and discussed. Section V concludes the paper.

II. Preliminaries

A. Term Tree Patterns

Let Σ and χ denote finite alphabets with $\Sigma \cap \chi = \emptyset$. Elements of Σ and χ are called a **edge label** and a **variable label**, respectively. Let V_t be a set of nodes and $E_t \subseteq V_t \times (\Sigma \cup \chi) \times V_t$ a set of edges. An edge labeled with a variable label is particularly called a **variable**. For an variable $e = (u, x, v)$, u and v are called a **parent port** and a **child port** of e, respectively. $t = (V_t, E_t)$ is called an **edge-labeled ordered term tree pattern** or simply **term tree pattern** if t has only one node u whose coming degree is 0, $(V_t, \{\{u, v\} \mid (u, a, v) \in E_t\}$ is a rooted tree having u as its root and any its internal nodes have ordered children. In this paper, we deal with term tree patterns having all mutually different variables. $\mathcal{OTTP}$ denotes the set of all term tree patterns having all mutually different variables. A term tree pattern having no variables is simply a **tree**. The set of all trees is denoted by $\mathcal{OT}$. The last leaf of a term tree pattern t in preorder is called a **rightmost leaf** of t. Moreover, the path from the rightmost leaf to the root is called a **rightmost path** of t.

For two children u' and u'' of a node u of a term tree pattern h, $u' <_u^h u''$ denotes that in the ordering of the children of u, u' is lower than u''. For term tree patterns $t = (V_t, E_t)$ and $f = (V_f, E_f)$, if a bijection $\pi : V_t \to V_f$ that satisfies the following conditions (1)-(3) exists, then t and f are **isomorphic**, denoted by $t \cong f$.

(1) For any $a \in \Sigma$, $(u, a, v) \in E_t$ if and only if $(\pi(u), a, \pi(v)) \in E_f$.
(2) There is $x \in \chi$ so that $(u, x, v) \in E_t$, if and only if there is $y \in \chi$ so that $(\pi(u), y, \pi(v)) \in E_f$.
(3) For a node u of t and two children u' and u'' of u, $u' <_u^t u''$ if and only if $\pi(u') <_{\pi(u)}^f \pi(u'')$.

For a variable label $x \in \chi$ and a tree g having r as its root and ℓ as its leaf, the form $x := [g, (r, l)]$ is called a **binding** of x and a finite set of bindings is called a **substitution**. Let $g = (V_g, E_g)$ be a term tree pattern having variables labeled with $x_0, x_1, \ldots, x_n$ and $\theta = \{x_0 := [g_0, (r_0, \ell_0)], \ldots, x_n := [g_n, (r_n, \ell_n)]\}$ a substitution. A new term tree pattern f can be obtained from g and θ by applying θ to g in the following way. f is obtained by, for each $0 \leq i \leq n$, identifying the parent port of the variable having the variable label x_i with r_i and identifying the child port with ℓ_i, and by removing the variable labeled with x_i. The resultant term tree pattern f is denoted by $g\theta$. For example, in Fig. 1, we can see that the tree T_1 is obtained the term tree pattern t by applying the substitution $\theta = \{x := [G_1, (v_1, v_2)], y := [G_2, (u_1, u_3)]\}$ to t, that is, $T_1 \cong t\theta$.

We have the following lemma.

Lemma *1:* Let t and s be term tree patterns such that $t \ncong s$ holds. Then, $L(t) \subset L(s)$ if and only if there exists a substitution θ such that $t \cong s\theta$ holds.

For a term tree pattern t, the set, denoted by $L(t)$, of trees obtained from t by substituting all of variables in t with appropriate trees, that is, $L(t) = \{T \in \mathcal{OT} \mid \exists\theta \text{ s.t. } T \cong t\theta\}$. For a set of trees S and a term tree pattern t in $\mathcal{OTTP}$, the language $L(t)$ of t is said to be **minimal** with respect to S if there exists no term tree pattern s in $\mathcal{OTTP}$ such that $S \subseteq L(s) \subseteq L(t)$ holds. Such a term tree pattern t is said to be **maximal** with respect to S. For example, in Fig. 1, the term tree pattern s isn't maximal with respect to the set $S = \{T_1,\ T_2,\ T_3\}$ because of $S \subset L(t) \subset L(s)$. On the other hand, we can see that t is maximal with respect to S, that is, $L(t)$ is minimal with respect to S.

B. Succinct Data Structures for Term Tree Patterns

We explain the basic data structure for dealing with a term tree pattern. In this paper, a word RAM with a word length of $\Theta(\log n)$ bits is used as the computation model. [1] proposed the depth-first unary degree sequence (**DFUDS**) representation, which is a succinct data structure for ordered trees. The DFUDS representation for an ordered tree t of m edges is defined inductively as follows. The DFUDS representation of the tree consisting only one node is $\boxed{(}\ \boxed{)}$. The DFUDS representation of a t that has k subtrees $t_1, \ldots, t_k$ is a sequence of parentheses constructed by concatenating $k+1$ $\boxed{(}$, one $\boxed{)}$, k DFUDS representations of $t_1, \ldots,$ and t_k in this order (here, the initial $\boxed{(}$ of the DFUDS representation of each subtree has been removed). The DFUDS representation is a sequence of balanced parentheses of length $2m$.

The DFUDS representation proposed by [1] is a data structure for an ordered tree with no edge labels. In the DFUDS representation of [1], $\boxed{)}$ must occupy the rightmost position for each node. Therefore, a hash function that returns the edge label that corresponds to that node for each $\boxed{)}$ makes a DFUDS representation of a tree possible. DFUDS representation for the term tree pattern t is shown in Fig. 2. For convenience in this example, a hash function that returns the edge labels that correspond to all of the $\boxed{)}$ has been executed. For a term tree pattern t, the length of DFUDS representation of t is denoted by $\|t\|$.

The sequence of parentheses that is a DFUDS representation can be interpreted as the result of visiting all nodes in preorder and outputting k $\boxed{(}$ for each node whose index is k the following one $\boxed{)}$. Hence, the following lemma obviously holds.

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

LEMMA *2:* ([6]) Given a term tree pattern t of m edges, the edge-labeled DFUDS representation for t can be computed in $O(m)$ time.

In [6], we presented an efficient algorithm for determining whether or not, for a given term tree pattern $t \in \mathcal{OTTP}$, the language $L(t)$ of t contains a given tree $T \in \mathcal{OT}$. We showed the following lemma.

LEMMA *3:* ([6]) When an ordered tree T and a term tree pattern t are given, we can determine whether or not T match with t in $O(\|T\| \times \|t\|)$ time.

III. ENUMERATION ALGORITHM FOR TERM TREE PATTERNS

In this section, we present an efficient algorithm for enumerating all of term tree patterns whose languages are minimal among term tree languages including a given set of ordered trees. We consider the following Enumerating Maximal TTPs Problem in this paper.

Enumerating Maximal TTPs Problem
Instance: Set of trees S
Question: Enumerate all term tree patterns whose languages include S.

In Algorithm 1, we give an efficient algorithm ENUMERATIONMAXIMALTTPS for solving Enumerating Maximal TTPs problem for a given set of trees. ENUMERATIONMAXIMALTTPS consists of three procedures, ENUMTREEPATTERNS, EDGESUBST$_S$ and MAXCHECK$_S$. Given a set of trees S, the first procedure ENUMTREEPATTERNS, presented in Procedure 2, enumerates all term tree patterns having no edges each of whose language contains S as a subset. ENUMTREEPATTERNS is an enumeration algorithm based on level-wise strategy in a similar way to the algorithm for enumerating ordered trees without duplication presented by Nakano [11]. ENUMTREEPATTERNSemploys a procedure, denoted by **RightmostExpansion**, which creates new term tree patterns obtained from a given term tree pattern t by attaching new one variable to each node in the rightmost path of t. Fig.3 explains about a detail of RightmostExpansion. For each variable (v_2, x, v_1) in the rightmost path of a term tree pattern g with k nodes, we add a node u_p such that $v_1 <^g_{v_2} u_p$ to g by connecting a variable (v_2, x, u_p). Then we can obtain a new term tree pattern g_1 with $k+1$ nodes. For the rightmost leaf v_l of g, we add a node u_s to g by attaching a variable (v_l, x, u_s), and we can obtain a new term tree pattern g_2 with $k+1$ nodes. Given a set E of term tree patterns created by ENUMTREEPATTERNS, the second procedure EDGESUBST$_S$, presented in Procedure 3, creates new term tree patterns obtained from term tree patterns in E by replacing variables with edges labeled with elements in Σ. Given a set D of term tree patterns created by EDGESUBST$_S$, MAXCHECK$_S$ presented in Procedure 4, select all maximal term tree patterns with respect to S in D by checking for each term tree pattern $t \in D$ whether or not there exists a term tree pattern s in D such that $L(s) \subseteq L(t)$ holds. For a term tree pattern $s \in D$, let s^* be a tree which is created from s by replacing all variables in s with an edge label $e \notin \Lambda$. For a term tree pattern $t \in D$, if s^* matches to t, then $L(s) \subseteq L(t)$, that is, we can determine that t is not a maximal term tree pattern.

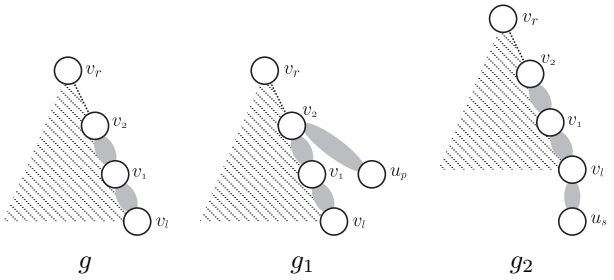

Fig. 3. Rightmost Expansion from g to g_1 and g_2

Algorithm 1 ENUMERATIONMAXIMALTTPS

Require: set of trees S
Ensure: set of maximal term tree patterns R whose languages contain S as a subset.
1: $E \leftarrow$ ENUMTREEPATTERNS(S)
2: $D \leftarrow$ EDGESUBST$_S(E)$
3: $R \leftarrow$MAXCHECK$_S(D)$
4: **return** R

In ENUMERATIONMAXIMALTTPS, we use a tree structure, called an **enumeration tree**, to manage enumerated term tree patterns. The initial term tree pattern created in line 1 of ENUMTREEPATTERNS is stored in the root of the enumeration tree. Let p be a term tree pattern created from a term tree pattern t in the procedure RightmostExpansion of line 6 in ENUMTREEPATTERNS or in EDGESUBST$_S$. Then, in the constructed enumeration tree in ENUMERATIONMAXIMALTTPS, p is stored in a child of the node storing t if $L(t) \supseteq S$ holds. Using this data structure, we can restrict an input set of term tree patterns of MAXCHECK$_S$ to the set of all leaves of the enumeration tree constructed at the end of ENUMTREEPATTERNS. For example, given the set $S = \{T_1, T_2, T_3\}$ of ordered trees T_1, T_2, T_3 in Fig. 1, ENUMERATIONMAXIMALTTPS constructs the enumeration tree in Fig. 4 at the end of ENUMTREEPATTERNS . Moreover, in Fig. 5, we show the descendant of the node storing the term tree pattern g_5 in the constructed enumeration tree at the end of EDGESUBST$_S$. In Figs 4 and 5, the $\times$ mark (cross) on a term tree pattern shows that its language doesn't contain the set S as a subset. In Fig. 6, we show the enumeration tree each of whose nodes has DFUDS representation of the assigned term tree pattern. When a term tree pattern t is generated from a term tree pattern s by the procedure RightmostExpansion, we can see that the DFUDS representation of t can be made by inserting $\boxed{(}$ at appropriate index and appending the variable label at the end of the DFUDS representation of s. For example, in Fig. 6, we can see that the DFUDS representation '(((#((xxxx' of the term tree pattern g_{10} can be obtained from the DFUDS representation '((#((xxx' of the term tree pattern g_4 by inserting $\boxed{(}$ at the index 2 and appending 'x' at the end of '((#((xxx'. Moreover, when a term tree pattern t is generated from a term tree pattern s by the procedure EDGESUBST$_S$, we can see that the DFUDS representation of t can be made by replacing the variable label at appropriate index of the DFUDS representation of s with an edge label. For example, in Fig. 6, we can see that the DFUDS representation '(((#(axx' of the term tree pattern $g_{5,0}$ can be obtained from

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Procedure 2 EnumTreePatterns

Require: set of trees S
Ensure: set E of all term tree patterns whose languages contain S as a subset

Let p be the term tree pattern consisting of only one variable
$F \leftarrow \{p\}, E \leftarrow \emptyset$
while F is not empty **do**
  $C \leftarrow \emptyset, D \leftarrow \emptyset$
  for all $t \in F$ **do**
    $C \leftarrow$ RightmostExpansion(t)
    for all $f \in C$ **do**
      if $L(f) \supseteq S$ **then**
        $E \leftarrow E \cup \{f\}, D \leftarrow D \cup \{f\}$
      end if
    end for
  end for
  $F \leftarrow D$
end while
return E

Procedure 3 EdgeSubst$_S$

Require: set of term tree patterns E created in EnumTreePatterns
Ensure: set D of all term tree patterns whose languages contain S as a subset

$D \leftarrow E$
for all $t \in E$ **do**
  $U \leftarrow \{(t, 0)\}$
  while $U \neq \emptyset$ **do**
    for all $(s, pos) \in U$ **do**
      $U \leftarrow U - \{(s, pos)\}$
      for all variable e in s appeared after index pos **do**
        for all edge label $a \in \Sigma$ **do**
          create a new term tree pattern p by replacing e with the edge labeled with a
          if $L(p) \supseteq S$ **then**
            $D \leftarrow D \cup \{p\}$
          end if
          let pos_e be the appeared index of e in DFUDS representation of s
          if there are variables in e appeared after $pos_e + 1$ **then**
            $U \leftarrow U \cup \{(p, pos_e + 1)\}$
          end if
        end for
      end for
    end for
  end while
end for
return D

Procedure 4 MaxCheck$_S$

Require: set of term tree patterns D created in EdgeSubst$_S$
Ensure: set R of all maximal term tree patterns whose languages contain S as a subset

$R \leftarrow \emptyset$
for all $t \in D$ **do**
  create the new term tree pattern p by replacing all variables in t with edges labeled with $c \notin \Sigma \cup \chi$
  check $\leftarrow$ false
  for all $s \in D$ $(t \neq s)$ **do**
    if $p \in L(s)$ **then**
      check $\leftarrow$ true
    end if
  end for
  if check=false **then**
    $R \leftarrow R \cup \{t\}$
  end if
end for
return R

the DFUDS representation '(((#(xxx' of the term tree pattern g_5 by replacing the variable at the index 5 of the DFUDS representation '(((#((xxx' of the term tree pattern g_5 with the edge label 'a'. Therefore, by assigning an integer i or an pair (j, a), instead of a term tree pattern, to a node of the constructed enumeration tree, where i indicates the index inserting $\boxed{(}$ and j indicates the index replacing variable label with the edge label a. For example, in Fig. 7, we show the compact expression corresponding to the enumeration tree grown by EnumerationMaximalTTPs.

From lemmas 1, 2, 3 and the proposed algorithm EnumerationMaximalTTPs, we have the following theorem.

Theorem *1:* We can enumerate all maximal term tree patterns with respect to a given set of ordered trees without duplication in incremental polynomial time.

IV. Experiment and Discussion

We report results of evaluate experiments of an algorithm for generating the enumeration tree. We implemented the algorithm described in section III on a computer. In this section, we describe the experimental setup, present the results.

The algorithm EnumerationMaximalTTPs for enumerating all maximal term tree patterns of an ordered tree set S was implemented in C++ on a computer equipped with a 2.8 GHz Intel Core i7 processor, main memory of 16.00 GB and running the apple OSX 10.8.5 operating system.

Let a, b be an edge label in Σ. We have artificially created data collections $D(n)$ of one hundred each of edge-labeled trees in $\mathcal{OT}_\Sigma$ that has the edge label a, b, n edges and maximum degree 5.

For each $n \in \{100, 200, 300, 400, 500, 600, 700, 800\}$, we have generated maximal term tree patterns by EnumerationMaximalTTPs using data collections $D(n)$ as inputs. Fig.8 shows the results of numbers of generated all term tree patterns, numbers of element of generated enumeration tree, numbers of leaves of generated enumeration tree, and numbers of enumerated maximal term tree patterns, respectively.

We compared EnumerationMaximalTTPs with the algorithm GEN-MFOTTP[9]. The algorithm GEN-MFOTTP is as follows.

1) Generate a set of term tree patterns E by using EnumTreePatterns .

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

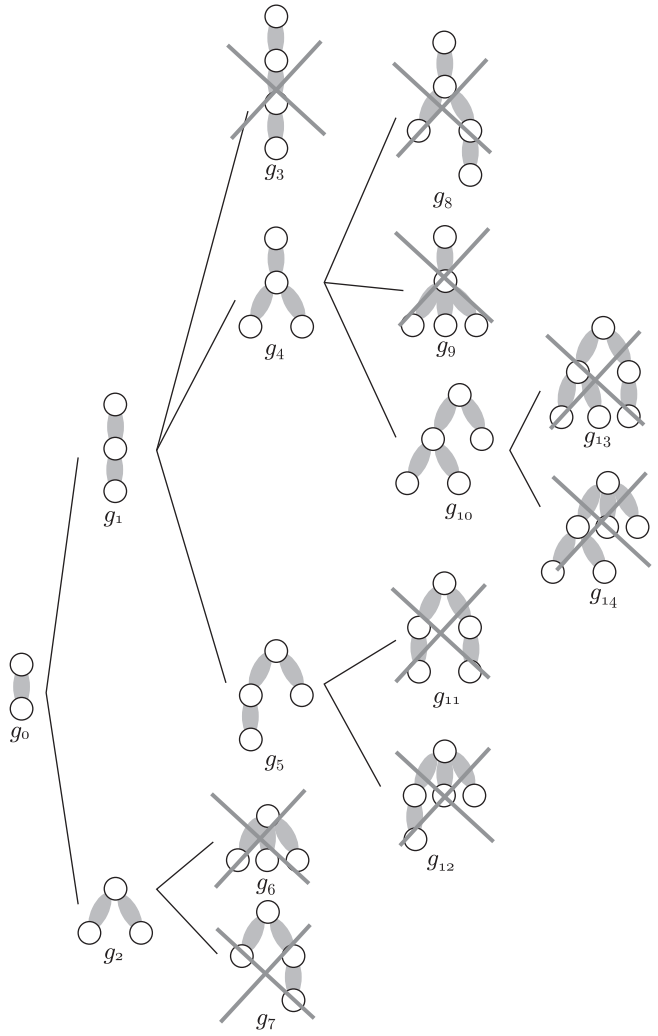

Fig. 4. Enumeration tree at the end of the Procedure ENUMTREEPATTERNS when S in Fig. 1 is given

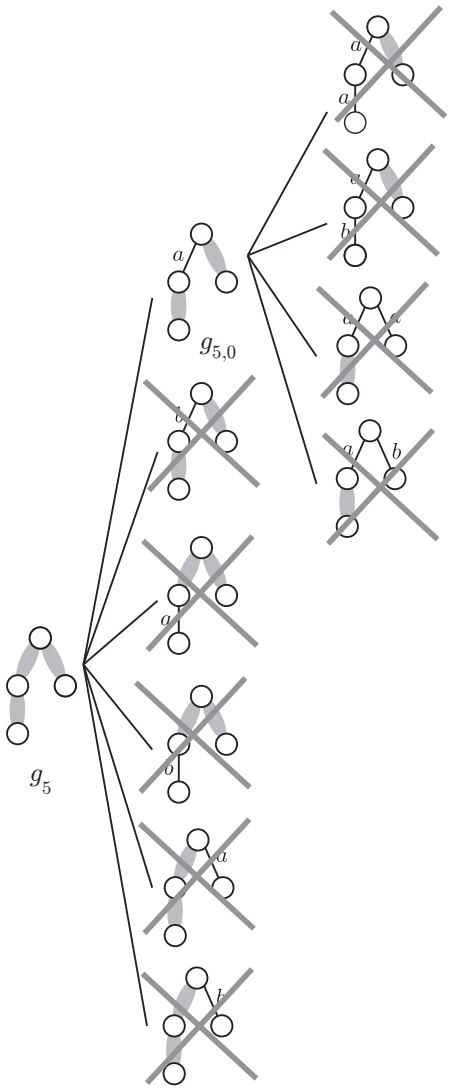

Fig. 5. Descendants of the term tree pattern g_5 in constructed enumeration tree at the end of the procedure EDGESUBST$_S$ when S in Fig. 1 is given

2) For each variables x in $t \in E$, we execute following operations:
   - Generate a term tree pattern t_p by replacing x to g_p of Fig.9. If $L(t_p) \supseteq S$, t is not a maximal term tree patterns.
   - Generate a term tree pattern t_s by replacing x to g_s of Fig.9. If $L(t_s) \supseteq S$ then t is not a maximal term tree patterns.
   - For $l \in \Lambda$, let g_l be a tree having only one edge labeled with l (Fig.9). Generate a term tree pattern t_l by replacing x to g_l. If $L(t_l) \supseteq S$ then t is not a maximal term tree pattern.
3) t is a maximal term tree patterns.

In this experiment, we have used a DFUDS as data structure and a matching algorithm proposed by [6] for matching term tree patterns to trees in GEN-MFOTTP. We have obtained running times of enumerating all maximal term tree patterns by ENUMERATIONMAXIMALTTPS and by GEN-MFOTTP. Those results are shown in Fig.10. An approximation for ENUMERATIONMAXIMALTTPS is $y = 0.36\exp(1.23n)$, and an approximation for GEN-MFOTTP is $y = 0.58\exp(1.48n)$.

Notice that the difference between running times of ENUMERATIONMAXIMALTTPS and GEN-MFOTTP is a cost of MAXCHECK$_S$ because for ENUMERATIONMAXIMALTTPS and GEN-MFOTTP, we adopted the same data structure and matching algorithm. In Fig.10, by comparing ENUMERATIONMAXIMALTTPSand GEN-MFOTTP, we can see that the running time is improved.

V. CONCLUSION

We have proposed an effective algorithm enumerating all maximal term tree patterns whose languages include a given tree set by employing the efficient matching algorithm presented in [6] using a succinct data structure, called DFUDS. Evaluation experiments performed with computer implementation of the algorithm demonstrated its efficiency.

As applications of this research, we are considering adaptation of the DFUDS representation proposed here to succinct data structures of TTSP graphs based on forest representation proposed by Miyoshi et al. [5] and the compressed tree proposed by Kato et al. [4]. Moreover, we are considering graph mining algorithms for semi-structured data using a succinct data structure.

REFERENCES

[1] D. Benoit, E. D. Demaine, J. I. Munro, and R. Raman. Representing trees of higher degree. *Algorithmica*, 43(4):275–292, 2005.

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

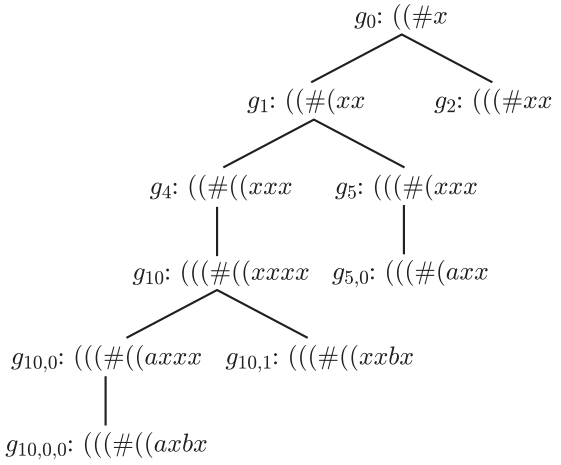

Fig. 6. Enumeration tree having DFUDS representations of enumerated term tree patterns

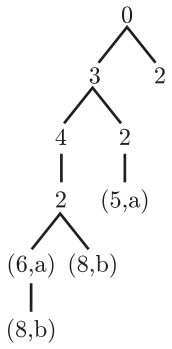

Fig. 7. Compact expression of constructed enumeration tree

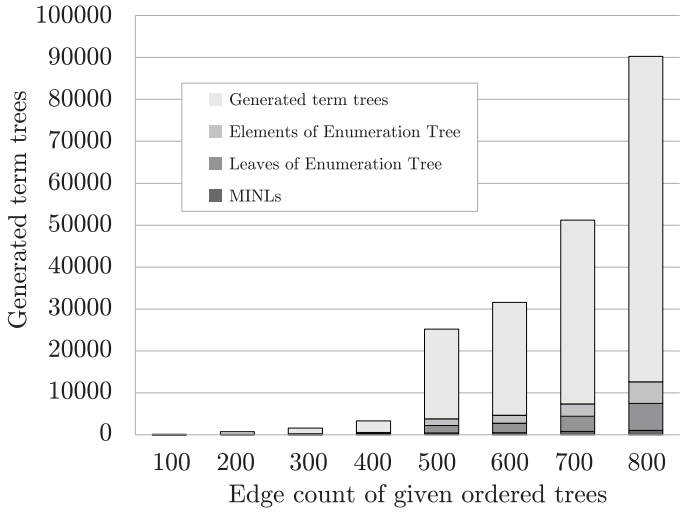

Fig. 8. The results of Numbers of generated all term trees, elements of generated enumeration tree, leaves of generated enumeration tree and generated maximal term tree patterns.

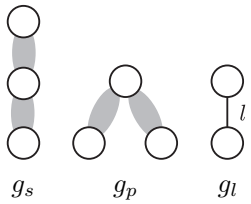

Fig. 9. Term trees using for the algorithm GEN-MFOTTP

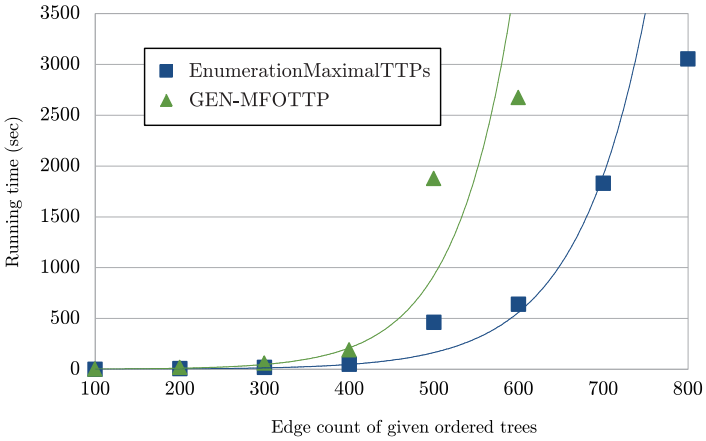

Fig. 10. Running times of enumeration of maximal term tree patterns by EnumerationMaximalTTPs and GEN-MFOTTP

Science and Engineering, pages 33–40, 2010.

[6] Y. Itokawa, M. Wada, T. Ishii, and T. Uchida. *Pattern Matching Algorithm Using a Succinct Data Structure for Tree-Structured Patterns*, pages 349–361. Lecture Notes in Electrical Engineering 110. Springer, 2012.

[7] G. Jacobson. Space-efficient static trees and graphs. In *IEEE FOCS*, pages 549–554, 1989.

[8] J. Jansson, K. Sadakane, and W.-K. Sung. Ultra-succinct representation of ordered trees. In *ACM-SIAM SODA 2007*, pages 575–584, 2007.

[9] T. Miyahara, Y. Suzuki, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of maximally frequent tag tree patterns with contractible variables from semistructured documents. In *PAKDD-2004*, LNAI 3056, pages 133–144. Springer, 2004.

[10] J. I. Munro and V. Raman. Succinct representation of balanced parentheses and static trees. *SIAM Journal on Computing*, 31(3):762–776, 2001.

[11] S. Nakano. Efficient generation of plane trees. *Inf. Process. Lett.*, 84(3):167–172, 2002.

[12] Y. Suzuki, K. Inomae, T. Shoudai, T. Miyahara, and T. Uchida. A polynomial time matching algorithm of structured ordered tree patterns for data mining from semistructured data. In *ILP-2002*, LNAI 2583, pages 270–284. Springer, 2003.

[2] Y.-T. Chiang, C.-C. Lin, and H.-I. Lu. Orderly spanning trees with applications. *SIAM Journal on Computing*, 34(4):924–945, 2005.

[3] R. F. Geary, N. Rahman, R. Raman, and V. Raman. A simple optimal representation for balanced parentheses. In *CPM*, pages 159–172, 2004.

[4] Y. Itokawa, K. Katoh, T. Uchida, and T. Shoudai. *Algorithm using Expanded LZ Compression Scheme for Compressing Tree Structured Data*, pages 333–346. Lecture Notes in Electrical Engineering. Springer, 2010.

[5] Y. Itokawa, J. Miyoshi, M. Wada, and T. Uchida. Succinct representation of ttsp graphs and its application to the path search problem. In *Sixth IASTED International Conference on Advances in Computer*

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)