Learning Unordered Tree Contraction Patterns in Polynomial Time

Yuta Yoshimura and Takayoshi Shoudai

Department of Informatics, Kyushu University, Fukuoka 819-0395, Japan
{yuuta.yoshimura,shoudai}@inf.kyushu-u.ac.jp

Abstract. In this paper, we present a concept of edge contraction based tree structured patterns as a graph pattern suited for representing tree structured data. A *tree contraction pattern* (*TC-pattern*) is an unordered tree structured pattern common to given tree structured data, which is obtained by merging every uncommon connected substructure into one vertex by edge contraction. In this paper, in order to give an algorithmic foundation of discovering knowledge from tree structured data, we show that TC-patterns are learnable in polynomial time.

1 Introduction

Many documents such as Web documents or XML files have tree structures. In order to extract meaningful and hidden knowledge from such documents, we need tree structured patterns which can explain them. As unordered tree structured patterns, a tree pattern [1], a type of objects [3], a tree-expression pattern [6], an unordered term tree [2] were proposed. In this paper, we introduce edge contraction based tree structured patterns, called *tree contraction patterns*, as graph patterns which are learnable in polynomial time.

A linear (or regular) unordered term tree (a term tree for short) is an unordered tree pattern which consists of internal variables and tree structures (please see [2, 4, 5] for details). A variable is a hyperedge of dimension 2, which can be replaced with an arbitrary tree. In this paper, a tree means a rooted unordered labeled tree. In Fig. 1, we give examples of term trees $s_1, \ldots, s_5$. We say that a tree T matches a term tree t if t becomes isomorphic to T by substituting a suitable tree for each variable. For example, in Fig. 1, all trees $T_1, \ldots, T_4$ match all $s_1, \ldots, s_5$. In [2], we showed that the matching problem for extended term trees with variables of dimension 4 is NP-complete, and presented a pattern matching algorithm for a term tree t and a tree T, which runs in $O(n^2N^{3.5})$ time, where n and N are the numbers of vertices of t and T, respectively.

A *tree contraction pattern* (*TC-pattern*) is a triplet $t = (V_t, E_t, U_t)$ where (V_t, E_t) is a tree with a specified root and U_t is a subset of V_t. A tree $T = (V_T, E_T)$ with root r_T matches a TC-pattern t with root r_t if there is a partition $\mathcal{W} = \{W(u) \mid u \in V_t\}$ of V_T such that (i) for $u \in V_t \setminus U_t$, $W(u)$ includes exactly one vertex, (ii) for any $u \in V_t$, any pair of vertices in $W(u)$ is connected, (iii) $W(r_t)$ includes r_T, and (iv) the tree obtained from T by merging all vertices

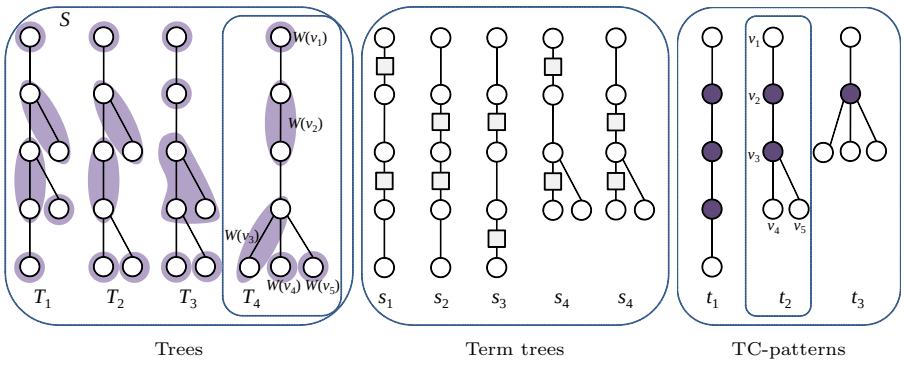

Fig. 1. All $T_1, \ldots, T_4$ match all term trees $s_1, \ldots, s_5$ and TC-patterns t_1, t_2, t_3. A variable of a term tree is represented by a box with lines to its elements. A contractible vertex of a TC-pattern is drawn with a dark-colored circle.

in $W(u)$ into one vertex for each $u \in U_t$ is isomorphic to (V_t, E_t). In the right frame of Fig. 1, we give three TC-patterns. A vertex in U_t is drawn with a dark-colored circle. Tree T_4 matches TC-pattern t_2 because there is a partition $\mathcal{W}$ of the vertex set of T_4, which is described by dark-colored areas, satisfying the above conditions (i)–(iv). It is easy to see that all trees $T_1, \ldots, T_4$ match all t_1, t_2, t_3. In general, TC-patterns can express different patterns from term trees.

In this paper, firstly we show that the TC-pattern matching problem is NP-complete unless the degree of every contractible vertex of t is bounded by a constant. Secondly we present an algorithm for computing the TC-pattern matching problem which runs in $O(nN^{\max\{d-1,1.5\}})$ time where d is a constant upper bound on the degree of a contractible vertex. Finally, we give a polynomial time algorithm for finding a minimally generalized TC-pattern explaining a given set of trees. Then we conclude that the class of tree languages obtained from TC-patterns is polynomial time inductively inferable from positive data.

2 Preliminaries

For a graph G, we denote by $V(G)$ the set of vertices of G and by $E(G)$ the set of edges. For $V' \subseteq V(G)$, we denote by $G[V']$ the subgraph of G induced by V'.

Definition 1. *Let G and H be connected graphs. An H-witness structure $\mathcal{W} = \{W(u) \mid u \in V(H)\}$ is a partition of $V(G)$ satisfying the following conditions.*

1. *For any $u \in V(H)$, $G[W(u)]$ is connected, and*
2. *for any two distinct vertices $u, u' \in V(H)$, $\{u, u'\} \in E(H)$ if and only if there is an edge $\{v, v'\} \in E(G)$ such that $v \in W(u)$ and $v' \in W(u')$*

We call each set $W(u) \in \mathcal{W}$ the *H-witness set* of u. When G has an H-witness structure, G can be transformed to H by contracting each of H-witness sets into

one vertex by edge contraction. A *graph contraction pattern* h (*GC-pattern*, for short) is a triplet $h = (V, E, U)$ where (V, E) is a connected graph and U is a subset of V. Let Σ be a finite alphabet. A GC-pattern h has a vertex labeling $\varphi_h : V \to \Sigma$. We call an element of U a *contractible vertex*. For a GC-pattern h, we denote by $V(h)$ the set of vertices of h, by $E(h)$ the set of edges of h, and by $U(h)$ the set of contractible vertices of h. Let h be a GC-pattern with vertex labeling φ_h and G a connected graph with vertex labeling φ_G. Let H be a graph $(V(h), E(h))$. We say that h *matches* G if there is an H-witness structure $\mathcal{W} = \{W(u) \mid u \in V(h)\}$ of G such that for all $v \in V(h) \setminus U(h)$, $W(v)$ contains exactly one vertex from $V(G)$, say u, with $\varphi_G(u) = \varphi_h(v)$. Such an H-witness structure of G is called an h-witness structure of G.

Definition 2. *We say that a GC-pattern t is a tree contraction pattern (TC-pattern) if $(V(t), E(t))$ is a tree. A TC-pattern t with root r_t matches a tree T with root r_T if there is a t-witness structure $\mathcal{W} = \{W(u) \mid u \in V(t)\}$ of T such that $r_T \in W(r_t)$.*

First of all, we show the NP-completeness on the following problem.

TC-PATTERN MATCHING
Instance: A TC-pattern t and a tree T.
Question: Does T match t?

Theorem 1. *TC-PATTERN MATCHING is NP-complete.*

Membership in NP is obvious. We transform EXACT COVER BY 3-SETS (X3C), which is a well-known NP-complete problem, to this problem. In the X3C problem we are given a set $X = \{a_1, \ldots, a_n\}$ with $n = 3q$ for a natural number q and a collection $\mathcal{C}$ of 3-element subsets $C_1, \ldots, C_m$ $(m \geq 1)$ of X. The question is: is there a subcollection $\mathcal{C}' \subseteq \mathcal{C}$ such that every element of X occurs in exactly one member of $\mathcal{C}'$. A formal proof is omitted.

3 A Polynomial Time Pattern Matching Algorithm

Let t be a TC-pattern or a tree with root r_t. For a vertex v of t, we denote the number of children of v of t by $ch_t(v)$. For a vertex v of t, $t[v]$ denotes the subtree induced by v and all the descendants of v.

In this section, we propose a pattern matching algorithm which decides whether or not a tree T matches a TC-pattern t. We assume that every vertex v of t has a different identifier, denoted by $I_t(v)$. Let $c_{v,1}, \ldots, c_{v,ch_t(v)}$ be the children of v and $Ch_t(v) = \{I_t(c_{v,1}), \ldots, I_t(c_{v,ch_t(v)})\}$. We give two collections of sets of identifiers of t, denoted by $\mathcal{I}_T(u)$ and $\mathcal{J}_T(u)$, to every vertex u of T.

Lemma 1. *Let t be a TC-pattern and T a tree.*

1. *For any $u \in V(T)$, $\{I_1, \ldots, I_m\} \in \mathcal{J}_T(u)$ if and only if there are m different proper descendants $u_1, \ldots, u_m$ of u such that for any i and j $(1 \leq i \neq j \leq m)$, u_i is neither an ancestor nor a descendant of u_j, and $\{I_i\} \in \mathcal{I}_T(u_i)$.*

Algorithm MATCHING;
Input: A TC-pattern t and a tree T.
Step 1: (Initialization) For all $u \in V(T)$, if u is a leaf then set $\mathcal{I}_T(u) = \{\{I(v)\} \mid v$ is a leaf of $t\}$ and $\mathcal{J}_T(u) = \emptyset$, otherwise set $\mathcal{I}_T(u) = \emptyset$ and $\mathcal{J}_T(u) = \emptyset$. Set $h = h_T - 1$ where h_T is the height of T..
Step 2: (Iteration) For all $u \in V(T)$ of height h, do Steps 2.1–2.4:
2.1 For all $v \in V(t) \setminus U(t)$, if COVER$(Ch_t(v), \mathcal{I}_T(c_{u,1}), \ldots, \mathcal{I}_T(c_{u,ch_T(u)})) = true$ then set $\mathcal{I}_T(u) = \mathcal{I}_T(u) \cup \{\{I_t(v)\}\}$.
2.2 For all $v \in U(t)$, if there is an index $j \in \{1, \ldots, ch_T(u)\}$ such that $\{I_t(v)\} \in \mathcal{I}_T(c_{u,j})$ then set $\mathcal{I}_T(u) = \mathcal{I}_T(u) \cup \{\{I_t(v)\}\}$.
2.3 For all $v \in U(t)$, if COVER$(Ch_t(v), \mathcal{I}_T(c_{u,1}) \cup \mathcal{J}_T(c_{u,1}), \ldots, \mathcal{I}_T(c_{u,ch_T(u)}) \cup \mathcal{J}_T(c_{u,ch_T(u)})) = true$ then set $\mathcal{I}_T(u) = \mathcal{I}_T(u) \cup \{\{I_t(v)\}\}$.
2.4 For all $v \in U(t)$ and all subsets $P \subsetneq Ch_t(v)$, if COVER$(P, \mathcal{I}_T(c_{u,1}) \cup \mathcal{J}_T(c_{u,1}), \ldots, \mathcal{I}_T(c_{u,ch_T(u)}) \cup \mathcal{J}_T(c_{u,ch_T(u)})) = true$ then set $\mathcal{J}_T(u) = \mathcal{J}_T(u) \cup \{P\}$.
Step 3: (Decision) If $h > 0$ then set $h = h - 1$ and goto Step 2. If $\{I_t(r_t)\} \in \mathcal{I}_T(r_T)$ then T matches t, otherwise T does not match t.

Procedure COVER$(X, \mathcal{C}_1, \ldots, \mathcal{C}_m)$;
input: X: a finite set, $\mathcal{C}_1, \ldots, \mathcal{C}_m$: collections of subsets of X;
begin
  if $\exists J \subseteq \{1, \ldots, m\}$ and for each $j \in J$, $\exists C_j \in \mathcal{C}_j$ s.t. $\bigcup_{j \in J} C_j = X$ **then**
    return *true* **else return** *false*
end;

Fig. 2. Algorithm MATCHING and procedure COVER which is called from MATCHING

2. *For any pair of $v \in V(t)$ and $u \in V(T)$, $\{I_t(v)\} \in \mathcal{I}_T(u)$ if and only if $T[v]$ matches $t[u]$.*

From Lemma 1, we get that $\{I_t(r_t)\} \in \mathcal{I}_T(r_T)$ if and only if T matches t. We have the following theorem.

Theorem 2. *We assume that the degree of every contractible vertex in TC-patterns is bounded by a constant d. Then, TC-PATTERN MATCHING for a given TC-pattern t and a given tree T is solved in $O(nN^{\max\{d-1,1.5\}})$ time, where $n = |V(t)|$ and $N = |V(T)|$.*

4 A Polynomial Time Algorithm for Minimal Language Problem for TC-patterns

For a TC-pattern t, we define the TC-pattern language of t as $L(t) = \{T \mid T$ matches $t\}$. For a finite set of trees S, a *minimally generalized TC-pattern* explaining S is defined as a TC-pattern t such that $S \subseteq L(t)$ and there exists no TC-pattern t' satisfying $S \subseteq L(t') \subsetneq L(t)$. For example, in Fig. 1, t_1, t_2, t_3 (resp.

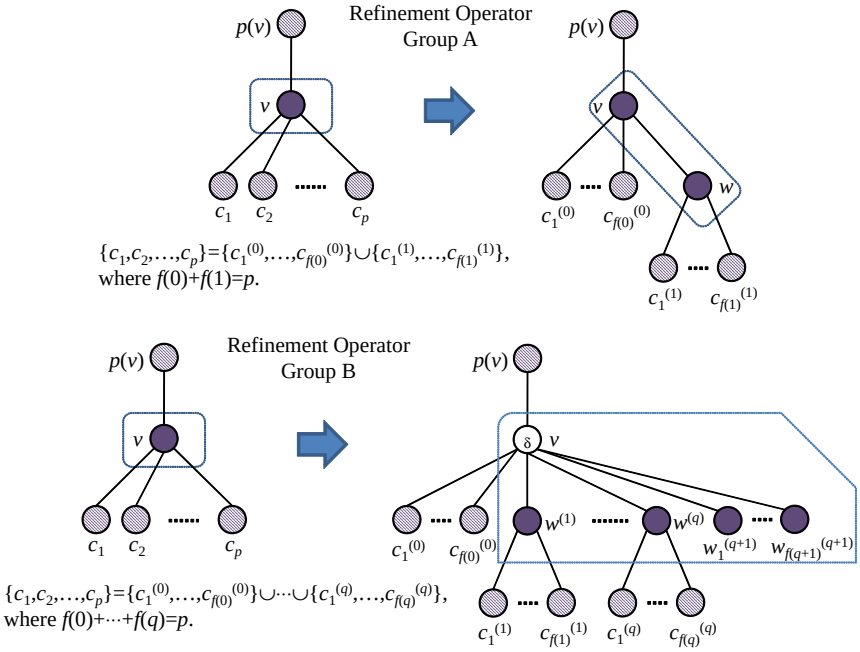

Fig. 3. Refinement operator groups A and B: Each group contains $O(2^d)$ and $O(D^d|\Sigma(S)|)$ refinement operators, respectively, where $D = \min_{T\in S} \max_{u\in V(T)} d(v)$ where $d(v)$ is the degree of v and $\Sigma(S) = \{\sigma \in \Sigma \mid \sigma \text{ appears in } S\}$.

$s_1, \ldots, s_5$) are all of the minimally generalized TC-patterns (resp. term trees) explaining S. We give a polynomial time algorithm for the following problem.

MINIMAL LANGUAGE PROBLEM FOR TC-PATTERNS
Instance: A finite set S of trees.
Problem: Find a minimally generalized TC-pattern explaining S.

We give the two refinement operators in Fig. 3, which are used to extend a TC-pattern as much as possible while $S \subseteq L(t)$ holds. We start with a TC-pattern t consisting of only one contractible vertex, and try to replace every contractible vertex v in t with one of the structures on the right hand sides of the arrows in Fig. 3 if it is possible. We omit a formal description of the algorithm. Instead, we describe in Fig. 4 an example of refinement processes.

Theorem 3. *We assume that there are infinitely many vertex labels in Σ, and that the degree of every contractible vertex in TC-patterns is bounded by a constant d. MINIMAL LANGUAGE PROBLEM FOR TC-PATTERNS for a given set of trees S is computed in $O(N_{\min}^{d+1} N_{\max}^{\max\{d-1,1.5\}}|S||\Sigma(S)|)$ time, where $N_{\min} = \min_{T\in S} |V(T)|$, $N_{\max} = \max_{T\in S} |V(T)|$, and $\Sigma(S) = \{\sigma \in \Sigma \mid \sigma$ appears in $S\}$.*

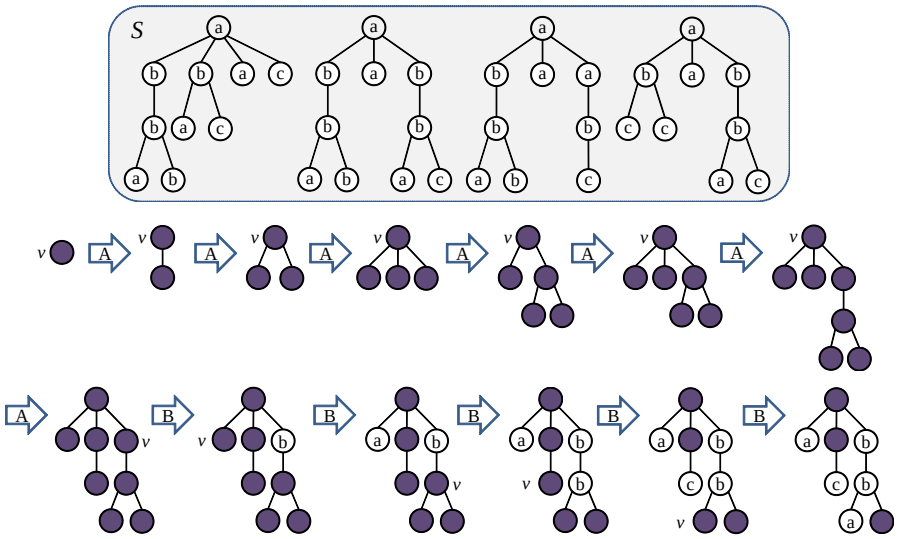

Fig. 4. A refinement process of our algorithm for finding a minimally generalized TC-pattern explaining S.

5 Conclusions

For Theorems 2 and 3, we can conclude that the class of TC-pattern languages $\mathcal{L} = \{L(t) \mid t \text{ is a TC-pattern}\}$ is polynomial time inductively inferable from positive data. Then, we have given an algorithmic foundation of discovering knowledge from tree structured data. We are now developing general data mining techniques for various real world data that can be modeled by unordered trees.

References

1. T. R. Amoth, P. Cull, and P. Tadepalli. On Exact learning of unordered tree patterns. *Machine Learning, 44(3)*, pp.211–243, 2001.
2. T. Miyahara, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Polynomial time matching algorithms for tree-like structured patterns in knowledge discovery. *Proc. PAKDD-2000, Springer, LNAI 1805*, pages 5–16, 2000.
3. S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 295–306, 1998.
4. T. Shoudai, T. Uchida, and T. Miyahara. Polynomial time algorithms for finding unordered tree patterns with internal variables. *Proc. FCT-2001, Springer, LNCS 2138*, pages 335–346, 2001.
5. Y. Suzuki, T. Shoudai, S. Matsumoto, T. Uchida, and T. Miyahara. Efficient Learning of Ordered and Unordered Tree Patterns with Contractible Variables. *Proc. ALT-2003, Springer, LNAI 2842*, pages.114–128, 2003.
6. K. Wang and H. Liu. Discovering structural association of semistructured data. *IEEE Trans. Knowledge and Data Engineering*, 12:353–371, 2000.