DOI ▪ TR 206, Department of Informatics, Kyushu Univeristy
ftp://ftp.i.kyushu ▪ u.ac.jp/pub/tr/trcs206.ps.gz
March 2002 (Submitting)

Optimized Substructure Discovery for Semi-structured Data

Kenji Abe[1], Shinji Kawasoe[1] Tatsuya Asai[1], Hiroki Arimura[1,2], and Setsuo Arikawa[1]

[1] Department of Informatics, Kyushu University
6–10–1 Hakozaki Higashi-ku, Fukuoka 812–8581, JAPAN
{k-abe,s-kawa,t-asai,arim,arikawa}@i.kyushu-u.ac.jp
[2] PRESTO, JST, JAPAN

Abstract. We address the problem of finding interesting substructures from a collection of semi-structured data such as XML or HTML. Our framework of data mining is optimized pattern discovery introduced by Fukuda *et al.*, where the goal of a mining algorithm is to discover a pattern that optimizes a given statistical measure such as the information entropy over a class of simple patterns. In this paper, modeling semi-structured data with labeled ordered trees, we study the efficient algorithm for the optimized pattern discovery problem for the class. In a previous paper, we developed the rightmost expansion technique and the incremental occurrence update technique by generalizing enumeration technique developed by Bayardo (SIGMOD'98) for discovering long itemsets to implement an efficient frequent pattern miner for the class of labeled ordered trees. By combining these technique with the pruning technique for optimized patterns of Morishita and Sese (PODS'00), we present an efficient algorithm for finding optimized patterns for labeled ordered trees of bounded size. Experimental results show that our algorithm perform well on a variety of size of data and range of parameters. We also show an approximation hardness result for labeled ordered trees of unbounded size.

1 Introduction

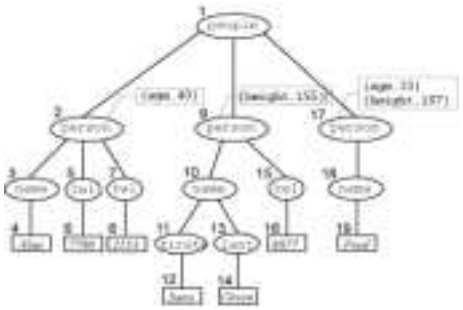
Recent progress of network and storage technologies have made it easier for an individual or an organization to collect, exchange, and accumulate massive amounts of electronic data through inter/intranet in the form of text streams, data sheets in PDF, HTML pages, and XML archives [18]. Such *semi-structured data* [1] are heterogeneous collections of weakly structured data that have no rigid structures, and thus traditional information retrieval and data mining methods do not work. Hence, there are potential demands for extracting unknown information from these semi-structured data [6, 11, 12, 16, 20].

Ordered Trees

In this paper, we study the problem of finding interesting substructure from a given collection of semi-structured data in HTML or XML format. In Fig. 1

```
<people>
   <person age="40">
      <name> Alan</name>
      <tel> 7786</tel>
      <tel> 2133</tel>
   </person>
   <person height="155">
      <name> <first> Sara</first>
           <last> Green</last>
      </name>
      <tel> 6877</tel>
   </person>
   <person age="33" height="187">
      <name> Fred</name>
   </person>
</people>
```

(a) An XML document



(b) The DOM tree for the left document

Fig. 1. XML data expressions

(a), we show an XML data as an example of semi-structured data, which are hierarchically structured texts with a set of tags as markups. We model such semi-structured data and the patterns for them both by *labeled ordered trees* as shown in Fig. 1 (b), where each node is labeled by symbols from an alphabet and the children of each node are ordered from left to right. We model XML attributes and text values with the nodes and tags in these trees.

The class of patterns we consider is that of labeled ordered trees of bounded size. Given labeled ordered trees called a pattern tree and a data tree, the pattern tree matches the data tree if there is a one-to-one mapping from the nodes of the pattern tree to the nodes of the data tree that preserves the direct ancestor relation, the possibly indirect sibling relation, and the labeling information.

Optimized pattern Discovery

We employed the *optimized pattern discovery* as our framework of semi-structured data mining, which has its origin in the statistical decision theory in 1970's [7] and rediscovered in data mining, machine learning and computational learning theory in the middle of 1990's [8, 10, 13–15].

Let $\mathcal{C}$ be a class of patterns to discover. Assume that we are given a collection S of *documents* and a *binary labeling function* ξ over documents in S that indicates if a document has a property of interest. A pattern $H \in \mathcal{C}$ splits the input collection into the disjoint sets S_1 and S_0 of matched documents and the unmatched documents. To measure the goodness of the split (S_1, S_0), we use a statistical measure $G_\xi(S_1, S_0) \in [0, 1]$ such as the classification error or the information entropy. Then, the goal of optimized pattern discovery is to find an optimized pattern $H \in \mathcal{C}$ that minimizes the statistical measure $G_\xi(S_1, S_0)$ over all patterns in the class $\mathcal{C}$.

Main Results

In a previous paper [3], we considered the frequent pattern discovery problem for the class of labeled ordered trees and presented an efficient algorithm FREQT. Previous algorithms for finding tree-like patterns basically adopted a straightforward generate-and-test strategy [12, 19]. In contrast, our algorithm FREQT is an incremental algorithm that simultaneously constructs the set of frequent patterns and their occurrences level by level. For the purpose, we devise an efficient enumeration technique for ordered trees by generalizing the itemset enumeration tree by Bayardo [4].

Based on the rightmost expansion and the incremental occurrence-update techniques of FREQT, in Section 3 we present an efficient mining algorithm OPTT that solves the optimized pattern discovery problem for labeled ordered trees. We also incorporate into OPTT a pruning technique with convexity measure by Sese and Morishita [14]. In the case that the maximum size of patterns is bounded but the number of labels are slowly growing in the total size N of input, we show that OPTT runs in $O(c^k N)$ time for some constant $c > 0$, and is more efficient than a straightforward algorithm with super linear time complexity in N if $k = O(1)$. Furthermore, we show that in the case that the maximum pattern size k is unbounded, the optimized patterns are hard to discover in contrast. Experimental results in Section 4 show that our algorithm scales well and efficiently finds optimized labeled ordered trees from a real datasets. In Section 5, we conclude.

Related Works

There are many studies on information and data retrieval from semi-structured databases [1, 18]. In contrast, not many researches have been done on semi-structured data mining [6, 11, 12, 19–21].

Wang and Liu [20] considered the problem of finding frequent tree-like patterns from a semi-structured data, and presented an Apriori-style algorithm for the problem. Since in their framework a pattern is represented with a collection of paths, the branching information of a tree is lost. Dehaspe *et al.* [6] presented the efficient algorithm for solving the frequent substructure discovery problem for labeled graphs, and applied it to the problem of function prediction of chemical compounds. Wang, Shapiro, Shasha *et al.* [19] devised the algorithm for discovering approximately common subtree, and applied it to motif discovery in genomics.

Matsuda and Motoda *et al.* [11] presented an efficient algorithm, called the graph-based induction, for discovering interesting patterns in directed graphs. Although they adopted a framework similar to the optimized pattern discovery, their interests are in developing an efficient heuristic search algorithm rather than exhaustive computation of optimized patterns. Inokuchi *et al.* [9] presented an Apriori-style algorithm for finding frequent subgraphs.

Miyahara *et al.* [12] consider discovery of labeled ordered trees in more general framework called tag-tree patterns, but their algorithm is a straightforward generate-test algorithm.

Independently to our previous work, Zaki [21] very recently proposed efficient algorithms for the frequent pattern discovery problem for ordered trees, which is essentially same to our rightmost expansion. Also, he reported that a depth-first search algorithm equipped with his enumeration technique performs very well.

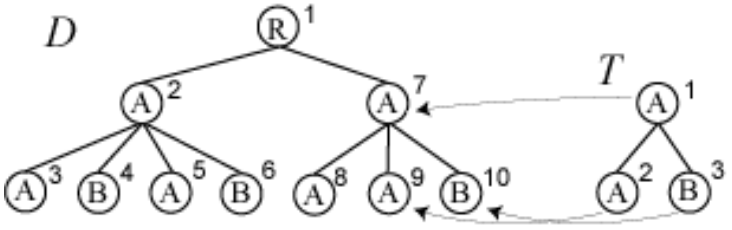


Fig. 2. A data tree D and a pattern tree T on the set $\mathcal{L} = \{A, B\}$ of labels

2 Preliminaries

2.1 Labeled Ordered Trees

Let $\mathcal{L} = \{\ell, \ell_0, \ell_1, \ldots\}$ be a finite alphabet of labels, which correspond to tags in XML and HTML. As a model of semi-structured databases and patterns such as XML [18] and OEM model [1], we adopt the class of labeled ordered trees defined as follows.

A *labeled ordered tree on* $\mathcal{L}$ (an ordered tree, for short) of size $k \geq 0$ [2] is a 6-tuple $T = (V, E, B, \mathcal{L}, L, v_0)$, where $V = \{1, \ldots, k\}$ $(n \geq 0)$, $E \subseteq V^2$ is the parent-child relation such that $(parent, child) \in E$, $B \subseteq V^2$ the (possibly indirect) sibling relation such that $(elder, younger) \in B$. $L : V \to \mathcal{L}$ is the *labeling function*, and $G = (V, E, v_0)$ forms a tree with root v_0. By convention, we assume that all nodes of T are numbered by the preorder traversal [2] of T. Consequently, The root v_0 is 1 and the rightmost leaf is k. In what follows, given an ordered tree $T = (V, E, B, \mathcal{L}, L, v_0)$, we refer to $V, E, B, L,$, respectively, as V_T, E_T, B_T and L_T if it is clear from context.

Given labeled ordered trees T, D, called the *pattern tree* and the *data tree*, resp., a function $\varphi : V_T \to V_D$ from nodes of T to nodes of D is a *matching function of* T *into* D if it satisfies the following conditions for any $v, v_1, v_2 \in V_T$:

- φ is a one-to-one mapping .
- φ preserves the parent relation, i.e., $(v_1, v_2) \in E_T$ iff $(\varphi(v_1), \varphi(v_2)) \in E_D$.
- φ preserves the sibling relation, i.e., $(v_1, v_2) \in B_T$ iff $(\varphi(v_1), \varphi(v_2)) \in B_D$.
- φ preserves the labels, i.e., $L_T(v) = L_D(\varphi(v))$.

A pattern tree T *matches* a data tree D (or T *occurs in* D) if there exists some matching function φ of T into D. Then, we define the *root occurrence of* T *in* D *w.r.t.* φ to be the node $Root(\varphi) = \varphi(1) \in V_D$ of D that the root of T maps, and the set of the root-occurrences of T in D by $Occ(T) = \{Root(\varphi) \mid \varphi \text{ is a matching function of } T \text{ into } D\}$.

For every $k \geq 0$, we denote by $\mathcal{T}_k$ the class of labeled ordered trees of size exactly k over $\mathcal{L}$, and by $\mathcal{T} = \cup_{k \geq 0} \mathcal{T}_k$ the whole class. Let $\mathcal{T}^k = \cup_{i \leq k} \mathcal{T}_i$. We assume that $\mathcal{T}_0$ contains the empty tree $\perp$ of size zero and $\perp$ matches to any tree at any node.

Example 1. In Fig. 2, we show examples of labeled ordered trees, say D and T, on the alphabet $\mathcal{L} = \{A, B\}$, where a circle with the number, say v, at its upper right corner indicates the node v, and the symbol appearing in a circle indicates its label $L(v)$. We also see that the nodes of these trees are numbered consecutively by the preorder. In this figure, a matching function, say φ_1, of the pattern T with three nodes into the data tree D with ten nodes is indicated by

a set of arrows from the nodes of T. The root-occurrences corresponding to φ_1 is $Root(\varphi) = 7$. Furthermore, there are two root-occurrences of T in D, namely 2 and 7, while there are five matching functions of T into D.

The *length* of a path of T is defined by the number of its nodes. For every $p \geq 0$ and a node v, the *p-th parent* of v, denoted by $\pi_T^p(v)$, is the unique ancestor u of v such that the length of the path from u to v has length exactly $p+1$. By definition, $\pi_T^0(v)$ is v itself and $\pi_T^1(v)$ is the parent of v. The *depth* of a node v of T, denoted by $depth(v)$, is defined by the length d of the path $x_0 = v_0, x_1, \ldots, x_{d-1} = v$ from the root v_0 of T to the node v.

In this paper, we deal with the full class of XML data and Dom trees. Thus, we transform a set of XML attribute-value pairs at a node into a set of two-node trees ordered in the lexicographic order of their labels [3], and the text value of a text node into a new child node of the node labeled with the text value. Note that this results that even for HTML/XML documents over fixed set of markup tags, the number of distinct labels may not be constant.

2.2 Optimized Pattern Discovery

We introduce the optimized pattern discovery according to [7, 13]. Suppose that we are given a set $D = \{D_1, \ldots, D_m\}$ of documents and an objective condition $\xi : S \to \{0, 1\}$, where each document D_i is a labeled tree in our problem. The value $\xi(D_i)$ indicates if a document D_i is interesting.

For every $\alpha \in \{0, 1\}$, let $N_\alpha = \sum_{D_i \in D} [\, \xi(D_i) = \alpha \,]$ and $M_\alpha = \sum_{D_i \in D} [\, \xi(D_i) = \alpha \ \wedge \ H \text{ matches } D_i \,]$. That is, N_1 and N_0 are the numbers of positive and negative documents, and M_1 and M_0 are the numbers of matched and unmatched positive documents. Then, a pattern T on a labeled sample (D, ξ) defines a *contingency table* (M_1, M_0, N_1, N_0).

An *impurity function* is any real-valued function $\psi : [0, 1] \to \mathbf{R}$ such that (i) it takes the maximum value $\psi(\frac{1}{2})$ at $1/2$, (ii) the minimum value $\psi(0) = \psi(1) = 0$ at 0 and 1, and (iii) ψ is convex, i.e., $\psi((x+y)/2 \geq (\psi(x) + \psi(y))/2$ for every $x, y \in [0, 1]$. The followings are examples of impurity functions:

- The prediction error: $\psi_1(x) = \min(x, 1-x)$ [7, 10].
- The information entropy: $\psi_2(x) = -x \log x - (1-x) \log(1-x)$ [13].
- The Gini index: $\psi_3(x) = 2x(1-x)$ [7].

Then, the object function of our optimized pattern discovery based on ψ, for every pattern T, D and ξ,

$$Obj_{S,\xi}^{\psi}(H) = N_1 \cdot \psi(M_1/N_1) + N_0 \cdot \psi(M_0/N_0),$$

where (M_1, M_0, N_1, N_0) is a contingency table defined by the pattern H over S and ξ. A pattern $\hat{H}$ is *ψ-optimal* within class $\mathcal{C}$ (or *optimal*, for short if $Obj_{S,\xi}^{\psi}(\hat{H}) = \min_{H \in \mathcal{C}} Obj_{S,\xi}^{\psi}(H)$.

Let $\mathcal{C}$ be the class of candidate patterns and ψ be any impurity function. Now, we state our data mining problem, called the *opitimal pattern discovery problem for labeled ordered trees* as follows.

ψ-OPTIMIZED PATTERN DISCOVERY PROBLEM($\mathcal{C}$)
Given: A set $D = \{D_1, \ldots, D_m\}$ of documents and an objective condition $\xi : D \to \{0, 1\}$.

Algorithm OPTT

Input: A label alphabet $\mathcal{L}$, an integer $k \geq 0$, a set D of labeled ordered trees over $\mathcal{L}$ and an objective condition $\xi : D \to \{0, 1\}$, and an impurity function $\psi : [0, 1] \to [0, 1]$.

Output: A ψ-optimal pattern T of size at most k within $\mathcal{T}_k$ on S and ξ.

Variable: A queue or stack $BD \subseteq \mathcal{T}$ of patterns, called *boundary set*, and a priority queue $R \subseteq \mathcal{T} \times \mathbf{R}$ of patterns with real weight.

1. $BD := \langle \bot, RMO(\bot) \rangle)$, where $RMO(\bot)$ is the preorder traversal of D.
2. While $BD \neq \emptyset$, do:
   (a) $\langle S, RMO(S) \rangle := Pop(BD)$;
   (b) Compute $eval := Obj^{\psi}_{D,\xi}(S, RMO(S))$;
   Insert $\langle S, eval \rangle$ into R with $eval$ as the key;
   (c) If $Lookahead(S, RMO(S)) > Opt(R)$ then
   – Skip Step 2 (d) and go to the beginning of Step 2;
   (d) For each $\langle T, RMO(T) \rangle \in$ Expand-A-Tree$(S, RMO(S))$, do:
   – $Push(\langle T, RMO(T) \rangle, BD)$;
3. Return an optimal pattern $\langle P, eval \rangle := DeleteMin(R)$.

Fig. 3. An efficient algorithm for discovering the optimal pattern of bounded size, where search strategy is either breadth-first or depth-first depending on the choice of the boundary set BD

Problem: Find an optimal pattern $H \in \mathcal{C}$ that minimizes the value $Obj^{\psi}_{S,\xi}(T)$ over all patterns in $\mathcal{C}$.

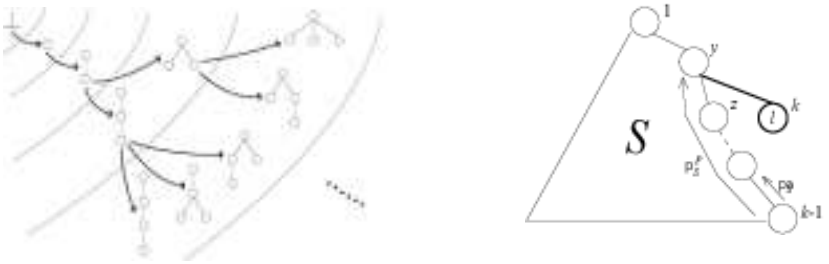
Particularly, we consider the optimal pattern discovery problem for the class of labeled ordered trees. From recent development in learning theory, it is known that any algorithm that efficiently solves the classification error minimization problem can best approximate arbitrary unknown probability distributions possibly with classification noise within a given hypothesis space [10].

3 Mining Algorithms

In this section, we present efficient algorithms for solving the optimal pattern discovery problem for ordered trees. Our algorithm employs the following observations: (i) Based on the rightmost expansion technique of [3, 21], one can generate all labeled ordered trees without duplicates by attaching a new rightmost leaf one by one; (ii) By taking the rightmost leaf of a pattern as the reference point [3], one can compactly represents the occurrence information of a pattern by maintaining the list of all nodes where the rightmost leaf of the pattern maps. This is enough to incrementally computing the rightmost occurrences through the expansion process of (i); (iii) Using the technique by Morishita and Sese [14] based on the convexity of the statistical measure $Obj^{\psi}_{S,\xi}(\cdot)$, one can efficiently prune unpromising branch in a search process.

In Fig. 3, we present our algorithms OPTT for discovering an optimal pattern that minimizes a given statistical measure within the class of labeled ordered trees of bounded size k on a data tree D. The algorithm uses either a *queue* (FIFO list) or a *stack* (FILO list) to implement the boundary set DB consisting of candidate patterns to be expanded. The search strategy is the *breadth-first search* or *levelwise search* if BD is a queue and the *depth-first search* if BD is a stack.

Let $k \geq 0$ be the maximum size of patterns, D be a set of data trees and ξ be an objective condition on D. Starting from the empty pattern $\bot$ of size



(a) A search graph for ordered trees

(b) The (p, ℓ)-expansion of tree S

Fig. 4. The rightmost expansion for ordered trees

zero, OPTT-B searches the hypothesis space $\mathcal{T}^k$ of labeled ordered trees of size at most k with growing candidate pattern trees by attaching a new node one by one using the subprocedure Expand-A-Tree. Whenever a new pattern tree T is generated from its predecessor S, its occurrence list $RMO(T)$ in the data tree D is incrementally computed from $RMO(S)$ of its predecessor S in Expand-A-Tree. In the rest of this section, we will describe the details of the algorithms.

3.1 Efficient Enumeration of Ordered Trees

To implement the search of optimal pattern efficiently, the search should enumerate all pattern trees in the hypothesis space $\mathcal{T}^k$ without duplicates. A possible way to do this is to design an acyclic binary relation $\rightarrow$ on $\mathcal{T}^k$, called an expansion relation, and for each candidate pattern $P \in BD$, to compute the set of all immediate successor of P w.r.t. $\rightarrow$. If the obtained graph $(\mathcal{T}^k, \rightarrow)$, called an *enumeration graph*, forms a rooted tree, then this strategy combined with breadth-first or depth-first search yields non-duplicate enumeration of $\mathcal{T}^k$. The basic idea will be illustrated in Fig. 4(a).

To do this, we expand a given pattern S by attaching a new leaf, but with the resriction that the new leaf should be attached on the rightmost branch and the rightmost child. Let $\mathcal{L}$ be the label alphabet, $S \in \mathcal{T}$ be any pattern tree of size $k-1 \geq 0$ and $rml(S) = k-1$ be the rightmost leaf of S. For every $0 \leq pdepth(rml(S))$ and every label $\ell \in \mathcal{L}$, the (p, ℓ)-*expansion* of S is the labeled ordered tree T obtained from S by attaching a new node, namely k, to the node $y = \pi_S{}^p(x)$ with label ℓ as the rightmost child of y. We define a *rightmost expansion* of pattern S or to be the (p, ℓ)-expansion T of S for any $p \geq 0$ and $\ell \in \mathcal{L}$, and write $S \rightarrow_{\mathrm{r}} T$.

By the following theorem, we can enumerate members of $\mathcal{T}$ without duplicates using an appropriate tree traversal method.

Theorem 1 (Asai *et al.*, [3]). *For every nonnegative integer k, the enumeration graph $(\mathcal{T}^k, \rightarrow_{\mathrm{r}})$ forms a tree with the root $\perp$.*

3.2 Updating Occurrence Lists

The second key of our algorithm is how to efficiently store the information of a matching φ of each pattern T into the data tree D. Instead of recording the full information $\langle \varphi(1), \ldots, \varphi(k) \rangle$ of φ, our algorithm maintains only the rightmost occurrences $Rmo(\varphi) = \varphi(k)$ as the partial information on φ, that is, $Rmo(\varphi)$ is the node of D that the rightmost leaf k of T maps. We define

Algorithm Expand-A-Tree$(S, RMO(S))$
Set $Succ := \emptyset$; For each pairs $(p, \ell) \in \{0, \ldots, d-1\} \times \mathcal{L}$, where $d = depth(rml(S))$ is the depth of the rightmost leaf of S, do the followings:
- Compute the (p, ℓ)-expansion T of S;
- $RMO(T) :=$ Update-RMO$(RMO(S), p, \ell)$; /* See Fig. 6 */
- $Succ := Succ \cup \{T, RMO(T)\}$;

Return $Succ$;

Fig. 5. The algorithm for computing all successors of a pattern

Algorithm Update-RMO(RMO, p, ℓ)
1. Set $RMO_{\rm new}$ to be the empty list ε and $check := null$.
2. For each element $x \in RMO$, do:
   (a) If $p = 0$, let y be the leftmost child of x.
   (b) Otherwise, $p \geq 1$. Then, do:
      - If $check = {\pi_D}^p(x)$ then skip x and go to the beginning of Step 2 (Duplicate-Detection).
      - Else, let y be the next sibling of ${\pi_D}^{p-1}(x)$ (the $(p-1)$st parent of x in D) and set $check := {\pi_D}^p(x)$.
   (c) While $y \neq null$, do the following:
      - If $L_D(y) \leq_{\mathcal{L}} \ell$, then $RMO_{\rm new} := RMO_{\rm new} \cdot (y)$; /* Append */
      - $y := next(y)$; /* the next sibling */
3. Return $RMO_{\rm new}$.

Fig. 6. The incremental algorithm for updating the rightmost occurrence list of the (p, ℓ)-expansion of a given pattern T from that of T

$RMO(T) = \{Rmo(\varphi) \mid \varphi$ is a matching function of T into $D\}$ to be the set of the rightmost occurrences of T.

Example 2. In Fig. 2, the pattern tree T has three rightmost occurrences $4, 6$ and 10 in the data tree D. Then, the root-occurrences 2 and 7 of T can be easily computed by taking the parents of $4, 6$ and 10 in D.

Fig. 6 shows the algorithm Update-RMO that, given the (p, ℓ)-expansion T of a pattern S and the list of the rightmost occurrence $RMO(S)$ of S, computes the list of the rightmost occurrences $RMO(T)$ without duplicates. This algorithm is base on the following observation: For every node y, y is in $RMO(T)$ iff there is a node x in $RMO(S)$ such that y is the strict younger sibling of the $(p-1)$-th parent of x. Although a straightforward implementation of this idea still results duplications, the *Duplicate-Detection* technique [3] ensures the uniqueness of the rightmost occurrences computed by the algorithm (See [3], for detail).

Lemma 1 (Asai *et al.* [3]). *For a pattern S, the algorithm* Update-RMO *exactly computes all the elements in $RMO(T)$ from $RMO(S)$ without duplicates, where T is a rightmost expansion of S.*

3.3 Pruning by Convexity

For the optimal pattern discovery, Morishita *et al.* [14] presented an efficient pruning technique as follows.

Let (D, ξ) be an input collection. For every pattern T, the value of the objective function $Obj^{\psi}_{S,\xi}(T)$ is determined by the contingency table $(M_1^T, M_0^T, N_1^T, N_0^T)$

defined by T on (D, ξ). If we fixed the instance (D, ξ), $Obj^{\psi}_{S,\xi}(T)$ can be regarded as a function of a point (M_1^T, M_0^T) in 2-dimensional plane $[0,1]^2$. Hence, we can write $Obj^{\psi}_{S,\xi}(M_1^T, M_0^T)$ for $Obj^{\psi}_{S,\xi}(T)$. Then, we say that $Obj^{\psi}_{S,\xi}(T)$ is *convex* if $Obj^{\psi}_{S,\xi}(\cdot,\cdot)$ is convex in the usual sense on $[0,1]^2$. We note that if a pattern tree T is a direct or indirect successor of S, i.e., $S \stackrel{+}{\to} T$, then the point (M_0^T, M_1^T) for T is interior of the point (M_0^S, M_1^S) for S. Then, Morishita and Sese [14] showed the following theorem.

Theorem 2 (Morishita and Sese [14]). *For pattern trees S, T, if $S \stackrel{+}{\to} T$ and the contingency table for T is (M_1, M_0, N_1, N_0) then*

$$\min\{Obj^{\psi}_{S,\xi}(0, M_0), Obj^{\psi}_{S,\xi}(M_1, 0)\} \leq Obj^{\psi}_{S,\xi}(M_1, M_0). \tag{1}$$

In the algorithm OPTT of Fig. 3, we used the pruning technique based on the above theorem in Step 2 (c), where the subprocedure Lookahead computes the value of the left hand side of Eq. 1. From [13, 14], the objective functions based on the following impurity functions ψ have the convexity: the classification error, the information entropy, Gini index function, and Chi^2 index function.

3.4 Theoretical Analysis: The Case of Bounded Pattern Size

Now, we give a case analysis on the performance of our algorithm in the case that maximum pattern size is bounded by a constant $k \geq 0$. Let $b \geq 0$ be the branching factor of input data trees. Suppose that we have a growing series $(D_i)_{i \in \mathbf{N}}$ of data trees. For every i, let N_i be the size of the data tree D_i and L_i be the number of distinct labels appearing in D_i. Since we encode attribute names, attribute values, and the text contents as the node labels, in practice L_i is not a constant and typically bounded by slowly growing function $L_i = f(N_i)$, e.g., $f(N) = N^{\alpha}$ for $0 < \alpha < 1$.

Now, we will estimate the time complexity of a straightforward enumeration-and-test algorithm. Let Z_i be the number of distinct labeled ordered trees of size k and with at most L_i labels. Since $Z_i = \Omega(2^{ck} \cdot {L_i}^k)$ for some $c > 0$, such an algorithm requires $\Theta(2^{ck} N^{k\alpha} \cdot N)$ time, which is not linear even when $k = O(1)$.

In contrast, the following theorem says that under the above assumptions, the running time of the breadth-first/levelwise version of the OPTT algorithm is linear in the total size N of $D \subseteq \mathcal{T}$ when k and b are constants. The same upperbound holds for the depth-first version, too. This method also gives a more precise estimation of the running time of the algorithm FREQT in [3].

Theorem 3. *Under the above assumptions, the running time of* OPTT *on the input collection D is bounded by $O(k^{k+1} b^k N)$, where N is the total size of D.*

Proof. We will estimate the upper bound of the sum R_k of the length the rightmost occurrences of the patterns generated in the k-th stage. By an argument in [3], it follows that the running time of OPTT is bounded above by the sum $\sum_{l=0}^{k-1} l R_l$. For every $0 \leq p \leq k$, if we define $R_{k,p}$ to be the sum of the rightmost occurrences of those trees in $\mathcal{T}_k$ generated in the k-th stage by (p, ℓ)-expansion for some ℓ then we have $R_k = \sum_{p=0}^{k-1} R_{k,p}$. Trivially, $R_0 = R_{0,0} \leq N_i$. Inductively, we can show that $R_{k,p} \leq b R_{k-1}$ for every $0 \leq p, q \leq k$. Then, we can show that $R_k \leq \sum_{p=0}^{k-1} R_{k,p} \leq kb R_{k-1}$. Solving this recurrence, we have $R_k = O(k^{k-1} b^{k-1} N_i)$, and thus the result immediately follows. □

3.5 Theoretical Analysis: The Case of Unbounded Pattern Size

The maximum agreement problem (MA, for short) is a dual problem of the classification error minimization problem and defined as follows: Given a pair (D, ξ), find a pattern T that maximizes the *agreement* of T, i.e., the ratio of documents in S that is correctly classified by T.

Recently, Ben-David *et al.* [5] showed that for any $\varepsilon > 0$, there is no polynomial time $(770/767 - \varepsilon)$-approximation algorithm for the maximum agreement problem for Boolean conjunctions if $P \neq NP$. When we can use arbitrary many labels, we can easily show the following theorem by using the approximation factor preserving reduction [17]. The proof is easy, but it indicates that the restriction of bounded pattern size is really necessary for efficient optimized pattern discovery. (The proof is attached in the appendix.)

Theorem 4. *For any $\varepsilon > 0$, there exists no polynomial time $(770/767 - \varepsilon)$-approximation algorithm for the maximum agreement problem for labeled ordered trees of unbounded size on an unbounded label alphabet if $P \neq NP$. This is true even when either the maximum depth of trees is at most three or the maximum branching of trees is at most two.*

4 Experimental Results

We have experimented with the algorithms on the following two data sets. The data sets are *Citeseers* (5.6MB) and *Imdb*, where HTML/XML attributes and text values are encoded by the tags and nodes in a labeled ordered tree. The data set Citeseers is a collection of CGI generated HTML pages collected from an online bibliographic archive Citeseers[3], whose data tree has 196,247 nodes and 7,125 unique tags. The data set Imdb is a collection of XML data generated with a hand-coded Perl script from the HTML pages collected from an online movie database, called the Internet Movie Database (IMDb) [4].

In the experiments, the tree mining algorithms we tested are the followings: The breadth-first/levelwise search version (OPTT+BF) and the depth-first search version (OPTT+DF) of the optimized tree miner OPTT of this paper (Fig. 3); The frequent tree miner FREQT of [3] with breadth-first/levelwise search (OPTT+DF). We also implemented pruning with convexity (denoted by C). All algorithms were implemented in Java (SUN JDK1.3.1 JIT) using a DOM library (OpenXML). Experiments were run on PC (Pentium III 600MHz) with 512 megabytes of main memory running Linux 2.2.14 or Windows 2000.

Scalability and Running Time

Fig. 7 shows the running time against the size of the data tree when the maximum size of the pattern trees is fixed to $k = 5$. The data set is Citeseers, and the size of the input collection ranges from 5(pages)/316(KB)/22847(nodes) to 180(pages)/5.61(MB)/402740(nodes). Then, the running time seems to linearly scale on this data set for fixed k. It fits to the theoretical bound in Section 3.4.

Fig. 8 shows the running time with varying the maximum pattern tree size k from 1 to 11. The data set is a fragment of Imdb, and the size of the input collection is fixed to 18(documents)/39.4(KB)/5835(nodes) containing 759 unique labels. Since the y-axis is log-scaled, the almost linear plot confirms the

[3] `http://citeseer.nj.nec.com/`

[4] `http://www.imdb.com/`

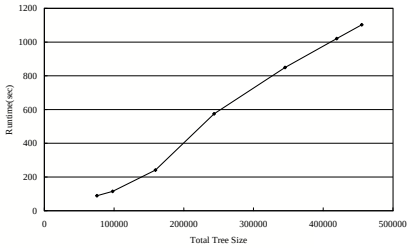


Fig. 7. The scalability: The running time with varying the input data size.

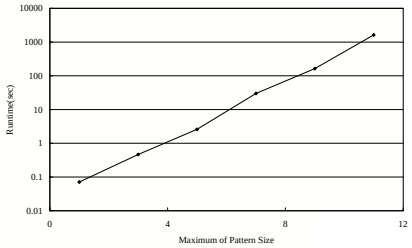


Fig. 8. The running time with varying the maximum pattern size.

Table 1. Comparison of tree mining algorithms in running time and space.

Algorithm	OPTT+DF	OPTT+DF+C	OPTT+BF	OPTT+BF+C	FREQT(0.1%)+BF
Time	29.7 (sec)	21.5 (sec)	20.2 (sec)	20.0 (sec)	10.4 (sec)
Space	8.0 (MB)	8.0 (MB)	96.4 (MB)	96.4 (MB)	20.7 (MB)

theoretical analysis of Section 3.5; when the data size is fixed, the running time is exponential in the maximum pattern size k.

Search Strategies and Pruning Techniques

Table. 1 shows the running time of versions of optimized tree miners OPTT+DF, OPTT+DF+C, OPTT+BF, OPTT+BF+C, and a frequent tree miner FREQTon the same data set to that of Fig. 8. This experiment shows that on this small data set, the difference in the running time between the depth-first search (denoted by DF) and the breadth-first search (denoted by BF) in Section 3.1 is not significant, while DF saves the main memory size more than ten times than BF. From this, the depth-first search strategy is an attractive choice in situations that only a limited amount of main memory is available. Also, the use of pruning with convexity (denoted by C) in Section 3.3 is effective in the depth-first search; OPTT+DF+C is 1.5 times faster than OPTT+DF.

5 Conclusion

In this paper, we studied the optimized pattern discovery problem for the class of labeled ordered trees by modeling semi-structured data as labeled ordered trees. We presented an efficient mining algorithm that finds the labeled ordered trees that optimize a given statistical objective function on a binary labeled collection of labeled ordered trees. Experimental results confirmed that the proposed algorithm is scalable for mining trees of bounded size. We also compared the breadth-first and the depth-first search strategies and the use of pruning technique with convexity.

Acknowledgments

The authors would like to thank Thomas Zeugmann, John Case, Akihiro Yamamoto, Masayuki Takeda, Ayumi Shinohara, and Shinichi Shimozono, Daisuke Ikeda and Akira Ishino for fruitful discussion on this issue. The fourth author also

would like to thank Shinichi Morishita, Masaru Kitsuregawa, Takeshi Tokuyama, and Yoshito Toyama for their valuable comments and discussions.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web*. Morgan Kaufmann, 2000.
2. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.
3. T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *Proc. the 2nd SIAM Int'l Conf. on Data Mining (SDM2002)*, 2002 (To appear).
4. R. J. Bayardo Jr. Efficiently mining long patterns from databases. In *Proc. SIGMOD98*, 85–93, 1998.
5. S. Ben-David, N. Eiron, and P. M. Long, On the difficulty of Approximately Maximizing Agreements, In *Proc. COLT 2000*, 266–274, 2000.
6. L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In *Proc. KDD-98*, 30–36, 1998.
7. L. Devroye, L. Gyorfi, G. Lugosi, *A Probablistic Theory of Pattern Recognition*, Springer-Verlag,1996.
8. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules. In *Proc. SIGMOD'96*, 13–23, 1996.
9. A. Inokuchi, T. Washio and H. Motoda An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data, Proc. PKDD 2000, 13–23, 2000.
10. M. J. Kearns, R. E. Shapire, L. M. Sellie, Toward efficient agnostic learning. *Machine Learning*, 17(2–3), 115–141, 1994.
11. T. Matsuda, T. Horiuchi, H. Motoda, T. Washio, K. Kumazawa, and N. Arai. Graph-based induction for general graph structured data. In *Proc. DS'99*, 340–342, 1999.
12. T. Miyahara, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of frequent tree structured patterns in semistructured web documents. In *Proc. PAKDD-2001*, 47–52, 2001.
13. S. Morishita, On classification and regression, In *Proc. Discovery Science '98*, LNAI 1532, 49–59, 1998.
14. S. Morishita and J. Sese, Traversing Itemset Lattices with Statistical Metric Pruning, In *Proc. PODS'00*, 226–236, 2000.
15. R. Rastogi, K. Shim, Mining Optimized Association Rules with Categorical and Numeric Attributes, Proc. ICDE'98, 503–512, 1998.
16. K. Taniguchi, H. Sakamoto, H. Arimura, S. Shimozono, and S. Arikawa. Mining semi-structured data by path expressions. In *Proc. DS2001*, 387–388, 2001.
17. V. V. Vazirani, Approximation Algorithms, Springer, Berlin, 1998.
18. W3C Recommendation. *Extensibe Markup Language (XML) 1.0*, second edition, 06 October 2000. `http://www.w3.org/TR/REC-xml`.
19. J. T. L. Wang, B. A. Shapiro, D. Shasha, K. Zhang, and C.-Y. Chang. Automated discovery of active motifs in multiple rna seconary structures. In *Proc. KDD-96*, 70–75, 1996.
20. K. Wang and H. Q. Liu. Discovering structual association of semistructured data. *IEEE Trans. Knowledge and Data Engineering (TKDE2000)*, 12(3):353–371, 2000.
21. M. J. Zaki. Efficiently mining frequent trees in a forest. Computer Science Department, Rensselaer Ploytechnic Institute, *PRI-TR01-7-2001*, 2001. `http://www.cs.rpi.edu/~zaki/PS/TR01-7.ps.gz`

A Appendix: Proof of Theorem 4

Note: This is not a part of the submitted paper.

Theorem 4 For any $\varepsilon > 0$, there exists no polynomial time $(770/767 - \varepsilon)$-approximation algorithm for the maximum agreement problem for labeled ordered trees of unbounded size on an unbounded label alphabet if $P \neq NP$. This is true even when either the maximum depth of trees is at most three or the maximum branching of trees is at most two.

Proof. Ben-David, Eiron, and Long [5] showed that for any $\varepsilon > 0$, there exists no polynomial time $(770/767 - \varepsilon)$-approximation algorithm for the maximum agreement problem for Boolean conjunctions if $P \neq NP$. Thus, we give an approximation factor preserving reduction (f, g) (See, e.g., [17] for definition) from the optimized pattern discovery problem (MA, for short) for the class of Boolean conjunctions to MA for labeled ordered trees as follows. We use *term notation* to represent labeled trees. Let $\{x_1, \ldots, x_n\}$ be the set of Boolean variables and $I_1 = (S, \xi)$ be an instance of MA for conjunctions. Let $\mathcal{L} = \{C\} \cup \{\, \mathbf{1}_i, \mathbf{0}_i \mid i = 1, \ldots, n \,\}$. Then, mapping f transforms I_1 into the instance $I_2 = (S', \xi')$ such that (i) for each $a = (a_1, \ldots, a_n) \in S$, $f(a) = C(\mu_1(a_1), \ldots, \mu_n(a_n))$, $S' = f(S)$, and $\xi'(f(a)) = \xi(a)$, where $\mu_i(1) = \mathbf{1}_i$ and $\mu_i(0) = \mathbf{0}_i$. Conversely, the mapping g transforms any solution $S' \in Sol_2(f(I_1))$ into some solution $S = g(S) \in Sol_1(I_1)$ defined as follows. For every labeled tree T over $\mathcal{L}$, $g(T) = L_1 \wedge \cdot \wedge L_n$ is the conjunction defined as follows: For every $i = 1, \ldots, n$, L_i is x_i if T contains $\mathbf{1}_i$, $\overline{x_i}$ if T contains $\mathbf{0}_i$, and $\top$ otherwise. Let $I_1 = (S, \xi)$ and $I_2 = f(I_1) = (f(S), f(\xi))$. Then, we can show that for every assignment $a \in \{0,1\}^n$ and every tree T, $g(T)(a) = 1$ iff T matches $f(a)$. Thus, it follows that $\forall T . \exists F = g(T)\ [Err_{S,\xi}(F) = Err_{f(S),f(\xi)}(T)]$.

Conversely, we can transform any conjunction $F = L_1 \wedge \cdot \wedge L_n$ into a tree $h(F) = C(T_1, \ldots, T_n)$, where T_i is $\mathbf{1}_i$ if L_i is positive, $\mathbf{0}_i$ if L_i is negative, and the empty tree if $L_i = \top$. Then, we have $\forall F . \exists T = h(F)\ [Err_{S,\xi}(F) = Err_{f(S),f(\xi)}(T)]$.

Combining above arguments, we have $\forall I_1 . \exists I_2 = f(I_1)\ [OPT(I_1) \geq OPT(I_2)]$ and $\forall T \in Sol(I_2) . \exists F = g(T) \in Sol(I_1)\ [Obj(I_1, F) \leq Obj(I_2, T)]$. Therefore, (f, g) is an approximation factor preserving reduction from MA for conjunction to MA for labeled ordered trees. Thus, it immediately follows from the result of Ben-David, Eiron, and Long [5] that for any $\varepsilon > 0$, there is no polynomial time $(770/767 - \varepsilon)$-approximation algorithm for the maximum agreement problem for labeled ordered trees if $P \neq NP$. This proves the theorem. □