# Mining Maximal Frequently Changing Subtree Patterns from XML Documents

Ling Chen, Sourav S. Bhowmick and Liang-Tien Chia

School of Computer Engineering,
Nanyang Technological University, Singapore, 639798, SINGAPORE

**Abstract.** Due to the dynamic nature of online information, XML documents typically evolve over time. The change of the data values or structures of an XML document may exhibit some particular patterns. In this paper, we focus on the sequence of changes to the structures of an XML document to find out which subtrees in the XML structure frequently change together, which we call Frequently Changing Subtree Patterns (*FCSP*). In order to keep the discovered patterns more concise, we further define the problem of mining *maximal FCSPs*. An algorithm derived from the FP-growth is developed to mine the set of *maximal FCSP*s. Experiment results show that our algorithm is substantially faster than the naive algorithm and it scales well with respect to the size of the XML structure.

## 1 Introduction

With the fast proliferation of available XML documents, data mining community has been motivated to discover knowledge from XML repositories. For example, recently, there has been increasing research effort in mining association rules from XML documents[2]; classifying XML data [8] and clustering XML[9]. One of the common features shared by existing work on XML mining is that they usually mine from a collection of XML documents at a certain time point.

Due to the dynamic nature of online information, XML documents typically evolves over time. Consequently, issues related to detecting changes to XML documents received considerable attention recently[13][5]. Detected changes to XML can be used in search engines, XML query systems etc, however, to the best of our knowledge, they have never been used for data mining. In this paper, we propose to learn knowledge from changes to XML. Particularly, we study mining frequent patterns from a sequence of changes to an XML document.

Current work on mining frequent patterns from XML documents exploited two types of data in XML: XML content and XML structure. The former discovers which data values occur together frequently[2]; the latter finds out which substructures usually appear together in an XML document[11]. Correspondingly, changes to XML documents can be divided as changes to XML content (also called *content deltas*) and changes to XML structure (also called *structural deltas*). Then, frequent patterns mined from XML content deltas discover which data values frequently change together, whereas frequent patterns mined from XML structural deltas find out which substructures are usually adjusted together. In this paper, we focus on the latter to mine frequent patterns from the changes to XML structure.

Discovered frequently changing subtree patterns can be useful in a wide range of applications. We enumerate some as follows.

- **Structure-based Document Clustering**: Clustering XML documents based on the structures embedded in documents is proposed in [12]. However, it may not be accurate enough to cluster according to structures existing in a particular snapshot only. In some cases, the evolutionary patterns of the structures can distinguish documents with higher precision. Then frequently changing subtree patterns mined from historical versions of different XML documents can be used to discriminate XML documents whose structures are similar in a snapshot but evolve differently.
- **Semantic XML Search Engine**: If two subtrees in an XML structure frequently change together, it is likely that the objects represented by the two subtrees have underlying semantic correlation. Then frequently changing subtree patterns can be used by semantic XML search engine [6], which returns semantically related document fragments that satisfy users' queries.

Therefore, novel knowledge obtained by mining changes to XML documents are useful. We then, in [4], proposed to mine Frequently Changing Subtree Patterns (*FCSP*) from a sequence of changes to an XML document. Given a sequence of historical versions of an XML document, we discovered which subtrees of the XML structure frequently change together. In order to keep the set of discovered *FCSP*s concise so that knowledge can be understood easily, we define the notion of *maximal FCSPs* in this paper, which contains only concise information of *FCSPs.* Our definition on *maximal FCSPs* is fundamentally different from the traditional definition on maximal frequent patterns because of the different framework, which necessitates developing new mining solutions.

The main contributions of this paper are summarized as follows.

- A new problem of mining maximal frequent pattern from structural changes to historical XML documents is formally defined. The inherent relationship between discovered *FCSP*s is exploited to define the concise set of *maximal FCSP*s.
- An algorithm, derived from the FP-growth algorithm, is developed to mine the set of *maximal FCSP*s.
- Experiments are conducted to evaluate the efficiency and scalability of the designed algorithm.

The remainder of the paper is organized as follows. We define the problem of *maximal FCSP* mining formally in Sections 2. Section 3 presents the algorithm we developed for *maximal FCSP* mining. In Section 4, we evaluate the performance of the algorithm based on some experimental results. We conclude our work and discuss future work in Section 5.

## 2 Problem Statement

In this section, we first describe some preliminary concepts and basic change operations resulting in structural changes. Then, we define several metrics to measure the change degree and the change frequency of substructures. Finally, the problem of *maximal FCSP* mining is defined based on the metrics.

An XML document can be represented as a tree according to Document Object Model (DOM) specification. In this paper, we model the structure of an XML document as an unordered tree $T = (N, E)$, where $N$ is the set of nodes and $E$ is the set of edges. Then substructures in the XML document can be modelled as *subtrees*. A tree $t = (n, e)$ is a subtree of $T$, denoted as $t \prec T$, if and only if $n \subset N$ and for all $(x, y) \in e$, $x$ is a parent of $y$ in $T$. Actually, we treat an XML tree $T$ as a *forest* which is a collection of subtrees $t \prec T$. Furthermore, we call subtree $\hat{t}$ an ancestor of subtree $t$ if the root of $\hat{t}$ is an ancestor of the root of $t$. Conversely, subtree $t$ is a descendant of subtree $\hat{t}$. In the context of *FCSP* mining, we consider the following two basic change operations: *Insert(X(name, value), Y)*, which creates a new node $X$, with node name "name" and node value "value", as a child node of node $Y$ in an XML tree structure and *Delete(X)*, which removes node $X$ from an XML tree structure.

Now we introduce some metrics which are used to measure the degree of change and the frequency of change for subtrees. Frequently changing subtree patterns can then be identified based on these metrics.

*Degree of Change* Let <$t^i$, $t^{i+1}$> be two historical versions of a subtree $t$ in an XML tree structure *T*. Let |*d(t, i, i+1)*| be the number of basic change operations which is required to change the structure of $t$ from the $i$th version to the (*i+1*)th version. Let |$t^i \cup t^{i+1}$| be the number of unique nodes of tree $t$ in $i$th version and *(i+1)*th version. Then the *degree of change* for subtree $t$ from version $i$ to version (*i+1*) is:

$$DoC(t,\ i,\ i{+}1) = \frac{|d(t, i, i+1)|}{|t^i \cup t^{i+1}|}$$

□

If the two versions are the same, *DoC* of the subtree will be zero; if the two versions are completely different, *DoC* of the subtree will be one. Obviously, the greater the value of *DoC*, the more dramatically the subtree has changed.

*Frequency of Change* Let <$T^1$,$T^2$,...$T^n$> be a sequence of historical versions of an XML tree structure T. Let <$\Delta_1$,$\Delta_2$,...$\Delta_{n-1}$> be the sequence of *deltas* generated by comparing each pair of successive versions, where $\Delta_i$ ($1 \leq i \leq n-1$) consists of subtrees changed in two versions. Let $S$ be a set of subtrees, $S=\{t_1, t_2, \ldots t_m\}$, *DoC*($t_j$, *i*, *i+1*) be the *degree of change* for subtree $t_j$ from $i$th version to (*i+1*)th version. The *FoC* of the set $S$ is:

$$FoC(S) = \frac{\sum_{i=1}^{n-1} V_i}{n-1}$$

$$where\ V_i = \prod_{j=1}^{m} V_{j_i}\ and\ V_{j_i} = \begin{cases} 1, \text{if } DoC(t_j, i, i+1) \neq 0 \\ 0, \text{if } DoC(t_j, i, i+1) = 0 \end{cases} \quad 1 \leq j \leq m \qquad \square$$

When subtrees in a set change together in every *delta*, *FoC* of the set will be one; when subtrees in a set never change together, *FoC* of the set will be zero.

*Weight* Let <$T^1$,$T^2$,...$T^n$> be a sequence of historical versions of an XML tree structure. Let <$\Delta_1$,$\Delta_2$,...$\Delta_{n-1}$> be the sequence of deltas. Let $S$ be a set of subtrees, $S=\{t_1,t_2,\ldots t_m\}$, *FoC*($S$) be the *frequency of change* for the set $S$. Suppose a user-defined *minimum DoC* for each subtree is $\alpha$, we define the *Weight* of the set of subtrees as follows:

$$Weight(S) = \frac{\sum_{i=1}^{n-1} D_i}{(n-1) * FoC(S)}$$

$$where\ D_i = \prod_{j=1}^{m} D_{j_i}\ and\ D_{j_i} = \begin{cases} 1, \text{if } DoC(t_j, i, i+1) \geq \alpha \\ 0, \text{otherwise} \end{cases} \quad 1 \leq j \leq m \qquad \square$$

If all subtrees in a set change significantly every time they change together, the *Weight* of the set will be one; if subtrees in a set never change significantly when they change together, the *Weight* of the set will be zero.

*Frequently Changing Subtree Pattern (FCSP)* An *FCSP* is defined to be a set of subtrees which not only frequently change together but also usually change significantly when they change together.

**Definition 1.** *: A set of subtrees S=*$\{t_1, t_2, ..., t_m\}$ *is a Frequently Changing Subtree Pattern if it satisfies the following two conditions:*
- *FoC of the set is no less than some user-defined minimum FoC* $\beta$*, FoC(S)* $\geq \beta$.
- *Weight of the set is no less than some user-defined minimum Weight* $\gamma$*, Weight(S)* $\geq \gamma$. □

*Maximal Frequently Changing Subtree Pattern* In classical frequent pattern mining, all subsets of a frequent pattern are frequent as well. Maximal patterns are defined to be those which do not have any frequent superset [10]. Then the complete set of frequent patterns can be represented by the concise set of maximal frequent patterns. However, in the context of *FCSP* mining, *maximal FCSPs* cannot be defined in the same way because of the "Non Downward Closure" property. That is, a subset of an *FCSP* is not necessary an *FCSP* as well. Then, even if we find a set of *FCSPs*, in which each pattern does not have any frequent supersets, we cannot use it to represent the complete set of *FCSP*s. Hence, we examine first which *FCSP*s can be inferred from others.

**Definition 2.** *Given two subtree sets S and* $S_1$*, where* $S = S_1 \cup \{t_1, t_2, ..., t_n\}$*. If* $\forall\ i\ (1 \leq i \leq n)$*,* $\exists\ t_j \in S_1$ *s.t.* $t_j \prec t_i$*, we say* $S_1$ *is subsumed by S, or S subsumes* $S_1$*, denoted as* $S_1 \prec S$. □

*Property 1.* Given two subtree sets *S* and $S_1$ s.t. $S_1 \prec S$. If subtree set *S* is an *FCSP*, then $S_1$ is an *FCSP* as well.

The proof is given [3]. Based on Property 1, we can infer that all subsumed subsets of an *FCSP* are *FCSP*s as well. Then we define *maximal FCSP* as follows.

**Definition 3.** *A set of subtrees is a maximal FCSP if it is an FCSP and it does not subsumed by any other FCSPs.* □

Obviously, the set of *maximal FCSP* is a tightened set of *FCSP*, $\{$*maximal FCSP*$\} \subseteq \{$*FCSP*$\}$. Moreover, the complete set of $\{$*FCSP*$\}$ can be inferred from $\{$*maximal FCSP*$\}$.

**Problem Definition** The problem of *maximal FCSP* discovery can be formally stated as follows: Let $<T_1, T_2, \ldots T_n>$ be a sequence of historical versions of an XML tree structure. Let $<\Delta_1, \Delta_2, \ldots, \Delta_{n-1}>$ be the sequence of *deltas*. A **Structural Delta DataBase** *SDDB* can be formed from the set of *deltas*, where each tuple <*DID*, *SID*, *DoC*> comprises of a delta identifer, a subtree identifier and a *degree of change* for the subtree. Let $S=\{t_1, t_2, \ldots t_m\}$ be the set of changed subtrees such that each $\Delta \subseteq S$. Given an *SDDB*, an *FoC* threshold $\beta$ and a *Weight* threshold $\gamma$, a subtree set $X \subseteq S$ is a **Frequently Changing Subtree Pattern** if $FoC(X) \geq \beta$ and $Weight(X) \geq \gamma$. A subtree set $Y \subseteq S$ is a **maximal Frequently Changing Subtree Pattern** if it is an *FCSP* and it does not subsumed by any other *FCSP*. The **problem of *maximal FCSP* discovery** is to find the set of all maximal frequently changing subtree patterns.

## 3 Algorithms

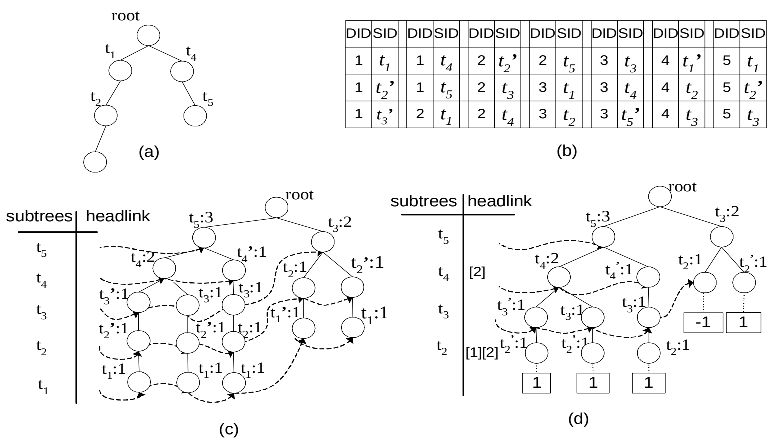
In this section, we present the algorithm to find the set of *maximal FCSPs* efficiently. Before the mining process, the *SDDB* should be generated first from the input of a sequence of historical XML versions. Since existing change detection systems [13][5] can detect all change operations resulting in structural changes (*insert* and *delete*), the *SDDB* can be constructed in a straightforward way. Hence, the following discussion is focused on the mining procedure, with the input of an *SDDB*.

### 3.1 Frequent Changing Subtree Pattern Discovery

We developed an algorithm in [4] to discover the set of *FCSPs*: *Weighted-FPgrowth*, which is based on the well known FP-growth[7] for classical frequent pattern mining. Basically, we construct a similar data structure as *FPtree*, where each node in the FPtree-like structure represents a changed subtree and different labels are used to indicate whether a subtree changes significantly or not in each *delta*. Interested readers can refer to [4] for the details.

### 3.2 Maximal Frequently Changing Subtree Pattern Discovery

The search for the set of *maximal FCSPs* can be achieved by a few optimization strategies as explained below, based on *Weighted-FPgrowth*. In order to mine the set of *maximal FCSPs* efficiently, it is advisable to examine a pattern only if none of its superset which subsumes it is frequent. Thus, a pattern $S$ should be generated before any pattern $S_1$ such



| DID | SID | DID | SID | DID | SID | DID | SID | DID | SID | DID | SID | DID | SID |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $t_1$ | 1 | $t_4$ | 2 | $t_2$' | 2 | $t_5$ | 3 | $t_3$ | 4 | $t_1$' | 5 | $t_1$ |
| 1 | $t_2$' | 1 | $t_5$ | 2 | $t_3$ | 3 | $t_1$ | 3 | $t_4$ | 4 | $t_2$ | 5 | $t_2$' |
| 1 | $t_3$' | 2 | $t_1$ | 2 | $t_4$ | 3 | $t_2$ | 3 | $t_5$' | 4 | $t_3$ | 5 | $t_3$ |

**Fig. 1.** Ancestor Relationship and Modified Weighted-FPtree

that $S_1 \prec S$. According to the Definition 2 on subsumption relationship, we have the following optimizing technique.

**Optimization 1** *FCSPs containing subtrees rooted at nodes of higher-levels in an XML tree should be generated earlier than those containing subtrees rooted at nodes of lower-levels.*

Optimization 1 guides how to order the list of subtrees in the head table of *Weighted-FPtree*. We decide to arrange individual changed subtrees in the reverse order of depth-first first traversal in consideration of the following property.

*Property 2.* Given two subtree sets, $S_1$ and $S$, such that $S_1 \prec S$, the projected *delta* sets of $S$ is same as the projected *delta* sets of $S_1$.

The proof of the property is given in [3]. According to Property 2, we are presented with the opportunity to mine both $S$ and $S_1$ from a same *conditional Weighted-FPtree*. For example, given the ancestor relationship shown in Figure 1 (a), we arrange individual subtrees in reverse order of depth-first traversal, such as $\{t_5,\ t_4,\ t_3,\ t_2,\ t_1\}$, then the *conditional Weighted-FPtree* constructed for subtree $t_1$ can be used to mine not only all patterns related to $t_1$ but also all patterns related to $t_2$ (or $t_3$) but $t_1$

As explained in [4], labels of some nodes in *conditional Weighted-FPtree* need to be shifted to record correct information. In this case, a *conditional Weighted-FPtree* may not be sharable. Hence, we propose an alternative technique of shifting node labels. We append a tag to each path in the *conditional Weighted-FPtree*, which indicates the states of subtrees whose patterns are currently being mined. For example, given a transformed structural delta database shown in Figure 1 (b), where $t_i$ and $t_i$*'* mean subtree $t_i$ change significantly or insignificantly in this delta, a *Weighted-FPtree* can be constructed as in Figure 1 (c). Figure 1 (d) shows the *conditional Weighted-FPtree* for subtree $t_1$. Each path is appended with a tag: "1" indicates that $t_1$ changed significantly in this path while "-1" indicates it changed insignificantly in this path.

Moreover, with the above scheme on ordering subtrees and the technique making *conditional Weighted-FPtree* shareable, we can quickly decide whether or not to examine a pattern, depending on the state of the pattern which subsumes it. For example, from the *conditional Weighted-FPtree* shown in Figure 1 (d), we mine pattern $\{t_1, t_2\}$ first. Only if $\{t_1, t_2\}$ is not an *FCSP*, we go on mining $\{t_2\}$ from the same data structure. Otherwise, $\{t_2\}$ must be an *FCSP* but not maximal. As shown in Figure 1 (d), there are two counts maintained by $t_2$ to record the *Weight* information for $\{t_1, t_2\}$ and $\{t_2\}$ respectively. Actually, all patterns related to subtree $t_2$ can be decided in the same way. Thus, we do not need to mine patterns related to $t_2$ from the original *Weighted-FPtree* and have the following optimization.

**Optimization 2** *From the head table of (conditional) Weighted-FPtree, only the last subtree and subtrees which are not descendant of the subtree whose patterns are just examined in the previous turn need to be mined.*

When the *Weighted-FPtree* contains a single path, we have the following optimization strategy.

**Optimization 3** *If the (conditional) Weighted-FPtree contains single path, maximal FCSPs can be generated directly from subtrees in the head table which are 1) with their node identifier as* $t_i$ *and 2) either last subtree or not descendant of the subtree mined in the previous turn.*

## 4 Experiment Results

In this section, we study the performance of the proposed algorithms. The algorithms are implemented in Java. Experiments are performed on a Pentium IV 2.8GHz PC with 512 MB memory. The operating system is Windows 2000 professional. We implemented a synthetic structural delta generator by extending the one used in [1]. The default number of deltas is 10000 and the number of changed subtrees is 1000.

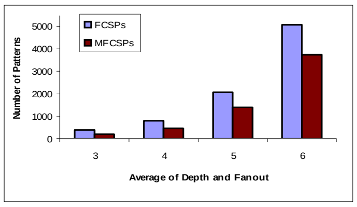
We carried out four experiments to show the conciseness of *maximal FCSP*s, the efficiency and scalability of designed algorithm compared with a naive algorithm which discover the complete set of FCSPs first and then filter non-maximal ones.

– Conciseness of *maximal FCSP*s: Firstly, we contrast the size of the set of *maximal FCSP*s with the size of the complete set of *FCSP*s by varying the average depth and fanout of the ancestor relationships. As shown in Figure 2 (a), the gap between the two sizes is greater when average depth and fanout of ancestor relationships are greater, since more *FCSP*s might be subsumed by their supersets.
– Efficiency Study: We compare the execution time of the developed algorithm against the naive algorithm by varying the threshold of *min_FoC* from 2% to 10%. As shown in Figure 2 (b), when the threshold is lower, our algorithm is more efficient than the naive one because the naive algorithm need to verify more patterns, on the contrary, designed algorithm has the chance to skip checking more patterns.
– Scalability Study I: We test the scale-up features of the two algorithms against the number of deltas, which is varied from 8K to 80K. Figure 2 (c) shows that, when the number of deltas is larger, the gap between the two algorithms is greater, since the more subtrees potentially be *FCSP*s.
– Scalability Study II: We also observed the scalability of the two algorithms with respect to the number of discovered *maximal FCSP*s. As presented in Figure 2 (d), for mining the same number of *maximal FCSP*s, the designed algorithm is faster than the naive one. Furthermore, when the size of the set of *maximal FCSP*s increases, the designed algorithm scales even better.

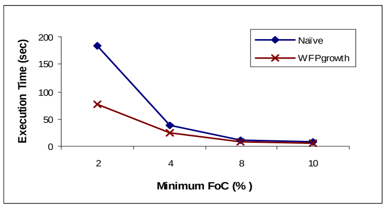
## 5 Conclusions & Future Work

This paper proposed a novel problem of mining maximal frequent patterns based on changes to XML structures: *maximal FCSP* mining. An algorithm, optimized *Weighted-FPgrowth*, is designed and implemented. Preliminary experiment results demonstrated that the conciseness of the set of *maximal FCSP*s. Moreover, the designed algorithm is significantly more efficient than the naive algorithm.

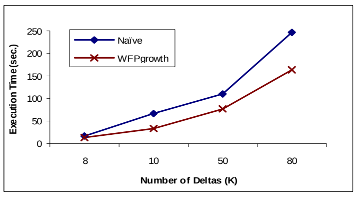
As ongoing work, we would like to collect real life dataset to verify the semantic meaning of discovered *maximal FCSPs*. In addition, we would also like to investigate issues about mining frequent patterns from content deltas and hybrid (content and structural) deltas of historical XML documents.



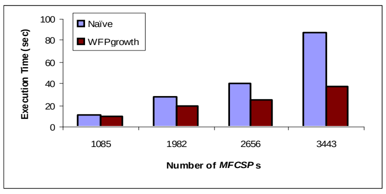
(a) Variation of Avg. Depth and Fanout



(b) Variation of Minimum FoC



(c) Variation of Number of Deltas



(d) Variation of Number of *MFCSP*s

**Fig. 2.** Experiment Results

## References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. VLDB*, pages 487–499, 1994.
2. D. Braga, A. Campi, S. Ceri, M. Klemettinen, and P. L. Lanzi. A tool for extracting xml association rules from xml documents. In *Proc. IEEE ICTAI*, pages 57–65.
3. L. Chen and S. S. Bhowmick. Web structural delta association rule mining: Issues, chanllenges and solutions. In *TR, NTU, Singapore*, http:// www.ntu.edu.sg/ home5/ pg02322722/ TR.html.
4. L. Chen, S.S. Bhowmick, and L.T. Chia. Mining association rules from structural deltas of historical xml documents. In *Proceedings of PAKDD 2004, Sydney, Australia*, 2004.
5. G. Cobena, S. Abiteboul, and A. Marian. Detecting changes in xml documents. In *Proc. ICDE*, 2002.
6. S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. Xsearch: A semantic search engine for xml. In *Proc. VLDB*, 2003.
7. J.W. Han, J. Pei, and Y.W. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD*, pages 1–12, 2000.
8. M. J.Zaki and C. C. Aggarwal. Xrules: An effective structural classifier for xml data. In *SIGKDD, Washington, DC, USA*, 2003.
9. M. L. Lee, L. H. Yang, W. Hsu, and X. Yang. Xclust: Clustering xml schemas for effective integration. In *ACM 11th CIKM, McLean, VA*, 2002.
10. Jr. R. J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the ACM SIGMOD, Seattle, WA*, 1998.
11. A. Termier, M.C. Rousset, and M. Sebag. Treefinder: A first step towards xml data mining. In *Proc. IEEE ICDE*, pages 450–457, 2002.
12. K. Wang and H. Liu. Discovering structural association of semistructured data. In *IEEE TKDE, vol.12*, pages 353–371, 2000.
13. Y. Wang, D.J. DeWitt, and J.Y. Cai. X-diff: An effective change detection algorithm for xml documents. In *Proc. ICDE*, 2003.