FASST Mining: Discovering Frequently Changing Semantic Structure from Versions of Unordered XML Documents

Qiankun Zhao and Sourav S Bhowmick

School of Computer Engineering
Nanyang Technological University, Singapore
{pg04327224, assourav}@ntu.edu.sg

Abstract. In this paper, we present a FASST mining approach to extract the *frequently changing semantic structures* (FASSTs), which are a subset of semantic substructures that change frequently, from versions of unordered XML documents. We propose a data structure, H-DOM$^+$, and a FASST mining algorithm, which incorporates the semantic issue and takes the advantage of the related domain knowledge. The distinct feature of this approach is that the FASST mining process is guided by the user-defined *concept hierarchy*. Rather than mining all the frequent changing structures, only these frequent changing structures that are semantically meaningful are extracted. Our experimental results show that the H-DOM$^+$ structure is compact and the FASST algorithm is efficient with good scalability. We also design a declarative FASST query language, FASSTQUEL, to make the FASST mining process interactive and flexible.

1 Introduction

Frequent substructure mining [3, 5] is one of the most well researched topics in the area of XML data mining. Current research on frequent substructure mining is to extract substructures that occur frequently in individual XML document or in collections of XML documents. However, most of the existing research of frequent substructure mining focuses on snapshot data collections, while XML data is dynamic in real life applications.

The dynamic nature of XML leads to two challenging problems. First, is the maintenance of frequent substructures. For this problem, incremental data mining techniques [1] can be applied to maintain the mining results. Second, is the discovery of novel knowledge such as *association rules* and *frequent changing structures*, which are hidden behind the historical changes to XML data as described in [6]. The frequent changing structure (FCS) is defined as substructures in the XML versions that change frequently and significantly in the history. In [6], we proposed a novel approach to discover the frequently changing structures from the sequence of historical structural changes to unordered XML. The usefulness and importance of such frequently changing structures, with corresponding applications, have also been discussed. To make the structure discovering process efficient, an expressive and compact data model, Historical-Document Object Model (H-DOM), is proposed. Using this model, two basic algorithms, which can discover all the *frequently changing structures* with only two scans of the XML version sequence, were designed and implemented. We deal with unordered XML documents since the unordered model of XML is more suitable for most database applications [4]. Hereafter, whenever we say XML, we mean unordered XML.

However, discovering all the frequently changing structure is challenging due to the presence of exponential number of substructures, while the mining results of our previous approach include any arbitrary substructure that change frequently. We observed that not all the frequently changing substructures are semantically significant and meaningful. Usually, in a specific domain, users are interested in only some specified structures that corresponding to certain semantic concepts. To reduce the number of structures in the mining results and keep all the meaningful structures, we propose to incorporate the semantic constraints in the form of user-defined *concept hierarchy* into the frequently changing structure mining process.

In this paper, we defined a subset of frequently changing structures as *Frequently chAnging Semantic STructures* (FASSTs) based on the *dynamic metrics* and *concept hierarchy.* Given a sequence of XML documents (which are different versions of the same XML document), the objective of FASST mining is to discover the frequently changing structures according to the user specified semantic concepts.

2 The FASST Mining Problem

In this section, we present the preliminaries and problem statement for the FASST mining problem. First, a set of dynamic metrics is proposed to measure the changes to XML structural data. After that, the concept of semantic structure is presented. Lastly, we formulate the FASST mining problem. Details and examples of the definitions are available in [6], only a brief introduction is presented here.

2.1 Dynamic Metrics

We model the structures of XML documents as unordered, labeled, rooted trees. We denote the structure of an XML document as $S = (N, E, r)$, where N is the set of labeled nodes, E is the set of edges, $r \in N$ is the root. We do not distinguish between elements and attributes, both of them are mapped to the set of labeled nodes. Each edge, $e = (x, y)$ is an ordered pair of nodes, where x is the parent of y. The *size* of the structure S, denoted by $|S|$, is the number of nodes in N.

Definition 1 (Substructure). *A structure s = (N', E', r') is a substructure of $S = (N, E, r)$, denoted as $s \preceq S$, provided i) $N' \subseteq N$, and ii) $e = (x, y) \in E'$, if and only if x is the parent of y in E.*

Definition 2 (Structural Delta). *Let S_i and S_{i+1} be the tree representations of two XML documents X_i and X_{i+1}. The structural delta from X_i to X_{i+1} is represented as $\triangle_i$, where $\triangle_i$ is a structural edit script $\langle o_1, o_2, \cdots, o_m\rangle$ that transforms S_i into S_{i+1}, denoted as $S_1 \overset{o_1}{\rightarrow} s_1 \overset{o_2}{\rightarrow} \cdots \overset{o_m}{\rightarrow} S_{i+1}$.*

Definition 3 (Consolidate Structure). *Given two structures S_i and S_j, where $r_i = r_j$. The consolidate structure of them is $S_i \uplus S_j$, where i) $N_{s_i \uplus s_j} = N_{s_i} \cup N_{s_j}$, ii) $e = (x, y) \in E_{s_i \uplus s_j}$, if and only if x is the parent of y in $E_{s_i} \cup E_{s_j}$.*

We observed that different substructures of the XML document might change in different ways at different frequencies. To evaluate their historical behaviors, we propose a set of dynamic metrics. The first metric is called structure dynamic.

Definition 4 (Structure Dynamic). *Let $\langle S_i, S_{i+1}\rangle$ be the tree representations of XML documents $\langle X_i, X_{i+1}\rangle$. Suppose $s \preceq S_i$. The structure dynamic of s from document X_i to document X_{i+1}, denoted by $N_i(s)$, is defined as: $N_i(s) = \frac{|\triangle_{s_i}|}{|s_i \uplus s_{i+1}|}$.*

Here $N_i(s)$ is the structure dynamic of s from version i to $i + 1$. $N_i(s)$ is the percentage of nodes that have changed from X_i to X_{i+1} in s against the number of nodes in its consolidation structure. A larger value of structural dynamic implies that the more *significantly* the substructure changed.

Definition 5 (Version Dynamic). *Let $\langle S_1, S_2, \cdots, S_n\rangle$ be the tree representations of XML documents $\langle X_1, X_2, \cdots X_n\rangle$. Suppose $s \preceq S_j$. The version dynamic of s, denoted as $V(s)$, is defined as:*

$$V(s) = \frac{\sum_{i=1}^{n-1} v_i}{n-1} \text{ where } v_i = \begin{cases} 1, \text{ if } |\triangle_{s_i}| \neq 0; \\ 0, \text{ if } |\triangle_{s_i}| = 0; \end{cases}$$

Similarly, it can also be observed that the larger the value of version dynamic is, the more *frequently* the substructure changed in the history.

Definition 6 (DoD). *Let $\langle S_1, S_2, \cdots, S_n\rangle$ be the tree representations of XML documents $\langle X_1, X_2, \cdots, X_n\rangle$. Suppose $s \preceq S_j$, $N_i(s)$ and $V(s)$ are the values of structure dynamic and version dynamic of s; α is the predefined threshold for structure dynamic. The DoD for s is defined as:*

$$DoD(s, \alpha) = \frac{\sum_{i=1}^{n} d_i}{(n-1) * V(s)} \text{ where } d_i = \begin{cases} 1, if\ N_i \geq \alpha \\ 0, if\ N_i < \alpha \end{cases}$$

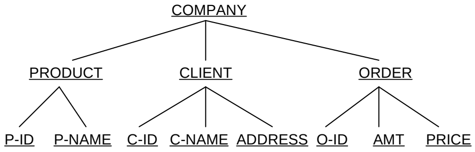


Fig. 1. An Example of Concept Hierarchy

The metric DoD is defined based on the threshold of structure dynamic. It represents the fraction of versions, where the structure dynamic values for the substructure are no less than the predefined threshold α, against the total number of version the substructure has changed over the history. Extended from the structure dynamic, the value of DoD implies the overall significance of the substructure, the larger the value is, the more *significant* the changes are.

2.2 Semantic Structure

One of the distinctive features of XML is that XML is semantic. Tags within the XML documents are self-describing. However, if we represent an XML document as a tree structure, not all the substructures are semantically significant to users. For example, users in the e-commerce domain may be more interested in the substructures corresponding to *products* and *clients* than other substructures such as *Name*. In this section, we define the *semantic structure* to represent substructures that are semantically meaningful.

Definition 7 (Semantic Structure). *Given a concept C in a specific domain, a structure s is a semantic structure of concept C, denoted as $s \simeq C$, if s provides the required information of the concept C.*

Based on the definition, it is obvious that the semantic structures are based on the underlining concepts, which is domain dependent. There are two approaches to obtain such concepts. The first approach is to extract interesting concepts from ontology in the corresponding domain. Another approach is to build the concepts based on DTDs (Document Type Definitions) used in this domain. Recently, DTDs are widely used to specify the legal building blocks in XML documents. Each legal building block corresponds to a concept in ontology.

However, users may not be interested in all the semantic concepts/structures while the number of concepts/structures can be huge. Moreover, even in the same domain, different users may have different interests. For instance, in the e-commerce domain, the material control people may be more interested in the semantic structure *products* than others; while the marketing people may be more interested in the semantic structure *clients* than others. Given a set of semantic concepts/structures, users can specify the concepts they are interested in.

Our research focuses on extracting the frequently changing structures that are *semantically meaningful.* The set of user-specified concepts is used to guide the FASST mining process. Similar to [2], interested concepts are represented in a hierarchy that specifies the relation among them. Nodes of the hierarchical structure can be classified as *primitive* or *nonprimitive.* The *primitive* concepts, which represent the basic elements in a domain, reside in the lowest level in the hierarchy; all *nonprimitive* concepts, which consist of a conglomeration of the primitive concepts, reside in the higher level of the hierarchy. The higher the node's level, the more complex is the concepts it represents. Figure 1 shows an example of concept hierarchy. The leaf nodes such as *P-ID*, and *P-NAME* are primitive concepts; while internal nodes and root node such as *CLIENT* and *COMPANY* are nonprimitive concepts. In our FASST mining, we assume that the specified concept hierarchy is provided by users.

2.3 Problem Statement

In our previous work [6], we have defined the frequently changing structures (FCS) based on the dynamic metrics as substructure that have version dynamic and degree of dynamic values no less than the user-defined thresholds. Similarly, here we give a formal definition of *Frequent chAnging Semantic STructure* (FASST).

Definition 8 (FASST). *Let* $\langle S_1, S_2, \cdots, S_n \rangle$ *be the tree representations of XML documents* $\langle X_1, X_2, \cdots, X_n \rangle$*; H is a concept hierarchy that contains a set of concepts* $\{c_1, c_2, \cdots, c_i\}$*; the thresholds for structure dynamic, version dynamic and DoD are* α, β, γ *respectively. A structure* $s \preceq S_j$ *is a* ***FASST*** *in this sequence if and only if i)* $V(s) \geq \beta$*, ii)*$DoD(s, \alpha) \geq \gamma$*, and iii)* $s \simeq c_m$*, where* $1 \leq m \leq i$*.*

The FASST is defined based on the predefined thresholds of the dynamic metrics and a set of user-defined concepts. To be a FASST, it must be a semantic structure (defined by H) and change at certain frequency (defined by β) and corresponding changes must be significantly enough (defined by α and γ). The FASST mining problem is to discover all the FASSTs from a sequence of XML documents with the user-defined concepts and thresholds for the dynamic metrics.

3 Algorithm

In this section, we present our FASST mining algorithm. First, we introduce the H-DOM^{+} data structure to store and represent relevant historical structural information. After that, detail of the FASST algorithm is presented.

3.1 The H-DOM$^+$ Structure

The structure of an XML document can be represented and stored as a tree such as the DOM tree proposed by W3C. In this section, we present an H-DOM$^+$ model to represent the history of changes to XML data. The H-DOM$^+$ is an extension of the DOM model with some historical properties so that it can compress the history of changes to XML into a single H-DOM$^+$ tree. Formally, we define an H-DOM$^+$ tree as follows:

Definition 9 (H-DOM$^+$). *An H-DOM$^+$ tree is a 4-tuple $H = (N, A, v, r)$, where N is a set of object identifiers; A is a set of labelled, directed arcs (p, l, c) where $p, c \in N$ and l is a string; v is a function that maps each node $n \in N$ to a set of values (C_n, C_v), C_n is an integer and C_v is a set of integers; r is a distinguished node in N called the root of the tree.*

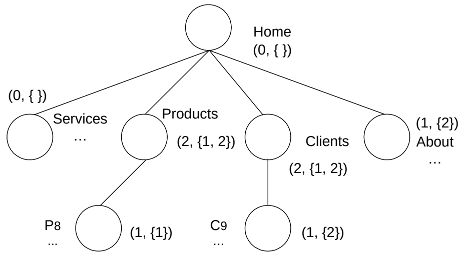
We now elaborate on the parameters C_n and C_v. The two parameters are introduced to record the historical changes for each substructure. C_n is an integer that records the number of versions that a substructure has changed significantly enough (the structure dynamic is no less the corresponding threshold). C_v is a set of integers that represents the versions where the substructure has changed in the history. For instance, a value of " i " denotes that the structure has changed from version i to version $i+1$. Differ from the H-DOM model in [6], the types of changes are specified using integers with " + " and " $-$ " in H-DOM$^+$. Such knowledge will be used in our proposed FASST query language, FASSTQUEL, to mine different types of FASSTs. A value of " $-i$ " in C_v means the structure is "deleted" in version *i+1* while a value of " $+i$ " means the structure is "inserted". In the H-DOM$^+$ tree, the C_v value for each structure is finally updated by using the formula: $C_v(s) = C_v(s_1) \cup C_v(s_2) \cup \cdots \cup C_v(s_j)$, where $s_1, s_2, \cdots, s_j$ are the substructures of s. In the updating process, insertion and deletion of a structure is determined by the majority of the changes to its substructures (if the number of insertions among its substructures is no less than deletions, then we consider it as an "insertion" in that version. Otherwise, it is considered as a "deletion").

With C_v and C_n, the values of structure dynamic, version dynamic, and DoD can be calculated based on this model as follows.

- $N_i(s) = \frac{1}{|s_i \uplus s_{i+1}|} \sum C_v(s_j)[i]$, where s_j is the list of substructures of s, $C_v(s_j)[i]$ is 1 if any of $\pm i$ is in $C_v(s_j)$, otherwise it is 0.
- $V(s) = \frac{1}{n-1} \sum_{i=1}^{n-1} C_v[i]$, where $C_v(s)[i]$ is 1 if any of $\pm i$ is in $C_v(s)$, otherwise it is 0; n is the total number of XML documents.
- $DoD(s) = C_n(\sum_{i=1}^{n-1} C_v[i])^{-1}$, where $C_v(s)[i]$ is 1 if any of $\pm i$ is in $C_v(s)$, otherwise it is 0; n is the total number of XML documents.

```
Algorithm 1 H-DOM$^+$ Construction
Input:
  $\langle X_1, X_2, \cdots, X_n \rangle$: A sequence of XML
  S(D): Tree representation of the DTD
Output:
  $H$: H-DOM$^+$ Tree
Description:
  $H \leftarrow (S(X_1) \cap S(D_2))$
  for $(k = 2; k \leq n; k++)$ do
      $\triangle$ = SX-Diff$(X_k, X_{k-1})$
      $H$ = Mapping$(H, \triangle)$
  end for
  Return$(H)$
```

(a) H-DOM$^+$ construction



(b) Part of an H-DOM$^+$ Tree

Figure 2.

3.2 FASST Mining

There are three major phases in our FASST mining. The *H-DOM$^+$ construction* phase, the *FASST extraction* phase, and the *visualization* phase. Since the visualization phase is straightforward, we discuss the first two phases in turn.

The H-DOM$^+$ Construction Phase: Figure 2 (a) describes the phase of H-DOM$^+$ construction. Given a sequence of historical XML documents, the H-DOM$^+$ tree is initialized as the structure of the first version. After that, the algorithm iterates over all the other versions by extracting the structural changes and mapping them into the H-DOM$^+$ tree. The SX-Diff function is a modification of the X-Diff [4] algorithm that generates only the structural change from two different versions of a document. The structural changes are mapped into the H-DOM$^+$ tree according to mapping rules described in Figure 2 (a). The SX-Diff function and the mapping phase iterate until no more XML document is left in the sequence. Finally, the H-DOM$^+$ tree is returned as the output of this phase. Figure 2 (b) is an example of an H-DOM$^+$ tree.

Given an XML document and the corresponding DTD, according to the DTD, it is possible to know that some of the elements (attributes) cannot be changed individually. For example, in a DTD, some elements (attributes) may be defined as *required* with exactly one occurrence. Such nodes cannot be inserted or deleted individually, which means they can only change with the insertion or deletion of their parent nodes. Based on this observation, elements (attributes) in the XML documents can be classified into two groups. Elements (attributes) that cannot be inserted or deleted individually are classified into group 1, others are in group 2. In the initialize process, rather than store the entire structure of the first version, we only map nodes that belong to group 2. Nodes in group 1 are ignored.

```
Algorithm 2 Mapping
Input:
  $H$: H-DOM$^+$ Tree
  $\alpha$: Threshold of structure dynamic
  $\triangle$: Structural delta
Output:
  $H$: The updated H-DOM$^+$ tree
Description:
for all $n_i \in \triangle$ do
  if $n_i \notin H$ then
    update $C_n(n_i)$
  end if
  if $N_i(n_i) \geq \alpha$ then
    update $C_v(n_i)$
    $n_i = n_i$.parent($H$)
  end if
end for
Return($H$)
```

(a) Mapping Algorithm

```
Algorithm 3 FASST Extraction
Input:
  $H$: H-DOM$^+$ Tree
  $T$: User specified concept hierarchy
  $\beta, \gamma$: Threshold of version dynamic and $DoD$
Output:
  $F$: A set of nodes where FASSTs are rooted
Description:
1: for all $n_j$=Bottom-upTrav(H)$\neq$ $null$ do
2:   while $T_i$=Bottom-upTrav($T$)$\neq$ $null$
3:   if $S(n_j) \simeq T_i$ , then
4:   for all $s \preceq S(n_j)$ do
5:     if $C_n < \gamma \times V(s)$, $\{n_j = n_j.next$, break$\}$
6:     if $V(n_j) \geq \beta$ & $DoD(n_j) \geq \gamma$, $\{F = F \cup n_j\}$
7:   end for
8:   break; end if
9: end for
10: Return($F$)
```

(b) FASST Extraction Algorithm

Figure 3.

Algorithm 2 in Figure 2 (a) describes the mapping function. Given the H-DOM$^+$ tree and the structural changes, this function is to map the structural changes into the H-DOM$^+$ tree and return the updated H-DOM$^+$ tree. The idea is to update the corresponding values of the nodes in the H-DOM$^+$ tree. The values are updated according to following rules:

i) If the node does not exist in the H-DOM$^+$ tree, then the node is inserted. The value of $\pm i$ is inserted into C_v where i is the version number of the structural delta. In addition, the N_i value is calculated. If $N_i \geq \alpha$, then C_n is set to 1 and the C_n values of its parent nodes are incremented by 1 until N_i is less than α. Otherwise, C_n is set to 0 and the process terminates.

ii) For nodes that exist in the H-DOM$^+$, the value of C_v is updated by inserting the value $\pm i$ into C_v if $\pm i$ is not in C_v. The value of C_n is also updated based on N_i and α. Similarly, If $N_i \geq \alpha$, then C_n is incremented by 1 and the C_n values of its parent nodes are updated based on the same rule until N_i is less than α. Otherwise, C_n does not change and the process terminates.

The FASST Extraction Phase: In this phase, given the H-DOM$^+$ tree, the FASSTs are extracted based on the user-defined concept hierarchy. First the substructures are compared with the user-specified concept hierarchy as shown in line 3 in Figure 3 (b). If the structures are instances of the concepts in the hierarchy, then the values of the required parameters (version dynamic, and DoD) for each node are calculated and compared against the predefined thresholds as shown in lines 5 and 6. Since for a FASST, both its version dynamic and DoD should be no less than the thresholds, we first calculate only one of the parameters and determine whether it is necessary to calculate the other parameter. In our algorithm, the version dynamic for a node is checked against the corresponding threshold first. If it is no less than the threshold, then we check its DoD. Considering the traversal strategy of the H-DOM$^+$ tree, we use the bottom-up method since the set of interesting concepts is represented in a hierarchical manner with primitive concepts in the lower level. Guided by the concept hierarchy, the FASST extraction phase can be more efficient.

Lemma 1. *Let S_1 and S_2 be any two structures, $S_2 \preceq S_1$. Given the threshold for DoD as γ, the necessary condition for structure S_1 to be a FASST is that $C_n(S_1) \geq \gamma \times V(S_2)$.*

From the above lemma, we observed that it is not necessary to traverse the entire H-DOM$^+$ tree. We can skip checking some structures that cannot be FASSTs. Based on this lemma, for any nodes, rather than calculate its version dynamic value, the C_n value of the node is checked against the value of $\gamma \times V(S_i)$, where S_i is any of its substructures. If $C_n < \gamma \times V(S_i)$, then it is not necessary to calculate the version dynamic and DoD for this structure since it cannot be a FASST. This pruning technique is shown in Figure 3 (b) in lines 5 and 6.

4 FASSTQUEL: Query Language for FASST

To make the FASST mining process interactive, we design a FASST query language called *FASSTQUEL*. In this section, we discuss the syntax of the language and how it is useful.

The FASST query language consists of the specifications of four major parameters in FASST extraction from a sequence of XML documents. They are *types of FASST, relevant source, a hierarchy of concepts,* and *threshold values.* The syntax for the FASST query language is defined in a simplified BNF grammar (words in `type writer` font represents keywords) as shown in Figure 5 (i):

- "`EXTRACT` ⟨*structure_type*⟩" specifies that the FASSTs to be discovered are of type "⟨ *structure_type* ⟩". The following types of FASSTs are supported in our query language:
  - *Insertion-based* (Only insertions are considered as changes)
    ⟨*structure_type*⟩ :: = `Ins_FASST`
  - *Deletion-based* (Only deletions are considered as changes)
    ⟨*structure_type*⟩ :: = `Del_FASST`
  - *FASST* (Both insertions and deletions are considered as changes)
    ⟨*structure_type*⟩ :: = `All`
- "`FOR` ⟨*concepts*⟩" specifies that the concept hierarchy to be used to guide the FASST mining. The concept hierarchy can be stored in an XML document.

- "`FROM` ⟨*source*⟩" specifies on which dataset the FASST extraction should be performed. It can the entire sequence or from version i to version j, which are specified as `All` and [i, j] respectively.
- "`WHERE THRESHOLD` = ⟨*N, V, DoD*⟩" specifies the thresholds for structure dynamic, version dynamic, and DoD. If the any of the threshold values is not specified by the user, then a default value is used.

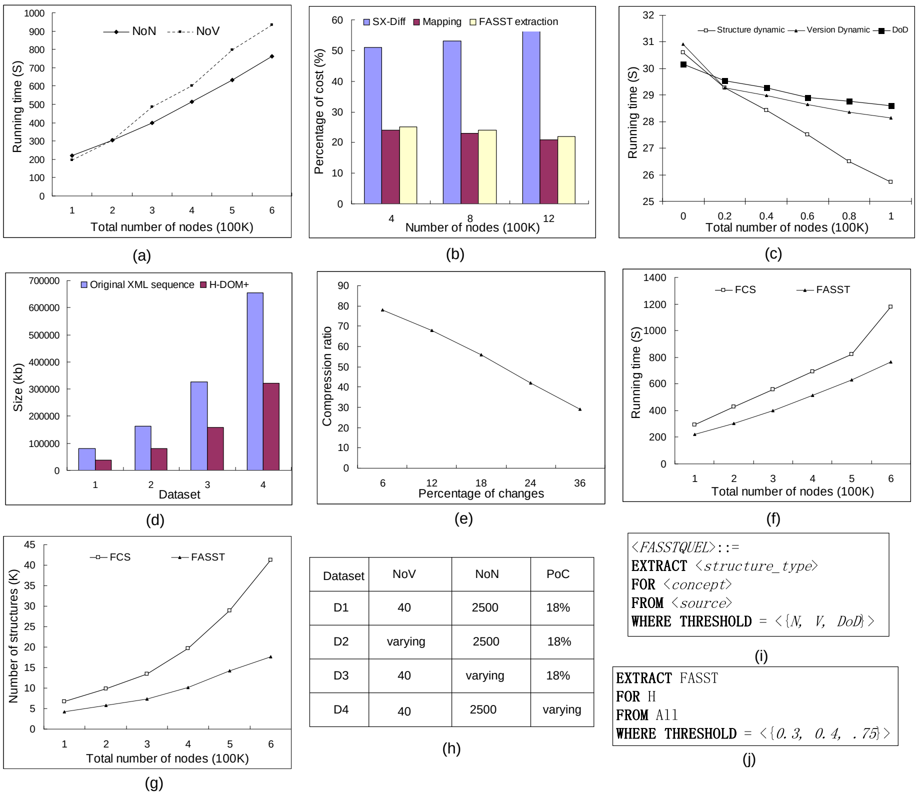
For example, given a hierarchy of concepts H, to extract all types of FASSTs from a sequence of n XML documents with the thresholds for structure dynamic, version dynamic and DoD are specified as $0.3, 0.4$, and 0.75 respectively. The FASST query can be formulated as shown in Figure 5(j). The FASST query language is proposed for interactive FASST mining. That is users may not be able to get their desired knowledge at the first hit. Based on the mining results, users can specify and modify their requirements explicitly using this query language. Moreover, the FASST query language makes the interactive mining process more efficient.

5 Performance Evaluation

Experiments are conducted on a P4, 1.7GHz PC with 256M RAM, running Microsoft Windows 2000 Professional. The algorithm is implemented in Java. In the following experiments, the real data, SIGMOD XML document, is downloaded from UW XML repository [1]. Based on this XML document, sequences of synthetic XML versions are generated using our synthetic XML delta generator. Similar to the experiments in [6], we vary the characteristics and the parameters for the algorithm to evaluate the performance of FASST mining. All the datasets are generated based on the basic dataset generated from the SIGMOD XML. The basic dataset, D_1 in Figure 4 (i), consists of 40 versions of XML documents, with an average number of 2500 nodes. The average percentage of changes between any consecutive versions is 18% in the basic dataset, which consists of 9% of insertion and 9% of deletion.

Figure 4 (a) shows how the running time changes by varying the total number of nodes in the XML sequence. There are two ways of increasing the total number of nodes in the sequence. One way is to increase the number of versions (NoV) in the XML sequence, another way is to increase the average number of nodes (NoN) in each version. Datasets D_2 and D_3 are used in the following experiments. The threshold values for *structure dynamic, version dynamic,* and *DoD* are fixed to 0.2, 0.2, and 0.2 respectively. Both results show good scalability with the total number of nodes, while the running time is more sensitive to the number of versions in the XML sequence than the average number of nodes in each

[1] http://www.cs.washington.edu/research/xmldatasets



Dataset	NoV	NoN	PoC
D1	40	2500	18%
D2	varying	2500	18%
D3	40	varying	18%
D4	40	2500	varying

(h)

```
<FASSTQUEL>::=
EXTRACT <structure_type>
FOR <concept>
FROM <source>
WHERE THRESHOLD = <{N,  V,  DoD}>
```

(i)

```
EXTRACT FASST
FOR H
FROM All
WHERE THRESHOLD = <{0.3,  0.4,  .75}>
```

(j)

Figure 4. Experiment Results

version. This is due to the fact that the change detection process is the major cost for FASST mining, as we can see from Figure 4 (b). The first three datasets in Figure 4 (i) are used. As shown in Figure 4 (b), among three processes: *SX-Diff*, *Mapping*, and *FASST Extraction*, we observed that the SX-Diff process is the most expensive process that takes more than half of the running time.

Figure 4 (c) shows how the running time changes by varying the thresholds for the dynamic metrics. The D_1 dataset is used. In the three experiments, we vary one of the thresholds and fixed the thresholds for the other two to 0.2. It can be observed that the running time does not change significantly when the thresholds of dynamic metrics change. This is due to the fact that the most expensive process, SX-Diff, is independent on the thresholds.

Figure 4 (d) shows the size of the H-DOM$^+$ tree comparing with the original size of the XML sequence in previous experiments. From the result it can be concluded that our H-DOM$^+$ model is very compact (around 50% of the original XML sequence). By varying the characteristics of the datasets, we find out that the compactness of the H-DOM$^+$ structure is sensitive to the percentage of changes in the dataset, while the number of versions and the average number of nodes in each version do not affect the compactness. As shown in Figure 4 (e), when the percentage of changes increases the compactness of the H-DOM$^+$ structure will decrease. It is because that when the percentage of changes increases, the overlap among the XML sequence will decrease. Consequently, the space saved by H-DOM$^+$ structure will decreases. The dataset D_4 is used in this experiment. Although the compactness of the H-DOM$^+$ data structure depends on the datasets, but it is guaranteed that the size of the H-DOM$^+$ is no larger than the original datasets in the worst case.

Figures 4 (f) and (g) show the performance comparison of FCS and FASST. Figure 4 (f) shows the comparison of the running time. The two set of experiments are conducted with the same threshold values for the dynamic metrics using the D_1 dataset. It can be observed that the running time of FASST has been improved significantly. Figure 4 (g) shows the number of structures in the mining results with the same thresholds for the dynamic metrics. It shows that the number of structures in FASST mining result is reduced by almost 40% from the FCS mining result. This two results shows that the object of our FASST mining has been achieved successfully.

6 Conclusions

In this paper, we propose an approach to extract the FASSTs from a sequence of historical XML documents. We propose an H-DOM$^+$ to store and represent the historical structural information of the XML documents sequence. Using the H-DOM$^+$, an algorithm is proposed to mine the FASSTs. Experimental results show that FASST has good scalability and efficiency. We also propose a declarative FASST query language to make the mining process interactive.

References

1. V. Ganti, J. Gehrke, and R. Ramakrishnan. DEMON: Mining and monitoring evolving data. In *Proc. IEEE ICDE*, pages 439–448, 2000.
2. J. Han and Y. Fu. Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases. In *Proc. KDD Workshop*, pages 157–168, 1994.
3. A. Inokuchi, T. Washio, and H. Motoda. An apriori based algorithm for mining frequent substructures from graph data. In *Proc. PKDD*, pages 13–23, 2000.
4. Y. Wang, D. J. DeWitt, and J.-Y. Cai. X-diff: An effective change detection algorithm for XML documents. In *Proc. ICDE*, pages 519–530, 2003.
5. M. J. Zaki. Efficiently mining frequent trees in a forest. In *Proc. ACM SIGKDD*, pages 71–80, 2002.
6. Q. Zhao, S. S. Bhowmick, M. Mohania, and Y. Kambayashi. Discovering frequently changing structures from historical structural deltas of unordered XML. In *Proc. ACM CIKM*, pages 188–198, 2004.