Complete Discovery of Weighted Frequent Subtrees in Tree-Structured Datasets

Rahman AliMohammadzadeh , Ashkan Zarnani, Masoud Rahgozar, and Mostafa H. Chehreghani

Database Research Group, Control and Intelligent Processing Center Of Excellence, Faculty of ECE,
School of Engineering, University of Tehran, Tehran, Iran

Summary
Mining frequent subtree patterns has many useful applications in XML mining, bioinformatics, network routing, etc. Most of the frequent subtree mining algorithms (such as FREQT, TreeMiner and CMTreeMiner) use anti-monotone property in the phase of candidate subtree generation. However, none of these algorithms seems to verify the correctness of this property in all kinds of tree structured data. In this paper, we investigate the correctness of anti-monotone property for the problem of weighted frequent subtree mining and use multiple examples to elaborate on our finding about this issue. It is shown that anti-monotonicity does not generally hold, when using weighed support in tree pattern discovery. Consequently, the tree mining algorithms that are based on this property would miss some of the valid frequent subtree patterns if one considers weighted support. We propose also a novel algorithm named W3-Miner that uses new data structures and techniques for full extraction of frequent subtrees. The experimental results confirm that W3-Miner finds some frequent subtrees that the previously proposed algorithms would not discover when if weighted support is required.

Key words:
Semi-Structured Data Mining, Anti-Monotone Property, Frequent Subtree Mining

Introduction

Mining frequent subtrees has many practical applications in areas such as computer networks, Web mining, bioinformatics, XML document mining, etc [2, 5]. These applications share a requirement for the more expressive power of labeled trees to capture the complex relations among data entities. Frequent subtree mining is a more complex task compared to frequent item-set mining. However most of existing frequent subtree mining algorithms borrows techniques from the relatively mature association rule mining area [1, 9]. So far, many algorithms have been developed for mining frequent subtrees from a collection of trees such as XML documents. We developed W3-Miner [14] for discovering weighted embedded subtrees from a collection of trees. In [2, 5, 11] M.J. Zaki presented an algorithm, TreeMiner, to discover all frequent embedded subtrees, i.e., those subtrees that preserve ancestor-descendant relationships, in a forest or a database of rooted ordered

trees. This algorithm used a new data structure, scope-list, to efficiently count the frequency of candidate subtrees. The algorithm was further extended in [6] to build a structural classifier for XML data. Asai *et al.* in [4] presented an algorithm, FREQT, to find frequent rooted ordered subtrees. Also two algorithms were proposed by Asai et al. and Yun Chi et al. to mine rooted unordered subtrees, based on enumeration graph and enumeration tree data structures [7, 8]. Another work has been done in [3] where a model-validating approach for non-redundant candidate generation has been proposed. Wang and Liu [25] developed an algorithm to mine frequently occurring subtrees in XML documents. They mine induced subtrees only.
There are several other recent algorithms that mine different types of tree patterns, which include FreeTreeMiner [20] which mines induced, unordered, free trees (i.e., there is no distinct root); FreeTreeMiner for graphs [19] for extracting free trees in a graph database; and PathJoin [21], uFreqt [16], uNot [7], and HybridTreeMiner [22], which mine induced, unordered trees. TreeFinder [17] uses an Inductive Logic Programming approach to mine unordered, embedded subtrees, but it is not a complete method, i.e., it can miss many frequent subtrees, especially as support is lowered or when the different trees in the database have common node labels. SingleTreeMining [18] is another algorithm for mining rooted, unordered trees, with application to phylogenetic tree pattern mining. Recently, XSpanner [23], a pattern growth-based method, has been proposed for mining embedded ordered subtrees.
Almost all of these methods are based on the well-known apriori algorithm and have used anti-monotone property for candidate generation. This property suggests that the frequency of a super-pattern is less than or equal to the frequency of a sub-pattern. However, none of these algorithms have verified the correctness of anti-monotone property in tree structured data when considering weighted support.
In this paper, we investigate the correctness of anti-monotone property in discovering frequent subtrees when

Manuscript received August 25, 2006.
Manuscript revised August 30 , 2006.

considering weighted support. When the frequency of a subtree is based on weighted support, the previously proposed algorithms would probably miss some of the frequent subtrees. The reason is that the anti-monotone property does not necessarily hold in all kinds of tree structured data. To ensure complete discovery of all possible frequent subtrees, we propose a new algorithm, named W3-Miner. In W3-Miner a new method is used to count the support of a candidate subtree. In addition, a new join method is applied in the candidate generation phase. These improvements will guarantee the discovery of all of the valid frequent subtrees in a forest.

W3-Miner is an extension of the well-known TreeMiner [2, 5] algorithm to mine weighted frequent subtrees. For complete generation of k-subtree candidates, we extend the concept of scope-list data structure [2, 5] by adding a new component, called *RootPath.* Also, a new join method is applied for k-subtree candidate generation. By means of multiple examples, the incorrectness of anti-monotone property and the solution proposed by our algorithm are fully demonstrated. We also compare W3-Miner with three other tree mining algorithms. The obtained results confirm that some frequent subtree patterns are only discovered by W3-Miner.

This paper is organized as follows. In section 2 some preliminaries and definitions are given. Section 3 describes the anti-monotone property in tree structured data. The extended scope-list is provided in section 4. Section 5 describes the details of the proposed algorithm. We empirically evaluate the effectiveness of the algorithm in section 6 and the paper is concluded in section 7.

2. Preliminaries and Definitions

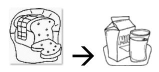
2.1 Association Rule Mining

Association rules were first introduced by Agrawal *et al.* [15] to analyze customer habits in retail databases. Association rule is an implication of the form X → Y, where the rule body X and head Y are subsets of the set I of items (I = {I1, I2,…, In}) within a set of transactions D and $X \cap Y = \varphi$. A rule $X \Rightarrow Y$ states that the transactions T that contain the items in X are likely to contain also the items in Y. Association rules are characterized by two measures: The support, which measures the percentage of transactions in D that contain both items X and Y; The confidence, which measures the percentage of transactions in D containing the items X that also contain the items Y [Figure 1].

In tree structured data context (XML document), both D and I are collections of trees [1], in the same way X and Y are trees (XML fragments) (Figure 2).

There are several algorithms for discovering association rules from XML documents. Braga et al. introduced XMINE operator for extracting association rules from XML documents [27]. Feng et al. proposed a theoretical framework for XML-Enabled association rule mining from XML documents [28]. An efficient model for discovering Association Rules from XML Documents are also developed in [13, 26] based on mining frequent subtree patterns. The same model can be used with appropriate modifications to mine XML association rules with weighted support using the algorithm proposed in here.

All of the traditional association rule mining algorithms and almost all of the XML association rule mining algorithms are using a property named anti-monotone to efficiently generate the candidate of size k+1 from the candidates of size k. In the next section, we will briefly review the definition of anti-monotone and anti-monotonicity.



[Support = 2%, Confidence = 95%]

Fig 1. Association rule between bread and milk

<Author> Rakesh Agrawal </author>

↓

<Keyword>Data Mining</keyword>

Fig 2. XML Association rule

2.2 Anti-Monotone Property and Anti-monotonicity

In mathematics, functions between ordered sets are monotonic (or monotone, or even isotone) if they preserve the given order [30]. These functions first arose in calculus and were later generalized to the more abstract setting of order theory. Although the concepts generally agree, the two disciplines have developed a slightly different terminology. While in calculus, one often talks about functions being monotonically increasing and monotonically decreasing, order theory prefers the terms monotone and antitone or order-preserving and order-reversing, respectively [30]. The anti-monotone property between frequent itemsets says that the support or frequency of itemsets is a monotonically decreasing function. This property helps traditional frequent itemset mining and association rules mining algorithms to efficiently generate the frequent itemsets and also prune the infrequent itemsets. This property is used in many frequent subtree mining algorithms such as TreeMiner [2], X3-Miner [3], FREQT [7], uFreqt [16], TreeFinder [17], Singletree Mining [18]. In these algorithms following the anti-monotone property in the generation of candidate frequent subtrees consisting of (k+1) nodes the infrequent subtrees with k nodes are not considered.

The anti-monotone property can be used in a condition which there is a non-increasing relation between the frequency of a pattern and the count of its elements. This condition is show in Figure 3, Where the frequency of the patterns never increase as the pattern grows.

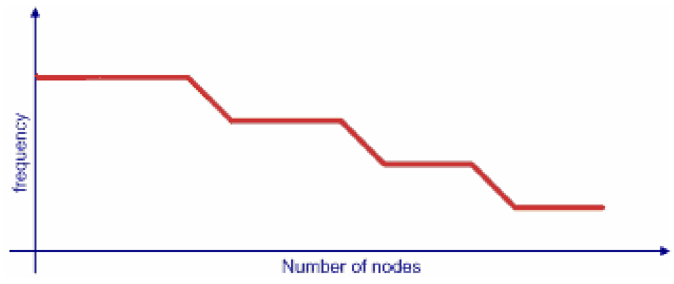


Fig. 3 Non-increasing relation between the frequency of a pattern and the count of its elements

In the following sections, we investigate the correctness of anti-monotone property for the problem of weighted frequent subtree mining.

2.3 Problem Definition

To explain the problem of mining frequent subtrees in a forest we provide the following definitions [1, 2 and 5]:

Definition 1. A rooted, labeled, tree, $T=(V,E)$ is a directed, acyclic, connected graph with $V=\{0,1,\ldots n\}$ as the set of vertices and $E=\{(x,y)\mid x,y\in V\}$ as the set of edges. One distinguished vertex $r\in V$ is selected as the root, and for all $x\in V$, there is a unique path from r to x. Further, $l:V\rightarrow L$ is a labeling function mapping vertices to a set of labels $L=\{l_1,l_2,\ldots\}$.

Definition 2. A tree T' with vertex set V' and edge set E' is an induced subtree of T if and only if (1) $V'\subseteq V$, (2) $E'\subseteq E$, (3) the labeling of V' is preserved in T', (4) $(v_1,v_2)\in E'$, where v_1 is the parent of v_2 in T', only if v_1 is a parent of v_2 in T. (5) if defined for rooted ordered trees, the left-to-right ordering among the siblings in T' should be a sub-ordering of the corresponding vertices in T.

Table 1. The frequencies of 1-subtrees in T1

a	b	c	d	e
$\sigma(a)=1$ $\sigma_w(a)=1$	$\sigma(b)=1$ $\sigma_w(b)=1$	$\sigma(c)=1$ $\sigma_w(c)=2$	$\sigma(d)=1$ $\sigma_w(d)=2$	$\sigma(e)=1$ $\sigma_w(e)=1$

Definition 3. For a rooted unordered tree T with vertex set V, edge set E, and no labels on the edges, a tree T' with vertex set V', edge set E', and no labels on the edges, is an *embedded subtree* of T if and only if (1) $V'\subseteq V$, (2) the labeling of the nodes of V' in T is preserved in T' and (3) $(v_1,v_2)\in E'$, where v_1 is the parent of v_2 in T', only if v_1 is an ancestor of v_2 in T. If T and T' are rooted ordered trees, then for T' to be an embedded subtree of T, a fourth condition must hold: (4) for $v_1,v_2\in V'$, preorder(v_1) < preorder(v_2) in T' if and only if preorder(v_1) < preorder(v_2) in T.

Definition 4. Let $\delta_T(S)$ indicate the number of occurrences of the subtree S in a tree T. Let d_T be an indicator variable, with $d_T(S)=1$ if $\delta_T(S)>0$ and $d_T(S)=0$ if $\delta_T(S)=0$. Let D denote a database of trees. The support of a subtree S in the database is defined as $\sigma(S)=\sum_{T\in D} d_T(S)$. The weighted support of S is defined as $\sigma_w(S)=\sum_{T\in D}\delta_T(S)$. Support is given as a percentage of the total number of trees in D. Some papers use *Frequency* instead of *Support*. In this paper, we use *Support* and *Frequency* interchangeably. Intuitively, the support is equal to 1 if the subtree exists in the tree otherwise it is equal to 0. Weighted support is equal to the exact number of occurrences of subtree S in a tree T.

Definition 5. An *l*-subtree *S*, which is a subtree with *l* nodes, is frequent if its (weighted) support is more than or equal to a user-specified minimum (weighted) support value.

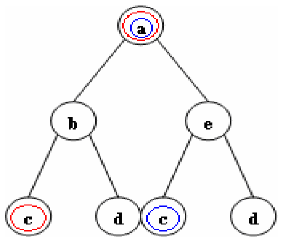


Fig. 4 A Sample Tree

Consider a sample tree (T1) depicted in Figure 4, following tables (Table 1 – Table 4) display some sample subtrees and explore the differences between support ($\sigma(S)$) and weighted support ($\sigma_w(S)$) in context of tree structured data.

As can be seen in Figure 4, the tree T1 has 5 nodes (1-subtrees). For some 1-subtrees (single nodes) the support and weighted support are equal. For example both support and weighted support of node 'a' are equal to 1. However, these values are not the same for node'd', which its weighted support is equal to 2 as it occurs two times in T1. The same case holds for node 'c'.

Table 2. Some Sample Induced 2-Subtrees and their frequencies in T1

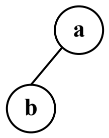
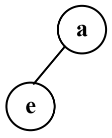
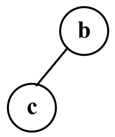
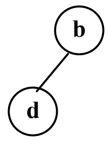
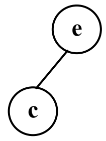
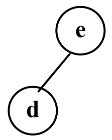
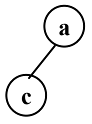
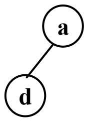
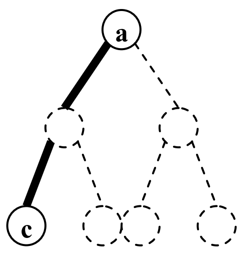
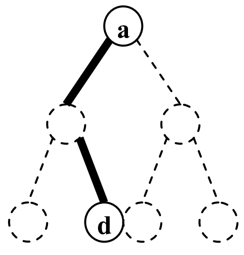
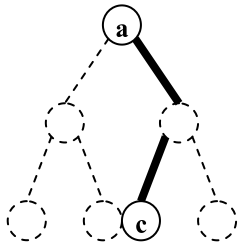
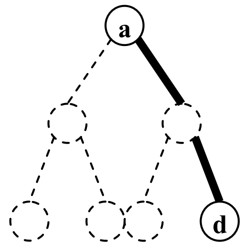
 $\sigma(a \xrightarrow{Induced} b) = 1$ $\sigma_w(a \xrightarrow{Induced} b) = 1$	 $\sigma(a \xrightarrow{Induced} e) = 1$ $\sigma_w(a \xrightarrow{Induced} e) = 1$	 $\sigma(b \xrightarrow{Induced} c) = 1$ $\sigma_w(b \xrightarrow{Induced} c) = 1$
 $\sigma(b \xrightarrow{Induced} d) = 1$ $\sigma_w(b \xrightarrow{Induced} d) = 1$	 $\sigma(e \xrightarrow{Induced} c) = 1$ $\sigma_w(e \xrightarrow{Induced} c) = 1$	 $\sigma(e \xrightarrow{Induced} d) = 1$ $\sigma_w(e \xrightarrow{Induced} d) = 1$

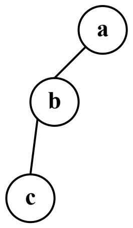
Table 2 displays some induced 2-subtree. As definition 2 suggests, all of the nodes in each of these induced subtrees are directly connected to each other. In contrast to table 1 in this table, the support and weighted support of these induced subtrees are equal. For example, the number of occurrences of 2-subtree 'a→e' in tree T1 is equal to 1. But again in table 3 these values become different. Consider the two embedded 2-subtrees displayed in the first row of this table. In these subtrees nodes 'b' and 'e' have been eliminated considering embedded relationship defined in Definition 3. As can be seen, the embedded subtree 'a→c' does exist in tree T1 so its support measure is equal to 1. But the number of occurrences of subtree 'a→c' in tree T1 is equal to 2. The first and second instances of 'a→c' are depicted in the second and third row, respectively. The embedded subtree 'a→d' has the same properties and its occurrences are shown in two different rows. The interesting fact in these samples is that the weighted supports of 2-subtree 'a→c' and 'a→d' are equal to 2 and the weighted support of node 'a' is equal to 1. Clearly 1-subtree 'a' is a sub-pattern of 2-subtree 'a→c'. Consequently in contrast to the famous anti-monotonicity principal, the super-pattern in this case has a greater frequency compared to its sub-pattern. This is where the contradiction to the anti-monotone property occurs and makes weighted support subtree mining a rather different challenge of large item set discovery problem.

Table 3. Two sample of embedded 2-Subtree and their frequencies in T1

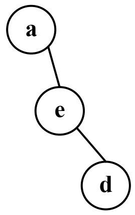
	
	
	
$\sigma(a \xrightarrow{Embedded} c) = 1$ $\sigma_w(a \xrightarrow{Embedded} c) = 2$	$\sigma(a \xrightarrow{Embedded} d) = 1$ $\sigma_w(a \xrightarrow{Embedded} d) = 2$

In Table 4, two induced 3-subtrees and one embedded 3-subtree are displayed. As can be seen, both support and weighted support of induced subtrees are equal to 1. But the weighted support of embedded subtree 'a→ (c, d)' is equal to 2.

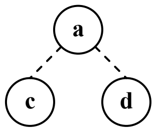
Table 4. Some sample of 3-Subtrees in T1 and their support values



$$\sigma(a \xrightarrow{Induced} b \xrightarrow{Induced} c) = 1$$
$$\sigma_w(a \xrightarrow{Induced} b \xrightarrow{Induced} c) = 1$$



$$\sigma(a \xrightarrow{Induced} e \xrightarrow{Induced} d) = 1$$
$$\sigma_w(a \xrightarrow{Induced} e \xrightarrow{Induced} d) = 1$$



$$\sigma(a \xrightarrow{Embedded} <c,d>) = 1$$
$$\sigma_w(a \xrightarrow{Embedded} <c,d>) = 4$$

The problem of mining frequent tree patterns in a forest of tree-structures transactions is to find all of the frequent k-subtrees, $1 \le k \le M$ where M is the maximum number of nodes in transactions. The desired type of frequent subtree patterns which is aimed in the mining process can differ based on the kind of application. In this paper, our goal is to generally mine all frequent, labeled, ordered, and embedded subtrees in a forest using *weighted support*, by proposing the W3-Miner algorithm.

3. Anti-Monotone Property in Tree Structured Data

Anti-monotone property says that the frequency of a super-pattern is less than or equal to the frequency of a sub-pattern. In this section we show that anti-monotone property does not hold in tree patterns when using weighed support. As a result, tree mining algorithms based on this property are unable to find all of the frequent tree patterns from a collection of trees.
An example of this case is shown in Figure 5, where the frequency of 1-subtree '*a*' is equal to 1 but the frequency of 2-subtree '*a-c'* is equal to 2.

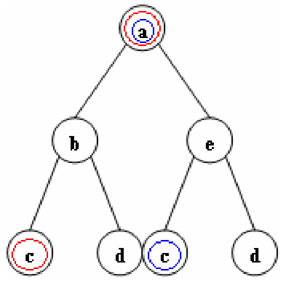


Fig. 5 Non-frequent subtree is in root

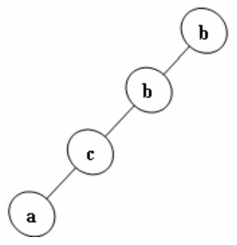


Fig. 6 Non-frequent subtree is in leaf

Figure 5 shows the state where the non-frequent subtree is placed in a higher level with respect to the frequent subtree of the transaction. An example of the other state that the non-frequent subtree is placed in a lower level with respect to the frequent subtree is depicted in Figure 6. As it is shown in the figure the frequency of 2-subtrees *'b-c'* and *'b-a'* are equal to 2 but the frequency of 1-subtree '*a*' and '*c*' are 1. Consequently we suggest the following proposition:

Proposition 1. Anti-monotone property does not hold in frequent tree mining when using weighted support.

4. Extended scope-list

We propose a new data structure named extended scope-list that will be exploited by W3-Miner. M.J. Zaki in TreeMiner algorithm [2, 5] introduced a new data structure called scope-list. Scope-list is generated for each candidate subtree *c* and is used to efficiently count its frequency. In scope-list of *c,* each element is a triple (t, m, s) , where *t* is a tree id in which c occurs, *m* is a match label of the k-1 length prefix of *c* in its string representation format, and *s* is the scope of the last node of *c*. The match label gives the positions of nodes in transaction tree (t) that match the prefix [2]. *S* is the scope of the right-most node of *t*. The scope of a node determines the range of vertices under that node. We extend the definition of scope-list by adding a new component, *RootPath,* to its elements. RootPath is an array of tuples *(x ,y)*, where *x* is the label of a node and *y* is preorder number of that node in transaction tree (*t*). *(x, y)* is generated for the root of the transaction tree (*t*) and all nodes between it and the root of the candidate tree (*c*) but not for the root of *c* itself. Figure 7 shows an example of extended scope list.

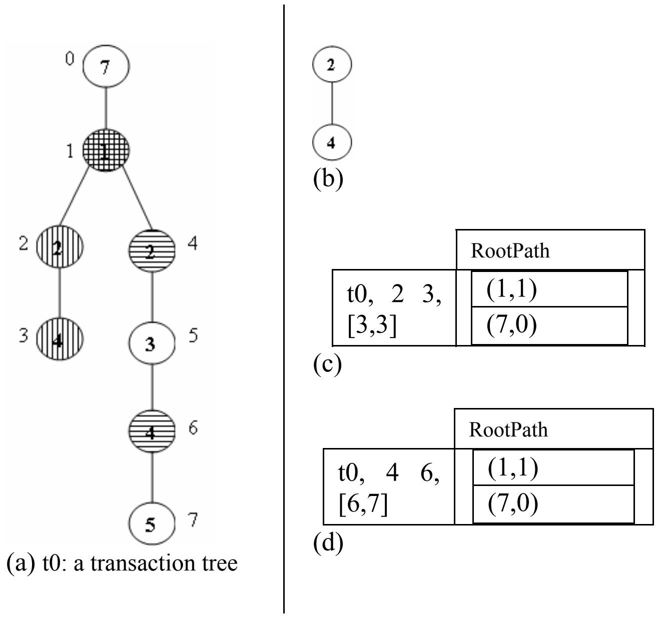


Fig. 7 (a) a sample transaction tree, (b) a sample candidate subtree which has 2 instances in t0, one of them is indicated by vertical lines and another is indicated by horizontal lines , (c) the extended scope-list of the first instance (marked with vertical lines) and (d) the extended scope-list of the second instance.

5. W3-Miner Algorithm

W3-Miner uses TreeMiner algorithm to generate *k+1* candidate subtrees from frequent *k* subtrees. A join operation is applied on the generated candidates to construct their extended scope-lists. Then the algorithm trims non-frequent *k+1* subtrees by using in-scope and out-scope tests. For more details about this process, the interested reader can refer to [2, 5]. To generate the *k+1* candidate subtrees that are missed by TreeMiner algorithm when using weighted support, W3-Miner joins a 1-tree with a k-tree in two steps as follows.

5.1 Extending Candidate Subtrees with RootPath Elements

For each element *e* in the extended scope-list of k-frequent subtree *k* and for each tuple in the RootPath array of *e*, the node of that tuple is joined to *k* by considering it as the root of *k*. For each obtained frequent (*k+1*)-subtree we generate its extended scope-list called *h,* by first copying extended scope-list of *k*. Then for each element *e* in *h* and for each element *r* in RootPath of *e,* we append the preorder number of *r* to the beginning of *e*'s match label. After this, the RootPath of each element in *h* is updated as follows. Each element *r* in RootPath of *e* in *h* is deleted if its preorder number *y* is greater or equal to the number appended to the match label of *e*. In Figure 8 node 1 is added to the root of candidate tree shown in Figure 7.

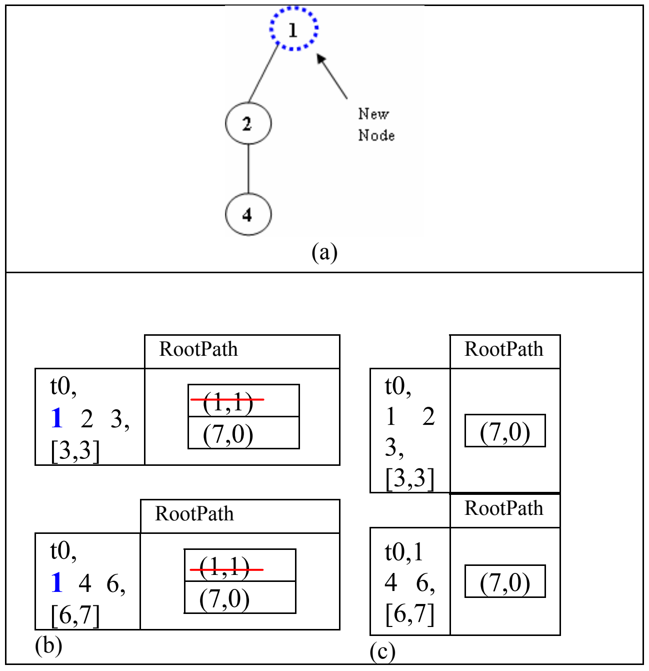


Fig. **8** A new node is added to the root of tree. (a) The new candidate subtree. (b) Updating scope-list (c) the final extended scope-list of the new subtree.

5.2 Extending candidate subtrees by using $\bar{F}_1$ elements

$\bar{F}_1$ is an array of triples (*x, y, z*), where *x* is the label of non-frequent node and *y* is number of that node in preorder traversal and *z* is the preorder number of last node in tree rooted by *x*. $\bar{F}_1$ contains all of the non-frequent nodes. After generating (k+1)-trees by Step 1 of W3-Miner, it is possible that some frequent (k+1)-Trees have not been generated yet (consider Figure 6). To solve this problem each node in $\bar{F}_1$ is added to the last node of frequent k-subtree *k*, if its scope is a proper subset of the scope of the last node of *k*.
We say that scope s_y is proper subset of scope s_x if and only if $l_x \le l_y$ and $u_x \ge u_y$, where *l* indicates the lower bound of a scope and *u* is its upper bound. We append the lower bound of each element of extended scope-list to its match label and the lower bound of the scope is set to *y* and the upper bound is set to *z*.

6. Experimental Results

6.1 Experiment 1: A Simple Forest

M.J. Zaki developed three variants of TreeMiner in [2, 5, 10 and 11]: VTreeMiner, HTreeMiner and TreeMinerD. In the current work we compare our proposed algorithm (W3-Miner) with TreeMinerD in terms of discovered frequent subtrees. Our sample input forest is shown in Figure 9. This forest consists of three (tree-structured) transactions. The total number of nodes is 32 and the number of distinct nodes is 9. We tested these algorithms with minimum weighted support being equal to 3. The results are presented in Table 1.

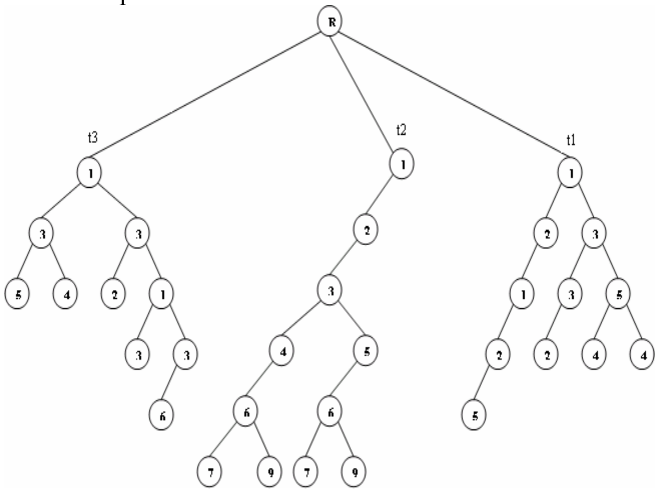


Fig. **9** Sample input forest containing 3 transactions

Table 5 Comparison between results of three different algorithms

TreeMinerD	W3-Miner
F1	**F1**
1 - 3	1 - **5**
2 - 3	2 - **5**
3 - 3	3 - **7**
4 - 3	4 - **4**
5 - 3	5 - **4**
	6 - 3
F2	**F2**
1 2 - 3	1 2 - **6**
1 3 - 3	1 3 - **9**
1 4 - 3	1 4 - **4**
1 5 - 3	1 5 - **5**
3 4 - 3	**1 6 - 4**
3 5 - 3	**2 5 - 3**
	3 2 - 3
	3 3 - 3
	3 4 - **4**
	3 5 - 3
	3 6 - 4

In column 1 of Table 5 the frequent subtrees discovered by TreeMinerD are displayed. However, as can be seen in column 2, W3-Miner discovers one frequent 1-subtree and five frequent 2-subtrees (shown in bold) that are missed by the TreeMinerD algorithm. The different discovered items are shown in bold. Actually, there are some frequent subtrees of size 3, 4, 5, …, 8 that are not shown in this table because of space limitations. The weighted support of subtree '2 5' is equal to 3 (1 instance in t2 and 2 instances in t1) thus must be considered as a valid frequent subtree. Also subtree '1 2 -1 3' has four instances in t1 and two instances in t3 making its weighted support equal to 6. W3-Miner was able to discover this frequent trees but the TreeMinerD was not.
We believe that these frequent patterns can be of high importance in many applications such as RNA structure mining and phylogenetic tree analysis [2, 12].

6.2 Experiment 2: Synthetic Dataset

We tested our algorithm on a synthetic dataset which is generated by the ToXgene XML Generator tool. ToXgene is a template-based generator for large, consistent collections of synthetic XML documents, developed as part of the ToX (the Toronto XML Server) project [29]. Our generated data set consists of 100 trees with the maximum level of 3. Each node has a maximum of 6 children. As a measure of repeated nodes there are at most 3 nodes with same labels among the children of a parent node.
The TreeMinerD algorithm developed by Zaki [2] is used in our comparisons as the most recognized algorithm in this domain. Both of the algorithms were run with support values between 2 and 50 to verify the number of discovered frequent subtrees with different support thresholds. As can be seen in Figure 10 the TreeMinerD algorithm will miss many of the frequent subtree patterns because it does not consider weighted support properly (see section 3).

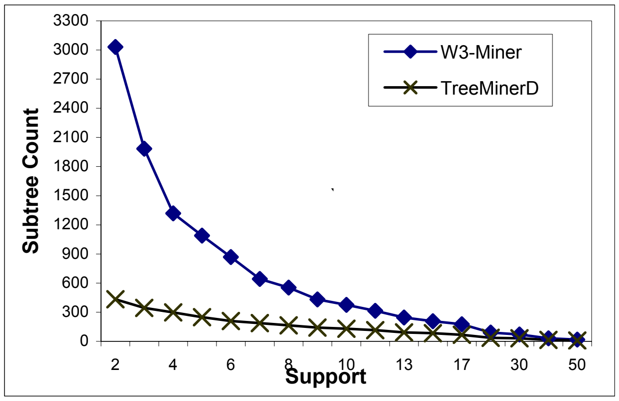


Fig. 10 Comparison between W3-Miner and TreeMinerD

The number of missed patterns increases as the support threshold value decreases. Of course, it should be noted that the number of extra discovered patterns in W3-Miner largely depends on the properties of the data set in hand. More accurately the both algorithms will discover the same set of patterns when there are not any nodes with the same label in each tree of the forest. In fact, the more the number of repeated nodes exists in the trees, the higher will be the difference between the numbers of discovered pattern in the two algorithms.

7. Conclusions

In this paper we investigated the anti-monotone property in tree structured data when weighted support is required. We showed that this property does not hold in such context. Consequently we proposed a novel algorithm, W3-Miner, to find all of the weighted frequent subtree patterns in a database of trees. We extended the scope-list data structure by adding a new component, called RootPath, and applied a new candidate generation procedure on this data structure. In each stage of this two step procedure, we cover a set of candidate subtrees that would not be considered by other algorithms (i.e. HTreeMiner, VTreeMiner and FREQT). The experimental results confirmed that W3-Miner can find some frequent subtrees missed by other algorithms.
The next step to the current work will be to conduct a performance comparison study on W3-Miner and other frequent subtree mining algorithms. We believe that a good future research direction is investigating the application of weighted frequent subtree mining and W3-Miner in real application areas such as RNA structure mining and web mining.

References

1. Y. Chi, S. Nijssen, R.R. Muntz, J. N. Kok, "Frequent Subtree Mining An Overview," Fundamental Informatics, Special Issue on Graph and Tree Mining, 2005.
2. M.J. Zaki, "Efficiently Mining Frequent Trees in a Forest: Algorithms and Applications," in *IEEE Transaction on Knowledge and Data Engineering*, vol. 17, no. 8, pp. 1021-1035, 2005.
3. H. Tan, T.S. Dillon, L. Feng, E. Chang, F. Hadzic, "X3-Miner: Mining Patterns from XML Database*," In Proc. Data Mining '05. Skiathos, Greece*, 2005.
4. K. Abe, S. Kawasoe, T. Asai, H. Arimura, and S. Arikawa, "Optimized Substructure Discovery for Semi-structured Data," *In Proc. PKDD'02*, 1–14, LNAI 2431, *2002.*
5. M. J Zaki,.. Efficient Mining of Trees in the Forest. SIGKDD '02, Edmonton, Alberta, Canada, ACM. 2002.
6. M. J. Zaki and C. C. Aggarwal. XRules: An effective structural classifier for XML data. In *Proc. of the 2003 Int. Conf. Knowledge Discovery and Data Mining*, 2003.
7. T. Asai, H. Arimura, T. Uno, and S. Nakano. Discovering frequent substructures in large unordered trees. In *Proc. of the 6th Intl. Conf. on Discovery Science*, 2003.
8. Y. Chi, Y. Yang, and R. R. Muntz. Mining frequent rooted trees and free trees using canonical forms. Technical Report CSD-TR No. 030043, UCLA, 2003.
9. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo, "Fast Discovery of Association Rules," Advances in Knowledge Discovery, and Data Mining, U. Fayyad et al., eds.,pp. 307-328, Menlo Park, Calif.: AAAI Press, 1996.
10. M.J. Zaki, "Fast Vertical Mining Using Diffsets," In. Proc. of Int. Conf. Knowledge Discovery and Data Mining (SIGKDD'03), 2003.
11. M. Zaki. Efficiently mining frequent embedded unordered trees. Fundamental Informatics, 65:1-20, 2005.
12. B. Shapiro and K. Zhang, "Comparing Multiple RNA Secondary Structures Using Tree Comparisons," Computer Applications in Biosciences, vol. 6, no. 4, pp. 309-318, 1990.
13. R. AliMohammadzadeh, S. Soltan, and M. Rahgozar, "Template guided association rule mining from XML documents". In Proceedings of the 15th international Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM Press, New York, NY, 963-964. DOI= http://doi.acm.org/10.1145/1135777.1135966.
14. R. AliMohammadzadeh, M. Haghir Chehreghani, A. Zarnani, M. Rahgozar, "W3-Miner: Mining Weighted Frequent Subtree Patterns in a Collection of Trees". In Proceedings of the Second International Conference on Pattern Analysis (Budapest, Hungary, May 26-28, 2006). ICPA'06. Transaction on Engineering, Computing and Technology, ISSN 1305-5313, Pages 164-168*,* World Enformatika Society.
15. R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases". In Proceedings of the 1993 ACM SIGMOD international Conference on Management of Data (Washington, D.C., United States, May 25 - 28, 1993). P. Buneman and S. Jajodia, Eds. SIGMOD '93. ACM Press, New York, NY, 207-216.
16. S. Nijssen and J.N. Kok, "Efficient Discovery of Frequent Unordered Trees," Proc. First Int'l Workshop Mining Graphs, Trees, and Sequences, 2003.
17. A. Termier, M-C. Rousset, and M. Sebag, "Treefinder: A First Step Towards XML Data Mining," Proc. IEEE Int'l Conf. Data Mining, 2002.
18. D. Shasha, J. Wang, and S. Zhang, "Unordered Tree Mining with Applications to Phylogeny," Proc. Int'l Conf. Data Eng., 2004.
19. U. Ruckert and S. Kramer, "Frequent Free Tree Discovery in Graph Data," Special Track on Data Mining, Proc. ACM Symp. Applied Computing, 2004.
20. Y. Chi, Y. Yang, and R.R. Muntz, "Indexing and Mining Free Trees," Proc. Third IEEE Int'l Conf. Data Mining, 2003.
21. Y. Xiao, J.-F. Yao, Z. Li, and M.H. Dunham, "Efficient Data Mining for Maximal Frequent Subtrees," Proc. Int'l Conf. Data Mining,2003.
22. Y. Chi, Y. Yang, and R.R. Muntz, "HybridTreeMiner: An Efficient Algorihtm for Mining Frequent Rooted Trees and Free Trees Using Canonical Forms," Proc. 16th Int'l Conf. Scientific and Statistical Database Management, 2004.
23. C. Wang, M. Hong, J. Pei, H. Zhou, W. Wang, and B. Shi, "Efficient Pattern-Growth Methods for Frequent Tree Pattern

Mining," Proc. Pacific-Asia Conf. Knowledge Discovery and Data Mining, 2004.
25. K. Wang and H. Liu, "Discovering Typical Structures of Documents: A Road Map Approach" Proc. ACM SIGIR Conf. Information Retrieval, 1998.
26 R. AliMohammadzadeh, M. Rahgozar, "An Efficient Model for Discovering Association Rules from XML Documents". In Proceedings of the Third International Conference on Knowledge Mining (Prague, Czech Republic, August 25-27, 2006). ICKM'06. Transaction on Engineering, Computing and Technology, ISSN 1305-5313, Pages 164-168, World Enformatika Society.
27. Braga D., A. Campi, M. Klemettinen, and P. L. Lanzi. "Mining association rules from XML data". In Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery, September 4-6, Aixen-Provence, France 2002.
28. Feng L. & T. Dillon. "Mining XML-Enabled Association Rule with Templates". In Proceedings of KDID04, 2004.
29. Barbosa D., A. Mendelzon, J. Keenleyside and K. Lyons, "ToXgene: A template-based data generator for XML". In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Madison, Wisconsin - June 4-6, 2002.
30. Abramsky S., A Jung, "Domain Theory", Handbook of Logic in Computer Science, 1994.