International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:7, 2008

W3-Miner: Mining Weighted Frequent Subtree Patterns in a Collection of Trees

R. AliMohammadzadeh, M. Haghir Chehreghani, A. Zarnani, and M. Rahgozar

***Abstract*—**Mining frequent tree patterns have many useful applications in XML mining, bioinformatics, network routing, etc. Most of the frequent subtree mining algorithms (i.e. FREQT, TreeMiner and CMTreeMiner) use anti-monotone property in the phase of candidate subtree generation. However, none of these algorithms have verified the correctness of this property in tree structured data. In this research it is shown that anti-monotonicity does not generally hold, when using weighed support in tree pattern discovery. As a result, tree mining algorithms that are based on this property would probably miss some of the valid frequent subtree patterns in a collection of trees. In this paper, we investigate the correctness of anti-monotone property for the problem of weighted frequent subtree mining. In addition we propose W3-Miner, a new algorithm for full extraction of frequent subtrees. The experimental results confirm that W3-Miner finds some frequent subtrees that the previously proposed algorithms are not able to discover.

***Keywords*—**Semi-Structured Data Mining, Anti-Monotone Property, Trees.

I. 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. In [2, 5 and 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.

R. AliMohammadzadeh is with the Database Research Group, faculty of ECE, School of Engineering, University of Tehran, Tehran, Iran (e-mail: r.mohammadzadeh@ece.ut.ac.ir).

M. Haghir Chehreghani is with the Database Research Group, faculty of ECE, School of Engineering, University of Tehran, Tehran, Iran (e-mail: m.haghir@ece.ut.ac.ir).

A. Zarnani is with the Database Research Group, faculty of ECE, School of Engineering, University of Tehran, Tehran, Iran (e-mail: a.zarnani@ece.ut.ac.ir).

M. Rahgozar is with the Control and Intelligence Processing Center of Excellence, faculty of ECE, School of Engineering, University of Tehran, Tehran, Iran (e-mail: rahgozar@ut.ac.ir).

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. 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 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 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 many 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 II the tree mining problem statement is given. Section III describes the anti-monotone property in tree structured data. The extended scope-list is provided in section IV. Section V describes the

details of the proposed algorithm. We empirically evaluate the effectiveness of the algorithm in section VI and the paper is concluded in section VII.

II. Problem Statement

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 $V=\{0,1,...n\}$ 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 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 $L=\{l_1,l_2,...\}$.

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.

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.

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.

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.

III. 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 Fig. 1, where the frequency of 1-subtree '*a*' is equal to 1 but the frequency of 2-subtree '*a-c*' is equal to 2.

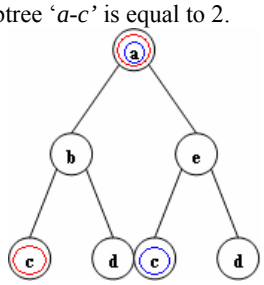

Fig. 1 Non-frequent subtree is in root

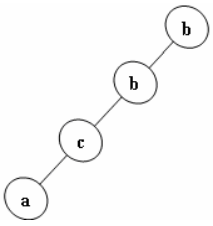

Fig. 2 Non-frequent subtree is in leaf

Fig. 1 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 Fig. 2. As can be seen 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.

IV. 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 I s 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]. 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 element. 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. Fig. 3 shows an example of extended scope list.

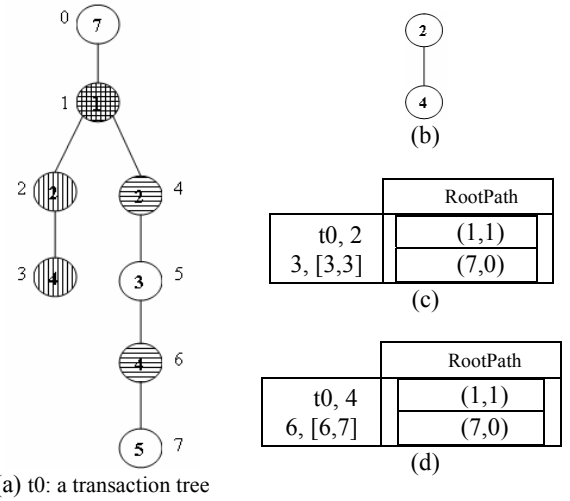

Fig. 3 (a) a sample transaction tree, (b) a sample candidate subtree which has 2 instances in t0, one of them is indicated by vertical lines and another is indicated by horizontal lines , (c) the extended scope-list of the first instance (marked with vertical lines) and (d) the extended scope-list of the second instance

V. 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.

A. 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 Fig. 4 node 1 is added to the root of candidate tree shown in Fig. 3.

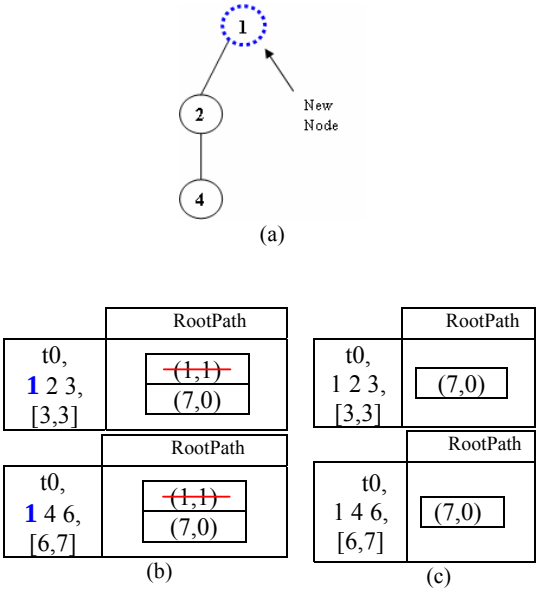

Fig. 4 A new node is added to the root of tree. (a) The new candidate subtree. (b) Updating scope-list (c) the final extended scope-list of the new subtree

B. 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 2). 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*.

VI. Experimental Results

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 HTreeMiner and VTreeMiner in terms of generated frequent subtrees. Our sample input forest is shown in Figure 5. 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 I.

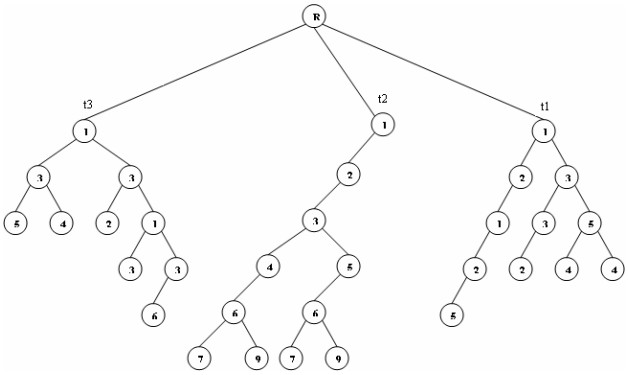

Fig. 5 Sample input forest containing 3 transactions

TABLE I
COMPARISON BETWEEN RESULTS OF THREE DIFFERENT ALGORITHMS

HTreeMiner	VTreeMiner	W3-Miner
F1	**F1**	**F1**
1 – 3	1 – 3	1 – **5**
2 – 3	2 – 3	2 – **5**
3 – 3	3 – 3	3 – **7**
4 – 3	4 – 3	4 – **4**
5 – 3	5 – 3	5 – **4**
		6 - 3
F2	**F2**	**F2**
1 2 – 3	1 2 – 3	1 2 – **6**
1 3 – 3	1 3 – 3	1 3 – **9**
1 4 – 3	1 4 – 3	1 4 – **4**
1 5 – 3	1 5 – 3	1 5 – **5**
3 4 – 3	3 4 – 3	3 4 – **4**
3 5 – 3	3 5 – 3	3 5 – 3
		1 6 -- 4
		2 5 – 3
		3 2 – 3
F3	**F3**	**F3**
1 3 4 – 3	1 3 4 – 3	1 3 4 – **4**
1 3 5 – 3	1 3 5 – 3	1 3 5 – 3
		1 3 2 – 3
		1 3 3 – 3
		1 3 6 –5
		1 2 5 – 4
		1 3 -1 3 – 5
		1 2 -1 3 – 6

In column 1 and 2 of Table I the frequent subtrees discovered by HTreeMiner and VTreeMiner are displayed respectively. The results obtained from these two algorithms are equal. However, as can be seen in column 3, W3-Miner discovers three frequent 2-subtrees and six frequent 3-subtrees (shown in bold) that are missed by the other algorithms. 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 can find this frequent tree but the other algorithms (i.e. HTreeMiner and VTreeMiner) can 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].

VII. CONCLUSION

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 this context. Consequently we proposed a novel algorithm, W3-Miner, to find all of the weighted frequent tree 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 tree mining algorithms. We are currently 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.