RSF - A New Tree Mining Approach with an Efficient Data Structure

Federico del Razo Lopez, Anne Laurent, Pascal Poncelet, Maguelonne Teisseire

RSF - A New Tree Mining Approach with an Efficient Data Structure

Federico Del Razo Lopez
LIRMM - CNRS
161 rue Ada
34392 Montpellier
FRANCE
delrazo@lirmm.fr

Anne Laurent
LIRMM - CNRS
161 rue Ada
34392 Montpellier
FRANCE
laurent@lirmm.fr

Pascal Poncelet
EMA - Site EERIE
69 rue G. Besse
30035 Nîmes
FRANCE
pascal.poncelet@ema.fr

Maguelonne Teisseire
LIRMM - CNRS
161 rue Ada
34392 Montpellier
FRANCE
teisseire@lirmm.fr

Abstract

Mining frequent subtrees from tree databases is currently a very active research topic. This research has many interests, including for instance mediator schema building from XML schemas. In this framework, many works have been proposed. However, the way the data are represented is often very memory-consuming. In this paper, we propose a new method to represent such data. We show that this representation has many properties, which can be used in order to enhance subtree mining algorithms. Experiments are led to assess our proposition (data representation and its associated algorithms).

Keywords: Data Mining, XML, Frequent Sub-Tree Mining.

1 Introduction

The volume of data available from the Internet is growing dramatically. Although it provides rich information, it raises many problems when querying these huge volumes of data in order to retrieve relevant information. Since users can indeed hardly be aware of the way the data are organized, it is necessary to provide them with mediator schemas that are built automatically. In this framework, a mediator schema is meant as an interface between the user and the data. This interface provides the user with a schema to query heterogeneous and distributed data in a very simple way, as shown on Figure 1.

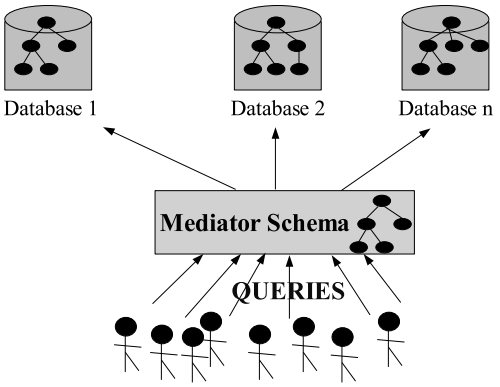

Figure 1: Querying Distributed Databases through a Mediator Schema

As highlighted by the World Wide Web Consortium, XML has been proposed to deal with huge volumes of electronic documents and is playing an increasing important role in the exchange of a wide variety of data on the Web. It is thus necessary to be provided with tools to integrate XML schemas in order to query XML data.

Even if recent works have been proposed to access data when a query schema is known [11], it is still a challenge to mine automatically such mediator schemas. Existing approaches from the literature are incomplete [8]. In this paper, we focus on frequent pattern mining from an XML database. In this framework, a frequent pattern is a subtree that occurs in *most* XML schemas from the database. This proportion is examined regarding a *support*, which corresponds to a minimal number of trees from the database that must contain the subtree in order to consider this subtree as being *frequent.* This mining process is complex since all the XML schemas from the database must be represented using a representation structure that can support data mining algorithms.

In many approaches, this transformation leads to memory-consuming representations requiring twice or three times the number of nodes. As far as we know, there is no work in the literature that provides both a compact representation and useful properties for data mining algorithms. In this paper, we thus propose a new structure being a compact representation as well as a relevant support for data mining, as some properties can be used when generating candidates and pruning non frequent candidates. This approach is compared to existing works [2, 7, 13] and we show its interest through experiments.

This paper is organized as follows. Section 2 presents related work. Section 3 presents the core of our proposition defining a new method to represent tree data and the associated algorithms for frequent subtree mining. Section 4 presents the results from experiments led on synthetic data. Finally, Section 5 concludes and presents some further work associated to our proposition.

2 Related Work

Mining frequent subtree has recently received a great deal of attention [2, 5, 7, 12, 13]. In the following, we first introduce preliminary definitions. Second we focus on principal approaches.

2.1 Preliminary Definitions

A *tree* is a connected graph containing no cycle. A tree is composed by nodes, which are linked by edges such that their exists a particular node called *root* and such that all the nodes but the root are composed by sub-trees. A tree is said to be an *ordered tree* if the children from a node are ordered. A tree is said to be an *unordered tree* otherwise.

Let $\mathcal{L} = \{a, b, c, ...\}$ be a set of labels. A *labeled ordered tree* is a tree $T = \{r, N, B, L, \prec_F\}$ where: r is the root, N is the set of nodes, B is the set of edges such that $B \subseteq V^2$, $(L : N \rightarrow \mathcal{L})$ is a mapping from the set of labels $\mathcal{L}$ to the set of nodes N, and $\prec_F$ is an ordered relation between brother nodes.

Several nodes can have the same label, which raises the polysemy problem, since a label can be associated to several nodes. The *size* of T is the number of nodes in T: $|T|$. The *order* of T is the number of edges in T.

Each node n is associated with a unique number i, corresponding to its position in a breadth-first traversal of the tree. n_i denotes the ith node in such a traversal ($i = 0, \ldots, |T| - 1$).

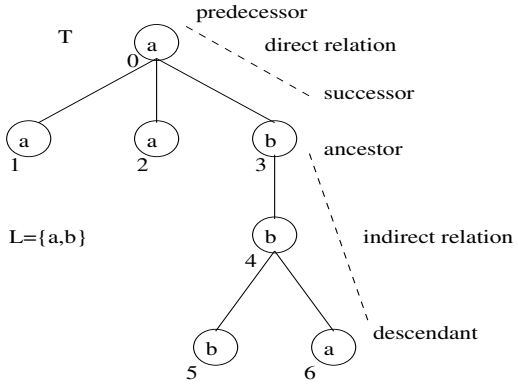

Figure 2: A Tree

For each pair $(x, y) \in N \times N$, it is said that there is a *direct* relation between x and y if there exists an edge from x to y. x is said to be the *predecessor* and y is said to be the *successor*. It is said that there is a *indirect* relation between x and y if there exists a path between x and y, meaning that there exists a non-empty set of nodes $\nu_1, \ldots, \nu_k$ such that there exists edges from x to ν_1, from ν_1 to ν_2, ... and from ν_k to y. In this case, x is said to be the *ancestor* and y is said to be the *descendant*.

Figure 2 shows a direct relation in T between the nodes 0 and 3, and an indirect relation between the nodes 3 and 6.

$S \preceq T$ denotes that S is a subtree of T (*cf* Figure 3). Considering an indirect link *ancestor-descendant*, $S \preceq T$ holds under the following conditions:

1. $N_S \subseteq N_T$
2. for each edge $b_S = (x, y) \in B_S$, x is an ancestor of y in T
3. for each edge $b_T = (x, y) \in B_T$, x is an ancestor of y in S
4. for each $b^1_S = (x, y_1) \in B_S$ and $b^2{}_S = (x, y_2) \in B_S$, $y_1 \prec_F y_2 \Rightarrow y_1 \prec_F y_2$ in T

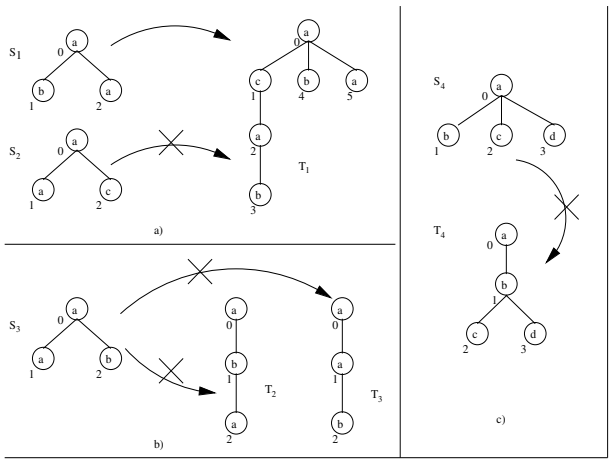

Figure 3: Inclusion and non inclusion of S in T

2.2 Mining Frequent Sub-trees

The approaches proposed in the literature for mining frequent subtrees are mostly based on the data representation they use in order to speed up the data mining algorithms.

The *TreeMiner* algorithm [13] proposes a method for frequent subtree discovery. As we propose here, this work is based on an original representation of the trees, which facilitates the management of the candidates. This approach needs a storage space of three times the size of each tree ($3|T|$), as shown by Figure 4.

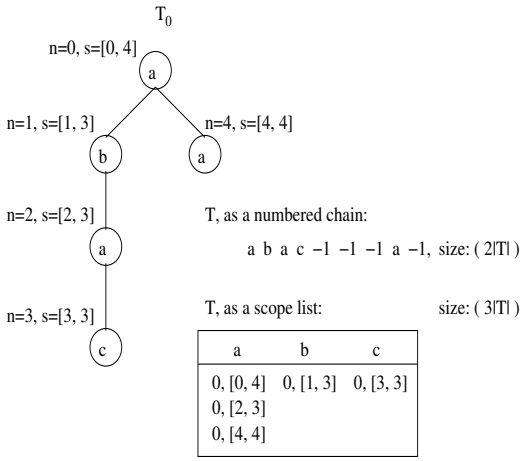

Figure 4: Data Structure proposed in [13]. *TreeMiner*

The *FreqT* approach proposed in [2] is devoted to ordered trees. The data structure that has been chosen leads to interesting results regarding runtime. However, this proposal is as memory-consuming as [13], *i.e.* $3|T|$, as shown by Figure 5.

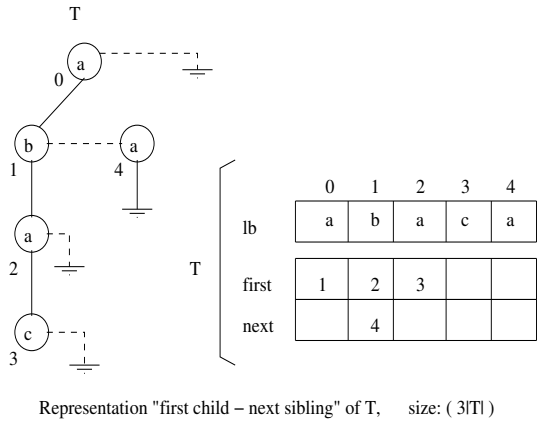

Figure 5: Data Structure proposed in [2]. *FreqT*

Although some other data structures have been proposed recently (*Chopper* [9], *FreeTreeMiner* [4] and *CMTreeMiner* [3]), they do not offer useful properties for the management of the candidates that could be as interesting as *TreeMiner* [13] and *FreqT* [2]. These representations are shown on Figures 6 and 7.

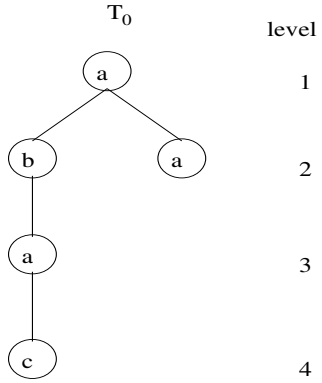

Figure 6: Data Structure proposed in [9]. *Chopper*

Our aim is thus to have both a data structure that is not memory-consuming and that has some good properties regarding data mining.

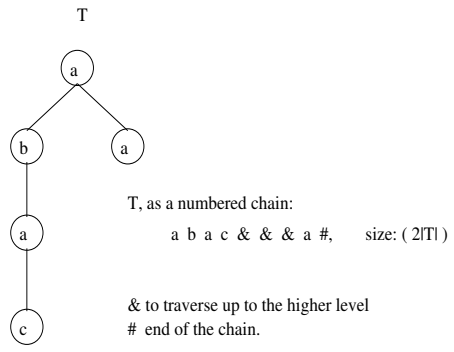

Figure 7: Data Structure proposed in [3, 4]. $FreeTreeMiner$ and $CMTreeMiner$.

3 Proposition

In this paper, we propose a new method for frequent subtree mining. Our proposition is based on an original representation of the trees which allows the efficient generation and pruning of candidate subtrees.

3.1 Tree Representation

When representing a tree T, we keep in mind the following property: all the nodes but the root have one and only one predecessor. We propose thus to use two vectors to represent a tree, as proposed in [10]. The first vector is denoted by st. It stores the position of each node predecessor. Nodes are numbered considering a depth-first traversal. The root is numbered as being at position 0, with $st[0] = -1$ since it has no predecessor. The values $st[i], i = 1, 2, ..., k-1$ correspond to all other predecessor positions, as shown on Figure 8.

This representation provides a constant-time method to retrieve the predecessor of a node. Moreover, it allows us to find directly the most right leaf when considering an index k. Finally, when visiting the tree, it is possible to build all direct links from predecessors to descendants.

The second vector is denoted by lb. It is used to store all the tree labels. $lb[i], i = 0, 1, ..., k-1$ are the labels of each node $n_i \in T$.

The data structure we have chosen needs very low memory since it is reduced to the size of $2|T|$. Moreover, it has good properties when mining frequent subtrees (see section 3.2).

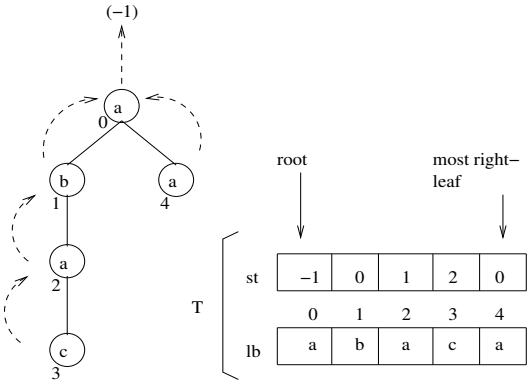

Figure 8: Representation of a Tree

3.2 Generating and Pruning Candidates

Candidates of size 1 (single nodes) are obtained by visiting all the nodes from the trees of the database. Each node is mapped to a support which is incremented during the traversal. Only the nodes having a support value greater than the minimum user-defined threshold are kept. The database is then transformed in order to delete all non frequent nodes, as shown on Figure 9.

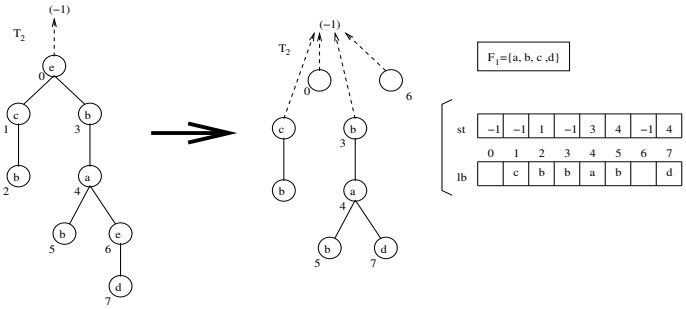

Figure 9: Transformation of the database after generation of F_1

Candidates of size 2 are generated by combining all the pairs of candidates of size 1.

The database is then updated by modifying the trees from the root, the nodes, and the leaves so that only frequent links are kept.

Figures 10 and 11 illustrate this process, with $\sigma = 7$ and $F_2 = \{a-d, a-b\}$.

The generation of candidates of size $k \geq 3$ is performed by a levelwise APriori-like algorithm [1],

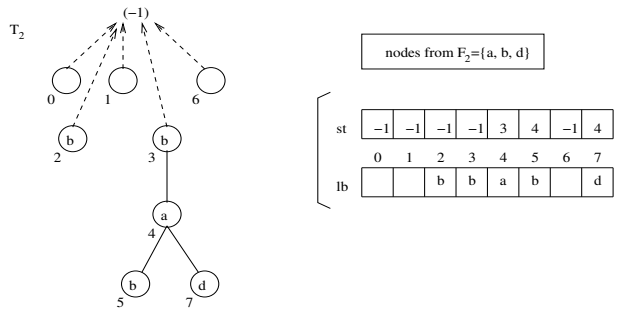

Figure 10: Transformation of the database - root

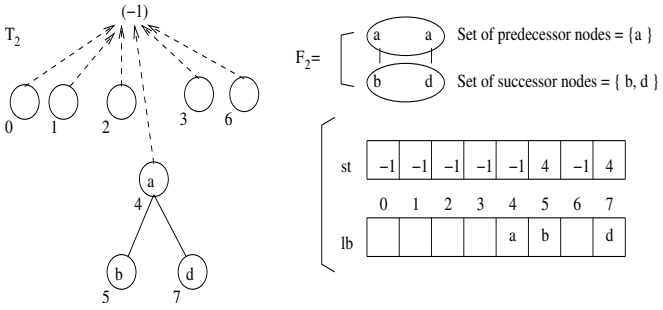

Figure 11: Transformation of the database - nodes

by combining frequent subtrees of size $k-1$.

The originality of our approach lies in the use of our representation for candidate pruning when deleting non frequent candidates. Computing the support of each candidate is performed by counting the number of trees which contain the candidate being considered. For this purpose, for each tree of the database, we aim at finding an *anchor node* on which the root of the subtree to be tested can be deployed. For each anchor point that can be found out, our method checks whether it is possible or not to deploy all the nodes. Please note that a *perfect* deployment is looked for. If all the nodes of the candidate tree can be found in a part of the tree from the database being considered, then this tree is counted and the support is incremented.

4 Experiments

Experiments are led on synthetic data obtained by the XML data generator developped by A. Termier [7]. Results, shown on Figures 12, 13 and 14, highlight the interest of our approach when considering scalability (runtime and memory) over the number of trees in the database. These results compare our approach (RSF) to *FreqT* (implementation provided by the authors) [2].

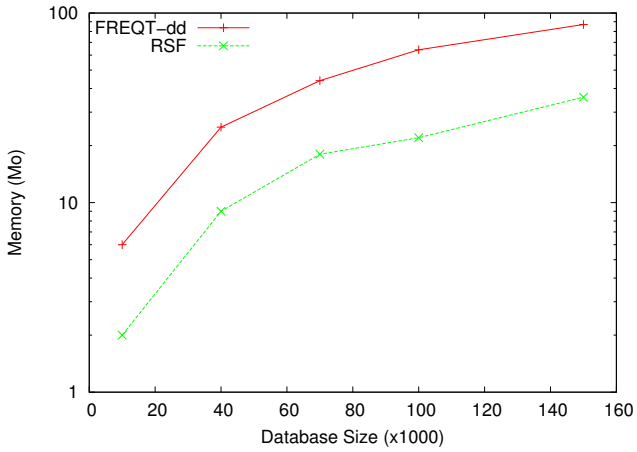

Figure 12: Memory over number of trees

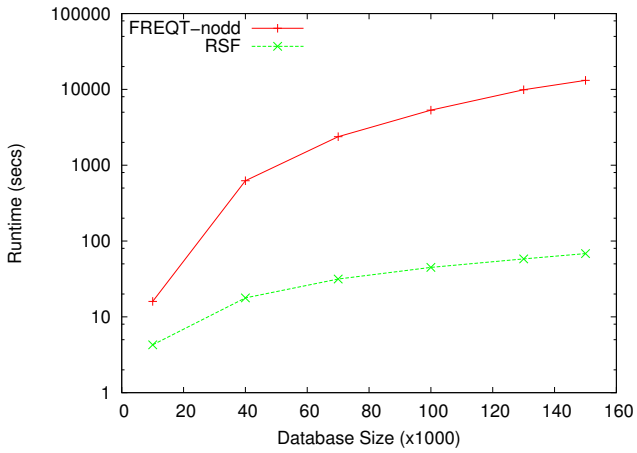

Figure 13: Runtime over database size, FreqT-nodd and RSF. $min\_sup = 0.05$

5 Conclusion and Perspective

In this paper, we propose a new approach for tree representation which provides good properties for efficient frequent subtree mining. Experiments led on synthetic data show very promising results compared to the works from the literature.

Our proposition can be applied in many domains, especially for data mediation. Frequent subtrees that are mined by our method can indeed help building a mediator schema. Such a solution can also be taken in the framework of on-line data

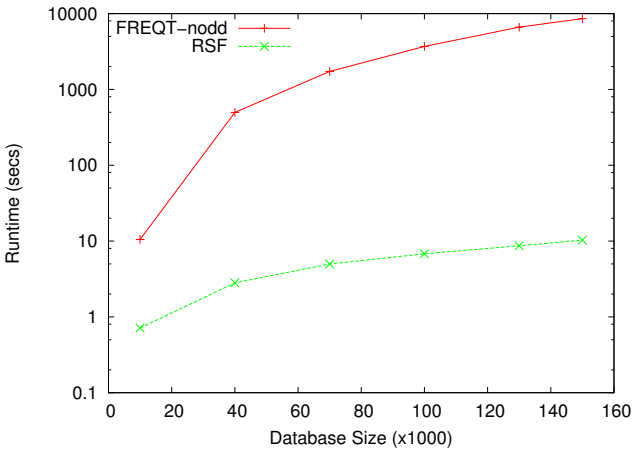

Figure 14: Runtime over database size, FreqT-nodd and RSF. $min\_sup = 0.5$

mining for data streams. This perspective is very promising since it provides efficient and fast methods to deal with high volumes of XML data on the Internet.

We also aim at softening our approach by using fuzzy logic when dealing with tree inclusion, as shown in [6]. In this framework, we argue that the data structure we present here has many properties that would be interesting in such a fuzzy context.

References

[1] R. Agrawal et R. Srikant, *Fast algorithms for mining association rules in large databases*, *in* Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994.

[2] T. Asai, K. Abe, S. Kawasoe, H. Arimura et H. Sakamoto, *Efficient substructure discovery from large semi-structure data*, *in* 2nd Annual SIAM Symposium on Data Mining, SDM2002, Arlington, VA, USA, 2002, Springer-Verlag.

[3] Y. Chi, Y. Yang et R. Muntz, *Cmtreeminer: Mining both closed and maximal frequent subtrees*, *in* The Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04), 2004.

[4] Y. Chi, Y. Yang et R. R. Muntz, *Indexing and mining free trees.*, *in* International Conference on Data Mining 2003 (ICDM2003), 2003.

[5] M. Kuramochi et G. Karypis, *Frequent subgraph discovery*, *in* IEEE International Conference on Data Mining (ICDM), 2001.

[6] A. Laurent, P. Poncelet et M. Teisseire, *Fuzzy data mining for the semantic web: Building XML mediator schemas*, *in* Fuzzy Logic and the Semantic Web, E. Sanchez, éd., Capturing Intelligence, ELSEVIER, to appear, 2005.

[7] A. Termier, M.-C. Rousset et M. Sebag, *Treefinder, a first step towards XML data mining*, *in* IEEE Conference on Data Mining (ICDM), 2002, p. 450–457.

[8] J. Tranier, R. Baraer, Z. Bellahsene et M. Teisseire, *Where's Charlie: Family based heuristics for peer-to-peer schema integration*, *in* Proceedings of the 8th International Database Engineering and Applications Symposium (IDEAS '04), Coimbra, Portugal, July, 7th - 9th 2004.

[9] C. Wang, Q. Yuan, H. Zhou, W. Wang et B. Shi, *Chopper: An efficient algorithm for tree mining*, Journal of Computer Science and Technology, 19 (May 2004), p. 309–319.

[10] M. A. Weiss, *Data Structures And Algorithm Analysis In C*, Addison Wesley, 1998.

[11] L. Xyleme, *A dynamic warehouse for xml data of the web*, *in* IEEE Data Engineering Bulletin, 2001.

[12] X. Yan et J. Han, *gspan: Graph-based substructure pattern mining*, *in* IEEE Conference on Data Mining (ICDM), 2002.

[13] M. Zaki, *Efficiently mining frequent trees in a forest*, *in* ACM-SIGKDD'02, 2002.