# Efficient structural similarity computation between XML documents

**Ali Aïtelhadj**
**Computer Science Department, Faculty of Electrical Engineering and Computer Science**
**Mouloud Mammeri University of Tizi-Ouzou (UMMTO)**
**Tizi-Ouzou, Algeria**

**Abstract**

This work is mainly motivated by the description of a new approach for calculating the structural similarity of XML documents. Practically, the majority of existing work on XML documents clustering considers the tree structures of these documents as mere vectors and, therefore, does not take into account their hierarchical contexts. Furthermore, in order to calculate the structural similarity of XML documents, most methods encountered in these works perform depth-first traversal to visit the nodes of the tree structures of these documents. More precisely, it is the preorder tree walk which is usually the most used. Recently, other studies present an alternative approach that takes into account the hierarchical contexts of these tree structures, but unfortunately, they have particularly high time complexity in the calculation of structural similarity. In this paper, we propose a new method based on breadth-first traversal of these tree structures. The goal consists in clustering more rapidly XML documents sharing similar structures. Besides the fact that the method is fast, it also takes into account the hierarchical contexts of XML documents. Reconciling the speed required for clustering XML documents with taking into account the hierarchical contexts of their tree structures ensures higher reliability of the proposed method. To validate our proposal, experiments were conducted on both real and synthetic XML data. The results clearly demonstrate the viability of our approach.

***Keywords:*** *Clustering, Structural similarity, hierarchical context, Tree level, Ancestor and descendant levels, depth- and breadth-first traversals.*

## 1. Introduction

XML has now become an unchallenged standard for the representation and exchange of data on the web. This has led to the increase in heterogeneous XML sources. Furthermore, not only the collections of XML documents are reused but their interchange volume is continuously growing. However, with the current available means, the search information in these documents is not a trivial task.

XML documents are characterized by content and structure. However, such documents cannot be exploited efficiently by the conventional information retrieval methods. Indeed, these methods are based on content oriented models, while the XML format allows adding structural constraints. This then requires adapting these models to better exploit the available XML data. Similarly, traditional approaches to data processing, such as relational databases have proven ineffective. These are mainly designed for strongly structured data, whereas XML data are semi-structured [20]. In addition to this, given the heterogeneity and proliferation of XML documents on the web, it becomes difficult for a user to access the desired information. In this context many authors propose methods of classification to organize and analyze large collections of XML documents. Our work falls within this perspective; we are interested in the clustering of XML documents based on their structures. The idea behind the clustering is that if XML documents share similar structures, they are more likely to correspond to the structural part of the same query. This therefore allows reducing the response time and increasing the accuracy of search engines. In other words, it can substantially improve the process of information retrieval. Thus, the search for relevant information in a large collection of documents will then return to interrogate small classes of documents.

XML clustering task consists in grouping XML documents into clusters containing similar documents. This similarity could be thematic or structural. In this paper, we are particularly interested in XML document clustering using the structural similarity of their descriptions, i.e., the XML ordered labeled tree providing the relations between the document elements. We will therefore address the "structural clustering of XML documents" problem as we would have done with a "clustering of tree structures" problem [1, 2, 3, 14, 41]. In other words, structural clustering of XML documents approach can be exploited in various areas that require management of hierarchical structures, such as the discovery of structurally similar web navigational pathways, or tree-like patterns, and the discovery of structurally similar macromolecular tree patterns in bioinformatics [14, 34, 17].

The structural similarity allows to group documents that share similar structures [12]. It will help to better organize XML documents on the one hand and, on the other hand, to better answer, in terms of efficiency and effectiveness, queries containing structural conditions. We recall that queries in XML information retrieval could contain

keywords only or keywords and structural conditions. The main question we address in this context is how to cluster structurally XML documents when their DTD is unknown. Our methodology in this paper is two-step. In the first step, each XML document is represented by its tree summary structure, which is used as a representation model to classify the corresponding XML document. In the second step, an efficient structural similarity measure based on breadth-first traversal of these tree summary structures is proposed. Within this framework, the most important question deals with the way to measure the structural similarity of XML documents. This is the question we attempt to answer in this work.
This paper is organized as follows. Section 2 provides a summary view of related work about classification of XML documents by structure. Section 3 describes our clustering approach. Section 4 is dedicated to the experimentation. Finally, in section 5 we conclude and describe the future work.

## 2. State-of-the-Art

The classification approaches are divided in two main variants called supervised classification and unsupervised classification (or clustering).
Existing works on the classification of XML documents can be distinguished by the way they represent the documents, but also by the classification and/or clustering methods used. We focus here on approaches that represent documents by structure only. Within this framework, we distinguish two main categories: document-based clustering, where the clustering is based on the document structure itself, and DTD-based clustering, where documents are clustered according to their DTD. We briefly describe below some of the most known approaches, highlighting their main features. We are particularly focus on approaches that represent the document structures by labeled trees.
In the first category, namely clustering based on document structure; the structure that is used to classify a document is issued from the document itself.
– This structure could be either a labeled tree corresponding to the original structure of the XML document (the whole structure of document) [5, 21, 29, 30, 31, 40] or a rooted ordered labeled tree summary [1, 2, 3, 14]. The latter has the advantage of reducing the computational complexity in the clustering, but has nevertheless the drawback of breaking the relationships between XML elements. In the study by [14], a tree summary is obtained by two transformations, as shown in Fig. 1: (i) the first one reduces the depth of the tree so that the children of any node having the same label as one of its ancestors become direct descendants (child) from this ancestor, (ii) the second one eliminates duplication of sibling nodes, while in the study by [1, 2, 3], the tree summary is obtained only by eliminating duplication of sibling nodes, i.e., hierarchical relationships between XML elements are not completely changed, and so there is no loss of information.

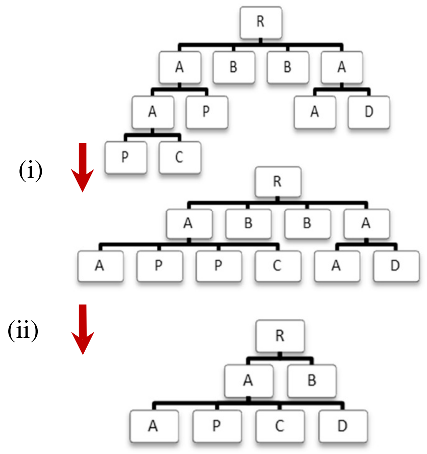


Fig. 1 Tree summary extraction

In the study by [11, 15, 39], another way for representing the structure of XML documents is proposed, namely these approaches are based on the discovery of sub-trees that most frequently occur in collections of labeled trees representing XML documents. More precisely, in the study by [39], the frequent sub-trees can be unordered, whereas, in the other two approaches [11, 15], they absolutely must be ordered. All sub-trees below the arrow in Fig. 2 are frequent according to the approach [39]. The last two of them are exact and ordered, so they are also frequent according to the approach [11, 15]. This indicates that the approach [39] is more general and easily applied to heterogeneous XML documents, as opposed to the approaches [11, 15] that only apply to XML documents sharing the same DTD or XML Schema. Another proposal [20] consists of linearizing the structure of each XML document, by representing it as a numerical sequence and, then, comparing such sequences through the analysis of their frequencies.
– The clustering consists then to compare the extracted structure with a cluster or its representative. This representative, usually called a centroid, is the most representative tree summary of all XML documents in the cluster [1, 2, 3]. The centroid may change, i.e., it can be replaced by a more appropriate tree, depending on the assignment of new documents to the cluster. In the study by [39], a cluster is characterized by the maximal frequent sub-tree, i.e., the frequent sub-tree that has the greatest number of nodes among all sub-trees in this cluster.

Note that with frequent sub-trees technique, an XML document may belong to several clusters. In the study by [14, 21, 29, 30, 31, 40], each cluster is represented by a subset of similarly structured XML documents, whereas with [1, 2, 4, 39] approaches, a cluster has only one representative (the centroid or the maximal frequent sub-tree); this means that, in the process of detecting the appropriate cluster, the representative of a new document is compared only with the representative of the cluster. Note that in some approaches like [1, 2, 4], a new comparison is undertaken for a possible change of the centroid.

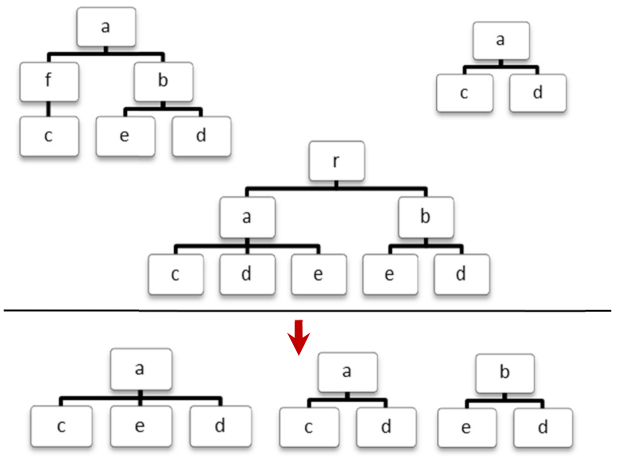


Fig. 2 Frequent sub-trees detection

– Concerning the clustering process itself, several approaches have been proposed. Authors in [30, 40] propose an incremental clustering based on common path similarity taking into account different criterions as the number of common nodes between the XML tree (that of the XML document to be classified) and the trees of the cluster considered, the number of common nodes paths, and the order of the nodes of the XML tree. In the study by [1, 2, 3], the authors also perform an incremental clustering, but it is based on structural similarity between the centroid and the structure of XML document to be classified. For detecting structural similarity between XML documents, the authors in [20] exploit the theory of discrete Fourier transform to effectively and efficiently compare the encoded documents (i.e., signals) in the domain of frequencies. This approach significantly differs from standard methods based on graph-matching algorithms and allows a significant reduction in the required computation costs. Indeed, if $N$ is the maximum number of tags in two documents, their matching complexity is $O(N \log N)$, whereas it is $O(N^2)$ with those based on edit distance, as the Chawathe's algorithm [6] and one proposed in [14]. In the study by [11, 14, 21, 31], the similarity between two trees is based on their edit distance. Edit distance measures the number of elementary operations to transform one tree into another. Most algorithms for calculating the tree edit distance are based on the dynamic programming techniques [6–9, 35, 37, 42, 4]. Note, however, that there may be several sequences of edit operations to transform one tree into another. Therefore, the cost of the operations in each sequence is considered, and the lowest cost sequence among these defines the edit distance between trees [31]. Edit distance allows performing a clustering of these trees using a bottom-up hierarchical classification method [11, 14, 21, 31]. In the clustering approaches based on frequent sub-trees, the authors [39] have developed an algorithm to detect the maximal sub-tree. Similarly, the authors in [15] also have developed their algorithm very close to the algorithms FREQT and TREEMINER proposed respectively by [4] and [45]. An XML tree can appear in multiple clusters. In other words, an XML document can be shared by several clusters, i.e., it consists of several sub-trees appearing respectively in several different clusters. Thus, under these characteristics, we can say that these approaches belong to the family of non-exclusive (or overlapping) clustering.

Concerning the second category approaches, namely DTD classification, we list below some of the most known. Recall that the DTD is considered as a context-free grammar that generates a potentially infinite number of the XML documents. From this fact, instead of classifying directly the XML documents, the approach proposes to classify their DTDs in clusters. Thus, each cluster becomes the representative of a set of structurally similar XML documents. The advantage is that it is possible to more quickly integrate a considerable number of XML documents together. The drawback is that the nodes of DTD trees often denote regular expressions whose handling is not always trivial task and furthermore causes (in some cases) a loss of information [28].

– In [28], the authors propose a DTDs clustering model, named "XClust". Each DTD is represented by its tree structure. The similarity of two nodes is calculated by exploiting different levels of the tree, namely the ontological similarity of the nodes (using a dictionary), the similarity of their immediate descendants (children), the similarity of their ancestors and finally, that of the sub-trees' leaves whose they are respectively the roots.

– In [36], the authors propose a mechanism which identifies syntactically the similarity of DTDs by adopting an ascending clustering strategy. Compared with "XClust", the authors in [41] exploit only the immediate descendant context.

– In [27], the authors develop an algorithm which is based on generic scheme of the DTD matching. The matching goal is to reach a median scheme corresponding to the DTDs that are similarly structured. Like [28], in order to

calculate the DTDs' similarities [27] relies on the dictionary, but it only exploits the leaves' context.
– Finally, the approach proposed in [18] is based on the learning and inference combined with an instance of DTD. In fact, it works with a supervised classification, i.e., the classes are known before starting this classification.
There exist other methods such as approaches of [16, 19, 23, 24, 25, 41, 43, 44] originally dedicated for classifying or clustering XML documents using both structure and content. These methods are in fact more general since they offer flexible models that can easily be adapted for dealing with structure only. For instance, in the study by [16, 43], the authors propose a network-based stochastic model that is able to describe different kind of relationships of XML elements. It was proved that this model is easily adapted to the structure alone. The model is based on Bayesian networks [32], to infer the different type of structural relationships in XML tree.

## 3. Clustering approach based on structural similarity

Our clustering approach belongs to the first category, namely, clustering based on document structure. The structure used to structurally classify a document is issued from the document itself. Specifically, for classifying (clustering in our case) XML documents by structure, we suggest the usage of their corresponding labeled tree structural summaries. The labels correspond to tags or attributes. In our approach, attributes are treated as mere tags. A labeled tree summary representing an XML document is automatically extracted from the document by a parser. This extracted "tree summary" is then used as a model of representation by a classifier to classify the corresponding XML document. We show and explain in Subsection 3.1, how this parser works. Finally, we give a description of the proposed similarity measure in Subsection 3.2. Note that the latter is based on the breadth-first traversal of XML documents' tree structures.

### 3.1 Tree structural summary extraction

We propose to represent XML documents by their tree structural summaries that need minimal processing and especially avoid the loss of information. The basic idea is that repetitions of tags and/or the possibility to have optional tags (and sub-trees consequently) are one of the reasons why XML documents can be structurally different even though they share the same DTD. In this context, our tree summary is regarded as a generic structure in the sense that, when sibling tags are duplicated, it is not necessary to have this duplication in the structure that we wish to extract. Note however, that to avoid losing information, duplications of nested and/or cousin tags are not removed as duplications of sibling tags. One way to avoid this is to consider them as immediate descendants (sub-tags) of tags in which they appear in XML documents. In Fig. 3 we show an overview of this representation approach focusing on all its features. Indeed, the transformation of the original tree (i) in the tree summary (ii) shows that the attribute "*t*" becomes in fact an immediate (direct) descendant (son) node of the root node "*a*". As for the duplication of sibling nodes "*b*", it is removed while keeping the children ("*c*", "*b*" and "*c*") attached to a single occurrence of "*b*", but the nodes "*c*", which were originally cousin nodes on (i), have become brothers, whose we also eliminated duplication. However, as recommended, duplications of nested nodes "*a*" and "*b*" are maintained.
Our extracting algorithm of tree summary is two-step. The first step is based on SAX (Simple API for XML) API (Application Programming Interface), which returns all the tags and attributes encountered in an XML document. These tags (or attributes) are intercepted, filtered and then transformed by our parser into an intermediate form as shown through the parenthesized expression in Fig. 4. In the second step, this intermediate parenthesis expression is transformed by another parser into the corresponding tree summary, according to projections of our approach. i.e., by eliminating duplication of sibling nodes and considering each attribute as an immediate descendant of the element (tag) which it is attached in the XML document. Most of the extraction task is performed during this second step. In fact, three essential operations are performed at this level:
– Passage from the linear form of the XML document to its hierarchical representation;
– Removal of repetitions of sibling nodes;
– Transforming possible attributes into immediate descendants of the elements which they are attached in the XML document.

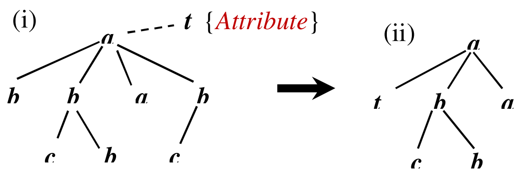
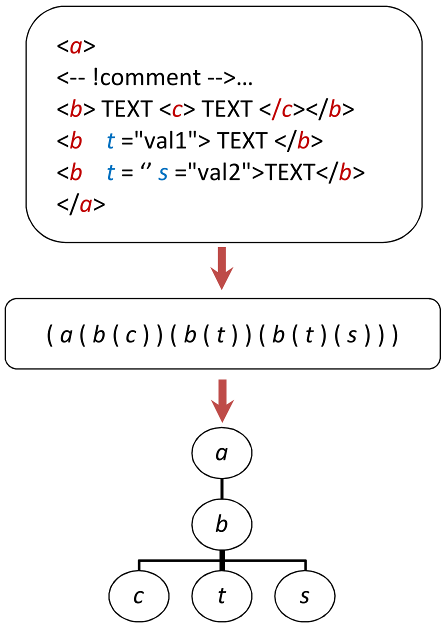


Fig. 3 Representation approach of an XML document

Thus, instead of the original trees to represent XML documents, we use their "tree structural summaries", but without loss of information, since we remove only the repetitions of sibling nodes. This allows, on the one hand, performing the matching of these trees more quickly and easily and, on the other hand, to provide high-quality clustering.



```
<a>
<-- !comment -->...
<b> TEXT <c> TEXT </c></b>
<b  t ="val1"> TEXT </b>
<b  t = '' s ="val2">TEXT</b>
</a>
```

Fig. 4 An XML document and its corresponding tree summary

## 3.2 XML documents clustering

### *3.2.1 Overview of the clustering technique used*

For clustering XML documents based on structural similarity we use well-known techniques in hierarchical agglomerative clustering (although any form of clustering could be used). Hierarchical methods perform mergers between data sets; the peers of elements (or clusters) are successively merged until there is only one large set containing all elements. The end result can be schematically represented as a tree of clusters named *dendrogram*, as shown in Fig. 5.

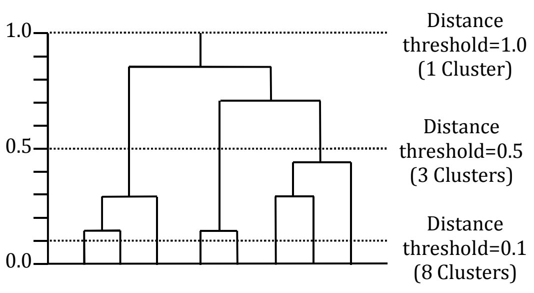


Fig. 5 Dendrogram of the ascending hierarchical classification

The dendrogram shows the clusters that were merged together, and the minimum similarity between these merged clusters.

There are several methods of hierarchical ascending classification. They all are based on the following idea:

a) Initially, each element of the data set to be classified is regarded as a cluster.
b) Clusters separated by a minimum distance (i.e., maximum similarity) are grouped together. The distances between the remaining clusters and the new cluster set are recalculated.
c) If there is more than one cluster or has not yet reached the minimum distance (or maximum similarity threshold), go to step b.

With some methods, the distance between two clusters *X* and *Y* is defined as the minimum distance (maximum similarity) between all the peers of elements (*x*, *y*) such that *x* is in *X* and *y* in *Y*. With other methods, this is the average distance (average similarity) which is considered as a parameter of the separation of clusters. We chose clustering method which is based on the minimum distance (i.e., the maximum similarity). We then used the *single link* clustering algorithm using Prim's algorithm [10] to calculate the MST (Minimum Spanning Tree or shortest path) of a graph.

Given a graph $G = (N, A)$ with a set of weighted edges *A*, and a set of nodes *N*. The minimum spanning tree (MST) of a graph is an acyclic subset $T \subseteq A$ that chain all nodes whose total weight (cost, distance, value, etc.) denoted $W(T)$ (the weight sum of *T*'s edges) is minimized. It was shown in [22] that the MST contains all the information required to implement the *single link* clustering.

Given a set of rooted labeled ordered trees representing XML documents, we form a complete graph *G* with *n* nodes $\in N$ and $\frac{n(n-1)}{2}$ weighted edges $\in A$. The weight of an edge is the structural distance between the nodes it connects. Nodes represent XML trees in our case. For example, the *single link* clustering for threshold *l* can be carried out by removing all the edges having a weight $\geq l$ of MST in the *G* graph. The connected nodes of the remaining graph are the *single link* clusters.

It can be seen in Fig. 6 a graph with 7 nodes (corresponding to 7 XML documents), and 10 edges.

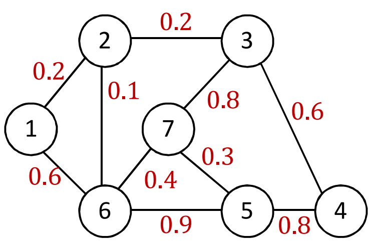


Fig. 6 Graphical representation of the distances between XML trees

As indicated above, the weight of an edge is the structural distance between XML documents. For example, the structural distance between the tree 1 and tree 2 is 0.2.
Missing edges are the additional edges which make the complete graph; their weights are equal to 1. Fig. 7 shows the shortest path on the graph in Fig. 6. It can be seen in Fig. 8 the parts of graph remaining after deleting all edges with weight $\geq 0.4$.

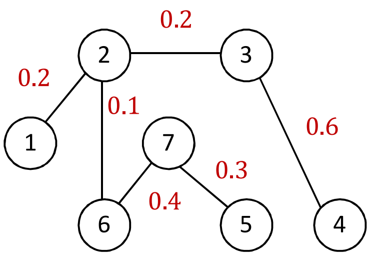


Fig. 7 The shortest path in the graph of Fig. 6

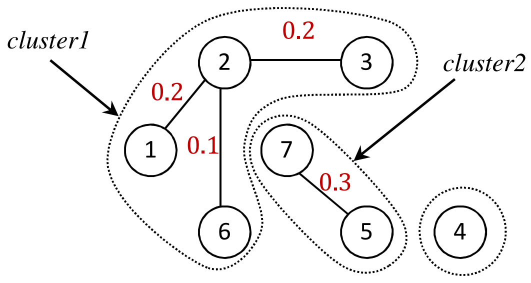


Fig. 8 Resulting graph after deleting all edges having weight $\geq 0.4$

There are two new components that are formed, containing the nodes (1, 2, 3, 6) and the nodes (5, 7), respectively.
This indicates the presence of two new clusters, namely *cluster1* with (1, 2, 3, 6) as members and *cluster2* with (5, 7) as members. Nodes that are not connected to other nodes are considered as a single node clusters.
The graph is represented by a matrix called the associated matrix. Is associated to the graph $G = (N, A)$ of order $n$, a square matrix of order $n$. This matrix is formulated as follows:

$$M(n \times n) = \{b_{ij} \mid_{i=1..n, j=1..n}\}$$
$$b_{ij} = \{weigths\ if\ a(x_i, x_j) \in A\}$$

In Tables 1, 2 and 3 are shown matrices respectively associated with graphs of Figs 6, 7 and 8.
It suffices now to use the matrix obtained after applying the threshold $\geq 0.4$ to deduce the remaining links between nodes (representing XML documents) and then build the corresponding clusters.

Table 1: Matrix associated with the graph of Fig. 6

| $M(7\times 7)$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | 0.2 | | | | | | |
| 3 | | 0.2 | | | | | |
| 4 | | | 0.6 | | | | |
| 5 | | | | 0.8 | | | |
| 6 | 0.6 | 0.1 | | | 0.9 | | |
| 7 | | | 0.8 | | 0.3 | 0.4 | |

Table 2: MSP matrix of the matrix of Fig. 7

| $M(7\times 7)$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | 0.2 | | | | | | |
| 3 | | 0.2 | | | | | |
| 4 | | | 0.6 | | | | |
| 5 | | | | | | | |
| 6 | | 0.1 | | | | | |
| 7 | | | | | 0.3 | 0.4 | |

Table 3: Matrix after applying threshold 0.4

| $M(7\times 7)$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | 0.2 | | | | | | |
| 3 | | 0.2 | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | 0.1 | | | | | |
| 7 | | | | | 0.3 | | |

*3.2.2 Overview of using Prim's algorithm*
As announced above, Prim's algorithm [33] allows calculating the shortest path (or MST) in a given weighted graph *G*. In an informal way, we apply the following points:

– Create a tree containing a single node, chosen arbitrarily from the graph *G*
– Create a set containing all the edges in the graph *G*
– loop until every edge in the set connects two nodes in the tree
  ▪ remove from the set an edge with minimum weight that connects a node in the tree with a node not in the tree
  ▪ add that edge to the tree

Thus, the algorithm continuously increases the size of a tree, one edge at a time, starting with a tree consisting of a single node, until it spans all nodes of the initial graph *G*.
A pseudo-code for Prim's algorithm is given in Fig. 9.
To show how to apply Prim's algorithm to find a minimum spanning tree in the weighted graph, we rely on the example of graph in Fig. 10. Prim's algorithm will proceed as follows. First we arbitrarily choose to start with the node *d*, and then we add edge {*d*, *e*} of weight 1. Next,

we add edge {$c$, $e$} of weight 2. Next, we add edge {$d$, $f$} of weight 2. Next, we add edge {$b$, $e$} of weight 3. And finally, we add edge {$a$, $b$} of weight 2. This produces a minimum spanning tree of weight = 10. The minimum spanning tree found is given in Fig. 11.

```
Input: Given a non-empty connected weighted graph
G = (N, A), (the weights can be negative)
Initializations:
  $N_{new} \leftarrow \{x\}$; $A_{new} \leftarrow \phi$;  (where x is an arbitrary
  node (starting point) from N)
repeat
   choose an edge {u, v} with minimal weight such
   that u is in $N_{new}$ and v is not (if there are multiple
   edges with the same weight, any of them may be
   picked)
   $N_{new} \leftarrow N_{new} \cup \{v\}$; $A_{new} \leftarrow N_{new} \cup \{u, v\}$
until $N_{new} = N$
Output: $N_{new}$ and $A_{new}$ describe an MST
```

Fig. 9 Prim's algorithm pseudo-code

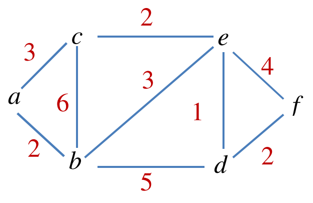


Fig. 10 An example of a weighted connected graph

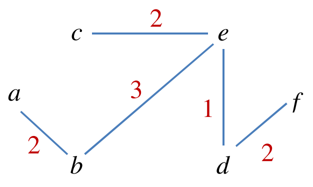


Fig. 11 Minimum spanning tree (MST) produced by applying Prim's algorithm on the graph in Fig. 10

We could start with any node to determine the MSP. In the case of the previous example (in Fig. 10), we arbitrarily chose to start with the node $d$. But any node can be used to start the process with Prim's algorithm.

The time complexity of the algorithm depends heavily on how the choice is implemented in the edge / node to add to the set at each stage. With a naive representation, using an adjacency matrix graph representation and searching an array of weights to find the minimum weight edge to add requires $O(N^2)$ running time. Using a simple binary heap data structure and an adjacency list representation, Prim's algorithm can be shown to run in time $O(A \log N)$. Using a more sophisticated Fibonacci heap, this can be brought down to $O(A + N \log N)$, which is asymptotically faster when the graph is dense enough that $A$ is $\omega(N)$, i.e. $A$ dominates $N$ asymptotically. However, we chose, for the purposes of our tests in this article, the adjacency matrix for the simplicity of its implementation.

At this stage, as previously announced, we focus in Subsection 3.3, on the description of the structural similarity measure proposed.

## 3.3 Tree structure similarity

Usually, to compare two words we use a thesaurus or dictionary. But when these words correspond to node names (labels) in a tree, it is necessary to take into account their respective tree relationships. The idea is that even though two nodes are represented by the same name, or by synonymous names, this does not mean that they remain necessarily similar in the context of their respective ancestors, descendants, siblings and/or cousins, which can be completely different. Thus, the similarity of two nodes depends not only on their ontological similarity (terms could be similar because they have same string or could be semantically related using a dictionary), but also on their respective tree relationships that play a crucial role in the similarity calculation.

Most methods for clustering XML documents by structure use the edit distance for measuring the similarity between their structures. We recall that tree edit distance measures the number of elementary operations (insertions, deletions and replacements of nodes) required to transform one tree into another. On the other hand, all these methods perform depth-first traversal to visit nodes of tree.

We propose a novel method for calculating the similarity:

- Firstly, instead of performing depth-first traversal to visit nodes of a tree, our proposal is to perform breadth-first traversal, also called *level by level* traversal. In other words, we explore the breadth, i.e., full width of the tree at a given level, before going deeper.
- Secondly, we take into consideration the hierarchical contexts of XML tree structures.

Before describing in detail our method, it is necessary to introduce some fundamental concepts.

### *3.3.1 Basic preliminary notions*

A tree level consists of sibling and/or cousin nodes. As suggested in our approach, repetitions of sibling nodes will be eliminated, but not those of the cousin nodes. Therefore, it is possible to encounter on a same tree level several duplications of cousin nodes. It is then necessary in such case to take them into account in the similarity calculation. To express that, we can use the concept of weight. Indeed, let $V = [x_1, x_2, \dots, x_n]$ be a vector $\in \mathbb{R}^n$; its norm (Euclidean distance) is $\|V\| = \sqrt[2]{\sum_{i=1}^{n} x_i^2}$. The usage of the norm allows exploiting efficiently the concept of weight. We can extend its use even to objects that are not necessary vectors of $\mathbb{R}^n$. Indeed, for example, if $L = (a, b, b, c, c, c, a)$ is a tree level, then the weights (or frequencies) of $a$, $b$ and $c$ are 2, 2 and 3, respectively. Therefore, if these weights are stored in a vector such as $M = [2, 2, 3]$ then the norm associated with $L$ is $\|M\| = \sqrt[2]{2^2 + 2^2 + 3^2} = \sqrt[2]{17}$. The norm will serve thereafter for the normalization of the similarities' values.

Moreover, in order to fully highlight features of our approach, it should also recall some notions on depth- and breadth-first traversals of trees. Indeed, there are essentially two different methods in which to visit systematically all the nodes of a tree, namely, depth-first traversal and breadth-first traversal. Certain depth-first traversal methods occur frequently enough that they are given names of their own: *preorder* traversal, *inorder* traversal and *postorder* traversal. To describe these concepts easily and clearly it is better to rely on concrete examples. In fact, we do not really need dwell too long on the details of the tree traversal; we give only the minimum necessary to distinguish the breadth-first traversal (which characterizes our proposed method) and the depth-first traversal that was used in most existing clustering methods. Thus, for example, given the tree in the Fig. 12:

a preorder traversal would visit the elements in the order: A, B, C, D, E, F, G, H, I. This type of traversal is called a depth-first traversal because it tries to go deeper in the tree before exploring sibling nodes.

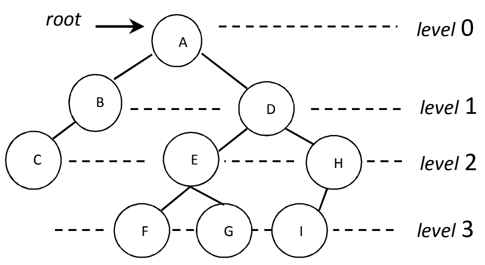


Fig. 12 Simple general tree

For example, the traversal visits all the descendants of B (i.e., keeps going deeper) before visiting B's sibling D (and any of D's descendants).

As we have seen, this kind of traversal can be achieved by a simple recursive algorithm given in Fig. 13.

Whereas the depth-first traversals are defined recursively, breadth-first traversal is best understood as a non-recursive traversal. The breadth-first traversal of a tree visits the nodes in the order of their depth in the tree. Breadth-first traversal algorithm first visits all the nodes at level 0 (i.e., the root), then all the nodes at level one, and so on. At each level the nodes are visited from left to right. Thus, a breadth-first traversal of the tree shown in Fig. 12 visits the nodes in the following order: A, B, D, C, E, H, F, G, I.

```
preorder (tree)
  if (tree not empty)
    visit root of tree
     preorder (left sub_tree)
     preorder (right sub_tree)
```

Fig. 13 Preorder traversal algorithm

*3.3.2 Breadth- first tree traversal*

To our knowledge, the breadth-first traversal algorithm has not been practically applied in existing work on clustering of XML documents. We encountered only one approach in [29] that addressed the similarity computation according to the similarities of the levels of XML tree structures.

Recall that in our approach, the representative structures of XML documents are tree structural summaries, structured as general trees, i.e., where each tree node can have any number of children. The algorithm in Fig. 14 allows exploring a general tree and retrieving its nodes, adopting the breadth-first traversal. The breadth-first traversal has linear time complexity *O (N)* in the worst case, as the depth-first traversal.

```
breadh_traversal (n : Node)
    begin
     level ← {n}
     while level ≠ $\phi$;
      {dept_level ← $\phi$;
       for each node a ∈ level
        {store a in list;
         depth_level ← depth_level ∪ child_of (a);}
       level ← depth_level ;}
    end
```

Fig. 14 Breadth-first traversal algorithm

Indeed, given a tree of *N* nodes, the algorithm in Fig. 14 clearly shows the linearity of the complexity time. At each level, the nodes are visited from left to right, and then stored in lists that will be used thereafter for calculating similarities. The advantage of storing the nodes in the lists is twofold: On the one hand, this allows easy calculation of basic similarities between levels of trees. On the other hand, given two tree levels belonging respectively to two trees, it is possible to know the similarities of their respective ancestor and descendant levels. As suggested above, the ancestor and descendant levels represent somehow hierarchical contexts to take into account in calculating the similarity of two levels of two given trees. These levels are somehow implicitly linked by hierarchical relationships in trees. The underlying idea is that even though two tree levels are identical, or very similar, this

does not mean that they remain necessarily similar in the context of their respective ancestor and descendant levels which can be completely different.
So, given all these characteristics, we describe and explain in Subsection 3.3.3, the structural similarity measure that we propose, taking into account the hierarchical relationships between levels in each tree.

### *3.3.3 Structural similarity measure based on breadth-first tree traversal*

Let $T_1$ and $T_2$ be two trees representing respectively two XML documents. We propose to compute their similarity as follows:

$$Sim(T_1, T_2) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} Sim_{level}(l_{1i}, l_{2j})}{\text{Max}(n,m)} \qquad (1)$$

$Sim_{level}(l_{1i}, l_{2j})$ is the similarity of the levels $l_{1i}$ and $l_{2j}$. The levels $l_{1i}$ and $l_{2j}$ belong respectively to $T_1$ and $T_2$. The bounds $n$ and $m$ are the levels' numbers of $T_1$ and $T_2$ respectively.
Given two levels $l_{1i}$ and $l_{2j}$, we define their similarity according to their hierarchical context as follows:

$$Sim_{level}(l_{1i}, l_{2j}) = w_1 * S_1 + w_2 * S_2 + w_3 * S_3 \qquad (2)$$

$w_1 \geq 0$, $w_2 \geq 0$ and $w_3 \geq 0$ are weights such that $w_1 + w_2 + w_3 = 1$.

$S_1$ is the basic similarity of $l_{1i}$ and $l_{2j}$. It is expressed as follows:

$$S_1 = \frac{\sum_{k=1}^{p}\sum_{l=1}^{q} Sim_{node}(e_{1k}, e_{2l})}{\|N_1\| * \|N_2\|} \qquad (3)$$

The term $Sim_{node}(e_{1k}, e_{2l})$ is the ontological similarity of the nodes $e_{1k}$ and $e_{2l}$ (obtained using a dictionary). In other words, $Sim_{node}(e_{1k}, e_{2l}) = 1$ if $e_{1k} = e_{2l}$, $Sim_{node}(e_{1k}, e_{2l}) \cong 1$ if $e_{1k}$ and $e_{2l}$ are synonymous, otherwise $Sim_{node}(e_{1k}, e_{2l}) = 0$. The nodes $e_{1k}$ and $e_{2l}$ belong respectively to the levels $l_{1i}$ and $l_{2j}$. The bounds $p$ and $q$ are the nodes' numbers of $l_{1i}$ and $l_{2j}$ respectively.
The product $\| N_1 \| * \| N_2 \|$ allows normalizing the sum $\sum_{k=1}^{p}\sum_{l=1}^{q} Sim_{node}(e_{1k}, e_{2l})$. The terms $N_1$ and $N_2$ are two vectors whose elements are weights of nodes belonging respectively to the tree levels $l_{1i}$ and $l_{2j}$. Thus, $S_1$ is calculated for each pair of levels ($l_{1i}$, $l_{2j}$). So the result is the basic similarity matrix of trees $T_1$ and $T_2$. In Subsection 3.3.4, we give an idea about the calculation of this matrix.
$S_2$ and $S_3$ in some way reflect the hierarchical context in calculating the similarity of each pair of levels ($l_{1i}$, $l_{2j}$). $S_2$ represents the similarity of descendant levels of $l_{1i}$ and $l_{2j}$ respectively. It is expressed as follows:

$$S_2 = \frac{\sum_{k=1}^{r}\sum_{l=1}^{s} Sim_{desc}(d_{1k}, d_{2l})}{\text{Max}(r,s)} \qquad (4)$$

The term $Sim_{desc}(d_{1k}, d_{2l})$ represents the basic similarity of the levels $d_{1k}$ and $d_{2l}$. The levels $d_{1k}$ and $d_{2l}$ belong respectively to $desc_1$ and $desc_2$. The terms $desc_1$ and $desc_2$ are the sets of descendant levels of $l_{1i}$ and $l_{2j}$, respectively. The bounds $r$ and $s$ are the levels' numbers of $desc_1$ and $desc_2$, respectively.

$S_3$ is the similarity of ancestor levels of $l_{1i}$ and $l_{2j}$ respectively. It is expressed as follows:

$$S_3 = \frac{\sum_{k=1}^{t}\sum_{l=1}^{u} Sim_{anc}(a_{1k}, a_{2l})}{\text{Max}(t,u)} \qquad (5)$$

The term $Sim_{anc}(a_{1k}, a_{2l})$ represents the basic similarity of the levels $a_{1k}$ and $a_{2l}$. The levels $a_{1k}$ and $a_{2l}$ belong respectively to $anc_1$ and $anc_2$. The terms $anc_1$ and $anc_2$ represent the sets of ancestor levels of $l_{1i}$ and $l_{2j}$, respectively. The bounds $t$ and $u$ are the levels' numbers of $anc_1$ and $anc_2$, respectively.

### *3.3.4 Illustrative example*

This example shows the different steps followed in computing the similarity of the two trees $T_1$ and $T_2$ in Fig. 15 using the proposed structural similarity measure based on breadth-first tree traversal.
The first step is to use Eq. (3) to calculate the similarity matrix of levels of $T_1$ and $T_2$. As there are three levels in each tree ($T_1$ and $T_2$), we will have a matrix (3×3). The calculation gives the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{2}{\sqrt[2]{3}*\sqrt[2]{2}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.82 \end{pmatrix}$$

We note that the similarity between the last levels of $T_1$ and $T_2$ respectively is equal to 0.82, while it is equal to 1 between the other levels of the same rank. It is equal to 0 everywhere else.

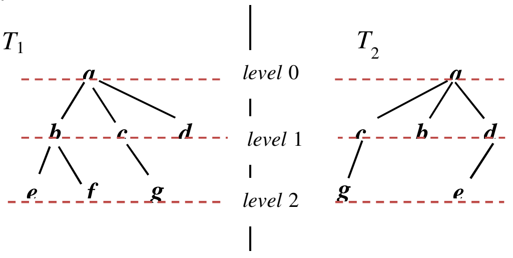


Fig. 15 Comparison of two XML trees using the calculation of the structural similarity based on the breadth-first traversal

Before calculating $S_2$ and $S_3$, it would be appropriate to define how to use the weights $w_1$, $w_2$ and $w_3$. Indeed, if we ignore the hierarchical contexts (descendant levels and ancestors levels), it is not necessary to calculate $S_2$ and $S_3$, in this case we take $w_1 = 1$ with $w_2 = 0$ and $w_3 = 0$. Otherwise, in particular in the case of XML documents, it is more natural to give to $S_1$, $S_2$ and $S_3$ the weights $w_1 = \frac{1}{3}$, $w_2 = \frac{1}{3}$, and $w_3 = \frac{1}{3}$, respectively. Thus, with respect to the first case mentioned, namely that we do not consider the hierarchical contexts, the similarity between two tree levels of two trees, respectively, is defined by $S_1$.

Indeed, with ($w_1$ = 1, $w_2$ = 0 and $w_3$ = 0), we have $Sim_{level}(l_{1i}, l_{2j}) = w_1 * S_1 = S_1$, because $w_1$ = 1. So the final similarity calculation of the two trees $T_1$ and $T_2$ becomes easy and requires only exploring the matrix (3×3) calculated above with the formula (1). There will therefore $Sim(T_1, T_2) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} Sim_{level}(l_{1i}, l_{2j})}{\text{Max}(n,m)} = \frac{1 + 1 + 0.82}{\text{Max}(3,3)} = \frac{2,82}{3} =$ 0.94 which is relatively a good similarity value that we could get by comparing two vectors, so it does not reflect the tree view of XML documents.

Having said this, but if we consider the case where $w_1 = \frac{1}{3}$, $w_2 = \frac{1}{3}$ and $w_3 = \frac{1}{3}$, the calculation is obviously more complicated, but in principle reflects more reliable similarity calculation. The elements of the new similarity matrix, before calculating the final similarity, are calculated using Eq. (2), with $w_1 = \frac{1}{3}$, $w_2 = \frac{1}{3}$ and $w_3 = \frac{1}{3}$. So we will have $Sim_{level}(l_{1i}, l_{2j}) = \frac{S_1+S_2+S_3}{3}$. In this case, we must also calculate $S_2$ and $S_3$ using Eqs. (4) and (5). But to go faster, we calculate $S_2$ and $S_3$ only for non-zero similarity of the matrix calculated above. The elements concerned are those of the main diagonal of the matrix, namely (1, 1), (2, 2) and (3, 3) which are represented by the values 1, 1 and 0.82, respectively. Moreover, it should also be noted that some elements of the matrix are not concerned by the calculation of $S_2$ or $S_3$, as for example those of the last row or those of the first row of the matrix. But for not distort the similarity computation, we attribute the value 1 to $S_2$ and $S_3$, in the case of the last row and the first row of the matrix, respectively. Thus, for each element of the matrix equal to 0, we calculate the values of $S_2$ and $S_3$ as follows:

The element (1, 1) has no ancestor levels so $Sim_{anc} = 1$, i.e., $S_3$ = 1, but it has two descendant levels, namely (2, 2) and (2, 3) such that $Sim_{desc}(2, 2) = 1$ and $Sim_{desc}(2, 3) = 0$, that is to say $S_2 = \frac{1+0}{Max(2,2)} = \frac{1}{2} = 0.5$. So by applying Eq. (2) we have $Sim_{level}(1, 1) = \frac{S_1+S_2+S_3}{3} = \frac{1+0.5+1}{3} = 0.834$. The element (2, 2) has only one ancestor level and one descendant level, corresponding respectively to (1, 1) and (3, 3), which gives $Sim_{anc}(1, 1) = 1$ and $Sim_{desc}(3, 3) = 0.82$, i.e., $S_2 = \frac{Sim_{desc}(3,3)}{Max(1,1)} = \frac{0.82}{1} = 0.82$ and $S_3 = \frac{Sim_{anc}(1,1)}{Max(1,1)} = \frac{1}{1} = 1$. So by applying Eq. (2) we have $Sim_{level}(2, 2) = \frac{S_1+S_2+S_3}{3} = \frac{1+0.82+1}{3} = \frac{2.82}{3} = 0.94$. The last case concerns the element (3, 3) that has no descendant levels, but has four ancestor levels, namely (2, 2), (2, 1), (1, 2) and (1, 1). Regarding the descendant level, we assign the value 1, as expected, to $Sim_{desc} = 1$, i.e., $S_2$ = 1. Other values are calculated as follows: $Sim_{anc}(2,\ 2) = 1$, $Sim_{anc}(2,\ 1) = 0$, $Sim_{anc}(1,\ 2) = 0$ and $Sim_{anc}(1,\ 1) = 1$. Thus, we have $S_3 = \frac{1+0+0+1}{Max(2,2)} = 1$. Finally, we obtain $Sim_{level}(3, 3) = \frac{S_1+S_2+S_3}{3} = \frac{0.82+1+1}{3} = 0.94$. The final matrix formed by the elements $Sim_{level}(l_{1i}, l_{2j})_{\ i=1..3, j=1..3}$ before calculating the similarity of the two trees $T_1$ and $T_2$ is given as follows:

$$\begin{pmatrix} 0.834 & 0 & 0 \\ 0 & 0.94 & 0 \\ 0 & 0 & 0.94 \end{pmatrix}$$

Applying the equation (1), we obtain $Sim(T_1, T_2) = \frac{\sum_{i=1}^{3}\sum_{j=1}^{3} Sim_{level}(l_{1i}, l_{2j})}{\text{Max}(3,3)} = \frac{0.834+0.94+0.94}{3} = 0.905$. Unlike the first result (namely 0.94) without taking into account the hierarchical contexts of trees $T_1$ and $T_2$, i.e., with $w_1$ = 1, $w_2$ = 0 and $w_3$ = 0, the latter result (namely 0.905), seems to better reflect the reality of the tree structure of XML documents. This example gives an idea on how to calculate the similarity according to our approach, but to validate our proposal we will make several tests in the experimental part of this paper.

### *3.3.5 Complexity of the structural similarity calculation*

Given a general tree of $M$ nodes and height $h$, this latter is equal to the number of tree levels. So, a tree level, other than that of the root contains on average $\frac{M}{h}$ nodes. Therefore, given two trees having levels containing respectively $\frac{M_1}{h_1}$ and $\frac{M_2}{h_2}$ nodes, then the calculation of their basic similarity matrix is achieved on average in $M_1 \times M_2$ operations since they have respectively $h_1$ and $h_2$ levels.

In other words, it requires time complexity of order $O(M_1 \times M_2)$, which is the same as that of calculating the similarity based on edit distance. However, in our approach, unlike approaches based on edit distance, we extend the similarity calculation taking into account the tree relationships between nodes. It will therefore be necessary to add the calculation of descendant and ancestor levels' similarities, respectively. Indeed, based on a basic similarity matrix $S_1$ [1..$h_1$, 1..$h_2$], the worst case time complexity of the additional calculation is on the order $O((h_1 \times h_2)^2)$. Note however, that the heights $h_1$ and $h_2$ are usually relatively much smaller than the tree sizes (numbers of nodes) $M_1$ and $M_2$ respectively. We thus obtain a time complexity slightly higher than that of the edit distance, but this is acceptable given the relevance of the proposed similarity measure that takes into account the hierarchical relationships of nodes.

***Remark*** given that we have proposed a similarity measure other than that based on the distance for clustering XML documents, on the one hand and, on the other hand, we relied on Prim's algorithm that computes the shortest path (MST) in a graph which is then exploited for clustering XML documents based on their structural distances (each node of the MST, symbolizes the structure of an XML document), it is then necessary to adapt our similarity measure. To do this, it suffices to replace the similarity value calculated on the basis of the similarity measure proposed by the distance value according to the following Eq. (6).

$$Distance = Similarity - 1 \qquad (6)$$

In the next section, we evaluate the effectiveness and efficiency of our approach. To this end, we conducted our experiments relying on several different tests.

## 4. Experiment and results

### 4.1. Implementation of the clustering system

We developed a first program in java, under the Jcreator environment. The developed program consists of two modules: the first one is based on SAX to carry out the first parsing as announced in Subsection 3.1. This module provides for each treated XML document an intermediate file intercepted by a second module to finalize the extraction of its corresponding tree summary. For clustering XML summary trees obtained using the previous parsing program (i.e., tree structural summaries' extraction program) we wrote a second program in C++ that uses the files (representing the tree summaries) generated by the first program for clustering them.

### 4.2. Experimental framework

Our experiments are carried out on a Lenovo, Intel Core 2 Duo 2 GHz CPU and 2.99 GB of RAM. For data set, the experiments were carried out on both real (ACM SIGMOD Record[1]) and synthetic XML collections. ACM SIGMOD Record corpus concerns scientific articles published by ACM SIGMOD conference and is composed of approximately 1,000 XML documents sharing 5 DTDs, namely HomePage, IndexTermsPage, OrdinaryIssuePage, ProceedingsPage, and SigmodRecord. These DTDs can, in fact, be considered as target classes against which we can assess our clustering approach. This corpus is distributed as shown in Table 4.

Table 4: ACM SIGMOD Record corpus distribution

| DTD | Number of XML documents |
|---|---|
| IndexTermsPage | 920 |
| OrdinaryIssuePage | 30 |
| ProcedingsPage | 16 |
| SigmodRecord | 1 |
| HomePage | 1 |

[1] http://www.sigmod.org/publications/sigmod-record/xml-edition

### 4.3. Evaluation metrics

The evaluation is to verify to what extent the clustering is susceptible to find clusters in agreement with the classes of the labeled corpus, which are considered as target classes. To validate our approach, we used the *F-measure*, *Recall* and *Precision* measures, which are commonly used metrics to assess the clustering results.
*F*1 (*F-measure*) [26] is a combination of *Precision* and *Recall*. It measures the balance between *P* (*Precision*) and *R* (*Recall*) expressed respectively by the following Eqs. (7) and (8).

$$P = \frac{X_d}{N_c} \qquad (7)$$

$$R = \frac{X_d}{N_d} \qquad (8)$$

$N_c$ is the number of documents in the cluster *C*, $N_d$ is the number of documents in the target class (DTD) and $X_d$ is the number of documents in the target class assigned to cluster *C*. We recall that each DTD is considered as a target class with which we can evaluate our clustering. So we know a priori these classes, i.e., we know their numbers and the names of the documents they contain. The *F-measure* $F_1$, in turn, is expressed by Eq. (9) representing the harmonic mean of *Precision* and *Recall*.

$$F_1 = \frac{2*P*R}{P+R} \qquad (9)$$

### 4.4. Evaluation and discussion

In this phase, we first derive from the previous XML collection, the corresponding "tree summaries", and we then respectively proceeded to their clustering. The first clustering test consists in comparing the measure of similarity proposed with the similarity measure based on "tree edit distance" and the similarity measure proposed in [3]. The second test is to compare some of our results with those of existing approaches. Finally, the third test is to confirm the asymptotic time complexity of our similarity measure.

#### *4.4.1 Similarity measure proposed versus other similarity measures*

In the first test, as expected, we compared the similarity measure proposed to another measures, namely the edit distance and the similarity measure proposed in [3]. We chose to compare our similarity measure with the edit distance, because the latter is a measure of similarity that has been widely used in many clustering approaches. The comparison with the work presented in [3], is justified by the fact that we use exactly the same model for representing XML documents, in this case, structural tree summaries. This comparison test is particularly motivated by the response time of our clustering on the one hand and, on the other hand, by the reliability of our similarity

measure. For this, we first replaced in our clustering algorithm, the similarity measure described by Eq. 6, respectively, by the edit distance and similarity measure proposed in [3]. We then performed three series of tests with the same values of distance threshold in the interval [0.1- 0.9]. However, given the recurring results (especially for the 920 documents corresponding to IndexTermsPage which are almost identical, thus structurally very similar), we used only $\frac{1}{3}$ of the corpus, namely 348 XML documents, i.e., (300 + 30 + 16 + 1 + 1) corresponding respectively to IndexTermsPage, OrdinaryIssuePage, ProceedingsPage, SigmodRecord, and HomePage.
To have a clear idea about the performance and reliability of our tests, it would be appropriate to report the comparative results (for the same type of test) published in [3]. These results are given in Table 5. Some abbreviations used in Table 5: NC is the number of clusters. The time unit on the column named "Time" is the second. The abbreviations SM and ED denote respectively "Similarity Measure" and "Edit Distance". Finally, the abbreviation TH represents the similarity threshold.

Table 5: [3]'similarity measure versus edit distance

| TH | [3]' SM | | ED | |
|---|---|---|---|---|
| | NC | Time | NC | Time |
| 0.1 | 1 | $1.67\times10^3$ | 1 | $1.43\times10^2$ |
| 0.2 | 2 | $1.52\times10^3$ | 2 | $1.31\times10^2$ |
| 0.3 | 2 | $1.52\times10^3$ | 2 | $1.31\times10^2$ |
| 0.4 | 3 | $1.43\times10^3$ | 2 | $1.31\times10^2$ |
| 0.5 | 4 | $1.35\times10^3$ | 4 | $1.16\times10^2$ |
| 0.6 | 5 | $1.28\times10^3$ | 5 | $1.02\times10^2$ |
| 0.7 | 5 | $1.28\times10^3$ | 5 | $1.02\times10^2$ |
| 0.8 | 7 | $1.17\times10^3$ | 5 | $1.02\times10^2$ |
| 0.9 | 9 | $1.10\times10^3$ | 7 | $0.85\times10^2$ |

Note that the clustering algorithm in [3] is completely different from the clustering algorithm we proposed in this article. In this regard, we recall that our clustering here is based on a conventional agglomerative hierarchical classification, while that of the approach [3] is an incremental clustering.
As anticipated above, after completing the first test with our similarity measure (based on Eq. 6), we replace, in our clustering algorithm, our similarity measure, successively, by the edit distance and [3]'similarity measure. We then perform two new series of tests whose the results are collected in Table 6. Other abbreviations concerning Table 6 are DT and OSM; they denote respectively the distance threshold and similarity measure (based on Eq. 6).
As can be seen (Table 6), in all cases, i.e., with the similarity measure proposed (OSM) or with other measures (ED and [3]' SM), clustering time remains practically the same when the similarity threshold changes (increases or decreases). Indeed, our clustering is based on the "minimum distance" as a "criterion for aggregation". In other words, the number of comparisons is practically the same for each threshold distance value.
As for differences, there is a lag in response times and differences between the similarities' values obtained. The difference in response times, as expected, is obvious, given the differences between the equations used by all three measures tested. The time parameter is not very restrictive and should not weigh heavily on the feasibility of such applications (clustering is not an interactive application where time is always critical parameter). Note, however, that differences in the values of the similarities are crucial, since it is on the basis of similarity that it is decided that a document is or is not assigned to a cluster. Moreover, these differences have a direct impact on the number of clusters (NC) obtained in each test. Indeed, with these thresholds, some documents are structurally very distant to stay together in the same cluster. This is due to [3]' similarity measure and our similarity measure that take into account the ancestor and descendant context of nodes, so that we find in the same cluster as the documents having very close hierarchical structures. Thus, XML documents that do not satisfy this condition, i.e., that are not sufficiently structurally similar, will migrate to other newly created clusters. In fact, these new clusters are considered as not corresponding to any DTD. We recall that each DTD is considered as target class against which we can assess our clustering.

Table 6: Our similarity measure versus [3]'measure and edit distance measure

| DT | [3]' SM | | ED | | OSM | |
|---|---|---|---|---|---|---|
| | NC | Time | NC | Time | NC | Time |
| 0.9 | 1 | $8.61\times10^2$ | 1 | $0.97\times10^2$ | 1 | $3.49\times10^2$ |
| 0.8 | 1 | $8.75\times10^2$ | 1 | $0.97\times10^2$ | 1 | $3.49\times10^2$ |
| 0.7 | 2 | $8.75\times10^2$ | 1 | $1.06\times10^2$ | 2 | $3.51\times10^2$ |
| 0.6 | 2 | $8.78\times10^2$ | 1 | $1.06\times10^2$ | 2 | $3.51\times10^2$ |
| 0.5 | 3 | $8.81\times10^2$ | 2 | $1.07\times10^2$ | 4 | $3.54\times10^2$ |
| 0.4 | 5 | $8.88\times10^2$ | 4 | $1.09\times10^2$ | 5 | $3.57\times10^2$ |
| 0.3 | 5 | $8.88\times10^2$ | 4 | $1.09\times10^2$ | 5 | $3.57\times10^2$ |
| 0.2 | 7 | $8.91\times10^2$ | 5 | $1.11\times10^2$ | 5 | $3.57\times10^2$ |
| 0.1 | 8 | $8.96\times10^2$ | 6 | $1.13\times10^2$ | 7 | $3.61\times10^2$ |

When we compare the results in Tables 5 and 6, there are some differences. Indeed, if we consider the column named "[3]' SM" in the two tables in question, we find that there is a clear difference in the clustering time. This is certainly due to our clustering algorithm, which is faster compared to the algorithm of the study by [3], which is relatively slow.
The number of clusters NC does not change rapidly with the distance threshold in Table 5 compared to NC in Table 6. This is due to clustering algorithms that are different. The clustering algorithm used in this article is a simple algorithm based on a conventional agglomerative hierarchical classification, while the clustering approach by [3] is an incremental clustering. Our clustering

algorithm uses only the minimum distance as aggregation criterion, while the clustering approach by [3] is characterized by the mobility of the centroid, representing each cluster. Each time an XML document must be added, and its representative is systematically compared with all existing centroids and all the trees in the cluster to which it is assigned. During the comparison process, we can either have a new centroid, which is systematically assigned to a newly created cluster, or an existing centroid can be replaced by another tree more representative, among those of the same cluster. This clearly explains the differences between the two approaches particularly regarding the time variation in the clustering (SM column) in Table 5.
If we compare the column OSM in Table 6, representing our approach, we see that it is somewhat close to the result of the ED column in Table 5, in terms of clustering time and the number of clusters NC. But it is somewhat far from the result of the ED column in Table 6, particularly in terms of time clustering.
We conclude that our method is better because it is reliable in terms of clustering time and the quality of clustering.

### *4.4.2 Comparison of some of our results with those of other clustering methods*

The second test is to compare some of our results with those obtained in [3, 6, 13, 31, 38] approaches on a portion of ACM SIGMOD collection. To this end we used the sample of XML documents in Table 7.

Table 7: Distribution of ACM Sigmod record subset

| Name of the DTD | Number of XML documents |
|---|---|
| IndexTermsPage | 52 |
| OrdinaryIssuePage | 30 |
| ProceedingsPage | 16 |
| SigmodRecord | 1 |
| HomePage | 1 |

We chose to compare our method with those developed in [3, 6, 13, 31, 38] approaches for several raisons. First, as our approach, these approaches use very close representation, namely tree structures to structurally represent XML documents. Second, because they use the same data set, namely ACM SIGMOD corpus. Third, their clustering methods are all based on edit distance or similarity that is different from our measure of similarity. Recall that these results do not depend only on the similarity measure, but also and especially, of the model (original XML tree structure or XML tree structure summary) used to represent the structures of XML documents. In Table 8, we can see the results of this comparison. Note that [3, 6, 13, 31, 38] results were reported in [3, 38]. These values represent the average *Precision*, *Recall*, and *F-measure*, in the interval [0, 1].
The results in Table 8 show that our clustering has a slightly lower *precision* than those of [6, 13, 38]' approaches, but it is very close to those of [3, 31]'approaches. But it nevertheless has a better *Recall* than the majority of other approaches, with the exception of that of the [3]' approach.
Finally, the *F-measure* obtained by our clustering also seems to be higher than all others, with the exception of that of the approach [3]. However, our clustering is better overall, since it has a better *Precision* than that of the approach [3].

Table 8: Comparison of our results with those of other approaches

| Approach | *Precision* | *Recall* | *F-measure* |
|---|---|---|---|
| [3] | 0.78 | 0.97 | 0.86 |
| [6] | 0.86 | 0.57 | 0.68 |
| [13] | 0.86 | 0.67 | 0.68 |
| [31] | 0.81 | 0.64 | 0.72 |
| [38] | 0.90 | 0.61 | 0.73 |
| Our approach | 0.82 | 0.71 | 0.76 |

### *4.4.3 Time needed to calculate the structural similarity between two XML documents*

Finally, as expected, in this third test, we will conduct experiments to determine the time required to calculate the structural similarity between two XML documents.
To conduct these experiments, we generated a set of 10 synthetic XML documents whose the number of nodes varies respectively from 50 to 500.
We conducted two sets of tests with the group of XML documents previously generated:

– The first one was conducted by setting the values of the weights $w_1$, $w_2$ and $w_3$ to $\frac{1}{3}$ in Eq. (2). As we have already considered, it is more natural, in the case of XML documents, to give to $S_1$, $S_2$ and $S_3$ the same weight, namely $w_1 = \frac{1}{3}$, $w_2 = \frac{1}{3}$, and $w_3 = \frac{1}{3}$. Recall that $S_2$ and $S_3$ represent respectively ascendant (ancestor) and descendant contexts. For more details see the equations for calculating the similarity.
– The second one was conducted by setting the values of the weights $w_1 = 1$, $w_2 = w_3 = 0$, in the same equation. In other words, we ignore the hierarchical contexts (descendant levels and ancestors levels). Therefore, it is not necessary to calculate $S_2$ and $S_3$. In this case our similarity measure behaves like the edit distance. Thus, the time complexity of calculating the similarity between two trees $T_1$ and $T_2$ is in the worst case $O(N^2)$.

What matters in this test is not the quality of clustering, but the time required for comparison of two XML documents' structures. Therefore, it is not necessary to have summaries of XML trees. For this, we slightly modified our parser, so as not remove repetitions of sibling nodes and thus to obtain the original XML tree structures (the whole structure of document). For more details about this question, see Subsection 3.1.

We recall that the timing experiments were carried out on a Lenovo PC Intel (R) Core (TM) 2 Duo CPU, 2.00 GHz for each Processor, and 2.99 GB of RAM.

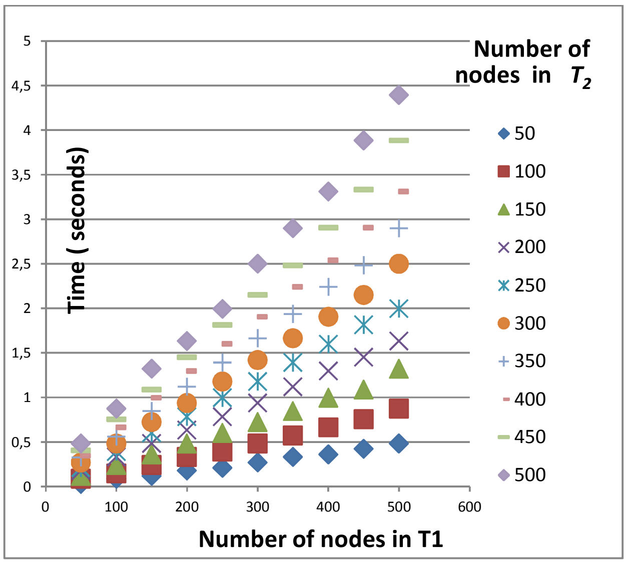


Fig. 16 Timing results with weights $w_1 = w_2 = w_3 = \frac{1}{3}$

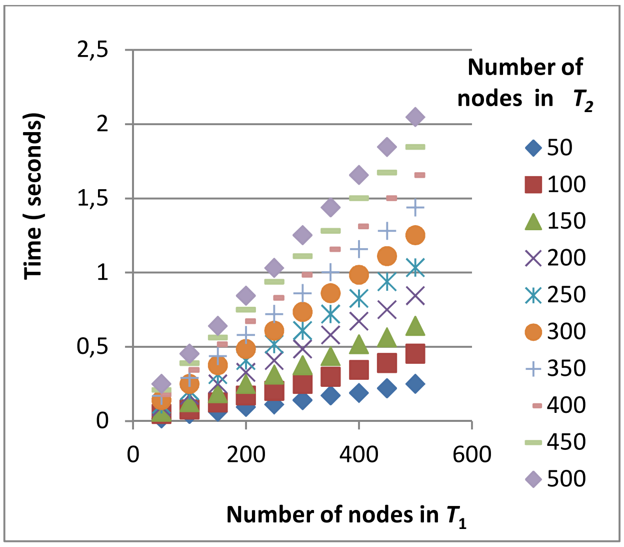


Fig. 17 Timing results with weights $w_1 = 1, w_2 = w_3 = 0$

The graphs in Figs. 16 and 17 express the results of timing experiments performed on the group of 10 previously generated XML documents.

In the first case, namely with weights $w_1 = w_2 = w_3 = \frac{1}{3}$ as shown in Fig. 16, the time required for the similarity calculation in the case of our approach is between the time required to calculate the similarities in the approaches [6, 13, 31, 38] and [3]. This time seems much closer to the time needed in the approaches [6, 13, 31, 38]. Indeed, according to Fig. 16, we can see that the time required for calculating the similarity between two trees $T_1$ and $T_2$ of various sizes grows in quasi linear fashion with tree size (of each tree).

In the second case, however, as shown in Fig. 17, namely with $w_1 = 1$, $w_2 = w_3 = 0$, our similarity calculation algorithm behaves like the algorithm of [6, 13, 31, 38], i.e., with a time complexity of $O(N^2)$. The time needed to find the similarity between pairs of trees of various sizes increases in linear fashion with tree size (of each tree).

Under all these experiments and according to the results obtained in all previous tests, we can say that our proposed similarity measure and our way of representing XML documents are very relevant particularly since it provides high-quality clustering.

## 5. Conclusion

We have proposed an approach for representing XML documents by their respective structures. We have particularly shown how to extract the tree structure of each XML document to be classified. We also proposed an efficient similarity measure (which is the primary purpose in this article), and an algorithm for clustering these structures. The clusters containing XML structures classified are generated through a conventional agglomerative hierarchical technique.

Our approach touches on two interesting fundamental aspects of the Information Retrieval. Indeed, on the one hand, the clustering allows to reduce the number of treated documents and finally to increase the number of the relevant documents returned by the search engine. On the other hand, the clusters obtained can constitute an interface allowing users to access XML documents they wish to query and to reach the specific “information units” that interest them.

The experiment conducted is a small outline for testing the feasibility and reliability of our approach. However, to perform a good experimentation, it is judicious to prepare tests on larger collections of documents.

## References

[1] A. Aïtelhadj, M. Mezghiche, F. Souam, "Classification de Structures Arborescentes: Cas de Documents XML", In Proceedings of the 6th French Information Retrieval Conference, CORIA 2009, Presqu’île de Giens, France, 5–7 May 2009, pp. 301–317.
[2] A. Aïtelhadj, F. Souam, M. Mezghiche, "XML Documents Clustering Based on Structural Similarity", In Proceedings of the IADIS international conference on WWW/INTERNET, Rome, Italy 19–22 November 2009, pp.559–566.
[3] A. Aïtelhadj, M. Boughanem, M. Mezghiche, F. Souam "Using structural similarity for clustering XML", Knowl Inf Syst, 32 (1), 2012, pp. 109–139.

[4] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, Arikawa S, "Efficient substructure discovery from large semi-structured data", In Proceedings of the 2nd SIAM international conference on data mining, Arlington, Virginia, VA, USA, 11–13 April 2002, pp. 158–174.

[5] E. Bertino, G. Guerrini, M. Mesiti "A matching algorithm for measuring the structural similarity between an XML documents and a DTD and its applications", Inf Syst, 2004, 29(1), pp. 23–46.

[6] S. Chawathe, "Comparing hierarchical data in external memory", In Proceedings of the 25th international conference on very large data bases, VLDB 1999, Edinburgh, Scotland, UK, 7–10 September 1999, pp. 90–101.

[7] S. Chawathe, Garcia-Molina H, "Meaningful change detection in structured data", In Proceedings of the 1997 ACM SIGMOD international conference on management of data, Tuscon, Arizona, May 1997, pp. 26–37.

[8] S. Chawathe, R. Rajaraman, H. Garcia-Molina, J. Widom, "Change detection in hierarchically structured information", In Proceedings of the 1996 ACM SIGMOD international conference on management of data, Montreal, Quebec, June 1996, pp. 493–504.

[9] G. Cobena, S. Abiteboul, A. Marian, "Detecting changes in XML documents", In Proceedings of the 18th international conference on data engineering, ICDE 2002, San Jose, California, 26 February–1 March 2002, pp. 41–52.

[10] T. Cormen, C. Leiserson, R. Rivest, Introduction to algorithms, MIT Press, 1990.

[11] G. Costa, R. Manco, R. Ortale, A. Tagarelli, "A tree-based approach to clustering XML documents by structure", In Proceedings of the 8th European conference on principles and practice of knowledge discovery in databases, PKDD 2004 Pisa, Italy, September 2004, pp.137–148.

[12] J. Cui, H. Liu, J. He, P. Li, X. Du, W. Puwei, "TagClus: a random walk-based method for tag clustering", Knowl Inf Syst, May 2011, 27(2), pp.193–225.

[13] T. Dalamagas, T. Cheng, K.J. Winkel, T.K. Sellis, "A methodology for clustering XML documents by structure", Inf Syst, 2006, 31(3), pp. 187–228.

[14] T. Dalamagas, T. Cheng, K.J. Winkel, T.K. Sellis, "Clustering XML documents using structural summaries", In EDBT 2004 (Extending Database Technology) international workshop on clustering information over the web, Heraklion, Crete, Greece, 14 March 2004, pp. 547–556.

[15] Del Razo Lopez F, Larent A, Poncelet P, Teisseire M "Recherche de sous-structures fréquentes pour l'intégration de schémas XML", In Proceedings of the 6th international French-speaking conference on knowledge discovery and management, EGC 2006, Lille, France, 17 January 2006, pp.487–498.

[16] L. Denoyer, "Apprentissage et inférence statistique dans les bases de documents structurés: Application aux corpus de documents textuels", Ph.D thesis, Université Paris 6, France, 2004.

[17] H.G. Direen, M.S Jones., "Knowledge management in bioinformatics", In A. B. Chaudhri, A. Rashid, and R. Zicari, editors, XML Data Management, 2003, Addison Wesley.

[18] A. Doan, P. Domingos, A. Halevy, "Reconciling Schemas of Disparate Data Sources: a machine-Learning approach", In Proceedings of the 2001 ACMSIGMOD international conference on management data, ACM New York, NY, USA, June 2001, pp. 509–520.

[19] A. Doucet, M. Lehtonen, "Unsupervised classification of text-centric XML document collections", In Proceedings of the 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, 17–20 December 2006, pp. 497–509.

[20] S. Flesca, G. Manco, E. Masciari, L. Pontieri, A. Pugliese, "Fast detection of XML structural similarity", IEEE Trans Knowl Data Eng, 17(2), pp. 160 –175.

[21] F.D. Francesca, R. Gordano, R. Ortale, A. Tagarelli "Distance-based clustering of XML documents", In Proceedings of the first international workshop on mining graphs, trees and sequences, MGTS 2003, Cavtat-Dubrovnik, Croatia, 22–26 September 2003, pp. 75–78.

[22] J.C. Gower, G.J.S. Ross, "Minimum spanning trees and single linkage cluster analysis", Applied Statistics, 1969, 18(1), pp. 54–64.

[23] Y Guo, D. Chen, J. Le, "Clustering XML documents by combining content and structure", In Proceedings of the 2008 international symposium on information science and engineering, ISISE 2008, Shanghai, China, 20–22 December 2008, published by IEEE Computer Society Washington, DC, USA 2008, pp. 583–587.

[24] M. Hagenbuchner, A. Sperduti, A.C. Tsoi, F. Trentini, F. Scarselli, M. Gori, "Clustering XML documents using self-organizing maps for structures", In Proceedings of the 4th international workshop of the initiative for the evaluation of XML retrieval, INEX 2005, Dagstuhl Castle, Germany, 28–30 November 2005, pp. 481–496.

[25] M. KcM, M. Hagenbuchner, A.C. Tsoi, F. Scarselli, M. Gori, A. Sperduti, "XML document mining using contextual self-organizing maps for structures", In Proceedings of the 5th international workshop of the initiative for the evaluation of XML retrieval, INEX 2006, Dagstuhl Castle, Germany, 17–20 December 2006, pp. 510–524.

[26] B. Larsen, C. Aone, "Fast and effective text mining using linear-time document clustering", In Proceedings of the 5th ACM SIGKDD international conference on knowledge discovery and data mining KDD 99, San Diego, California, CA, USA 15–18 August 1999, publisher: ACM New York, NY, USA 1999, pp. 16–22.

[27] J. Madhavan, A.P. Bernstein, E. Rahm, "Generic schema matching with cupid", In Proceedings of the 27th international conference on very large data bases, VLDB 2001, Roma, Italy, September 2001, pp. 49–58.

[28] L.L Mong, Y. Liang Huai, H. Wynne, Y. Xia, "XClust: Clustering XML schemas for effective integration", In Proceedings of the 11th ACM CIKM international conference on information and knowledge management, CIKM 2002, Mclean, Virginia, USA, 4–9 November 2002, pp. 292–299.

[29] R. Nayak, "Fast and effective clustering of XML data using structural information", Knowl Inf Syst, 2008, 14(2), pp. 197–215.

[30] R. Nayak, Xu S, "XML documents clustering by structures", In Proceedings of the 4th international workshop of the initiative for the evaluation of XML retrieval, INEX 2005, Dagstuhl Castle, Germany, 28–30 November 2005, pp. 432–442.

[31] A. Nierman, H.V. Jagadish, "Evaluating structural similarity in XML documents", In Proceedings of the fifth international workshop on the web and databases Web DB 2002, Madison, Wisconsin, USA, 6–7 June 2002, Publisher: Citeseer, pp. 61–66.
[32] D. Patnaik, S. Laxman, N. Ramakrishnan, "Discovering excitatory relationships using dynamic Bayesian networks", Knowl Inf Syst, 2011, 29(2), pp. 273-303.
[33] R.C. Prim, "Shortest connection networks and some generalizations", Bell System Technical Journal, 1957, 36, pp. 1389–1401.
[34] D. Sankoff, J. Kruskal, "Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison", CSLI Publications, 1999, D. Sankoff and J. Kruskal, eds., pp. 11–44.
[35] S.M. Selkow, "The tree-to-tree editing problem", Inf Process Lett, February 1977, 6(6), pp. 184–186.
[36] H. Su, S. Padmanabhan, "Identification of syntactically similar DTD elements in schema matching across DTDs", In Proceedings of the 2th international conference on web-age information management, WAIM 2001, Xi'an, China, 9–11 July 2001, LNCS Springer 2118, 2001, pp. 145–159.
[37] K.C. Tai, "The tree-to-tree correction problem", J ACM (JACM), 1979, 26(3), pp. 422–433.
[38] J. Tekli, R. Chbeir, K. Yetongnon, "Efficient XML Structural similarity detection using sub-tree commonalities", In Proceedings of the 22nd Brazilian symposium on databases, SBBD 2007, Joao Pessoa, Paraiba, Brasil, 15–19 October 2007, pp. 116–130.
[39] A. Termier, M.C. Rousset, M. Sebag, "Tree finder: a first step towards XML data mining", In Proceedings of the 2002 IEEE international conference on data mining, ICDM 2002, 9–12 December 2002, Maebashi City, Japan, published by IEEE Computer Society, 2002, pp. 450–457.
[40] Tran T, Nayak R, "Evaluating the performance of XML document clustering by structure only", In Proceedings of the 5th international workshop of the initiative for the evaluation of XML retrieval, INEX 2006, Dagstuhl Castle, Germany, 17–20 December 2006, pp. 473–484.
[41] A.M. Vercoustre, M. Fegas, S. Gul, Y. Lechevallier, "A flexible structured-based representation for XML document mining", In Proceedings of the 4th international workshop of the initiative for the evaluation of XML retrieval, INEX 2005, Dagstuhl Castle, Germany, 28–30 November 2005, pp. 443–457.
[42] Wang J, Zhang K, Jeong K, Shasha D, "A system for approximate tree matching", IEEE Trans Knowl Data Eng, 1994, 6(4), pp. 559–571.
[43] G. Wisniewsky, L. Denoyer, P. Gallinari, "Classification automatique de documents structurés: Application au corpus d'arbres étiquetés de type XML", In Proceedings of the 2nd French information retrieval conference, CORIA 2005, Grenoble, France, 9–11 march 2005, pp. 167–184.
[44] S.L. Yong, M. Hagenbuchner, A.C. Tsoi, F. Scarselli, M. Gori, "XML document mining using graph neural network", In Proceedings of the 5th international workshop of the initiative for the evaluation of XML retrieval, INEX 2006, Dagstuhl Castle, Germany, 17–20 December 2006, pp. 458–472.
[45] M.J. Zaki, "Efficient mining frequent trees in a forest", In Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD 2002, ACM New York, NY, USA 2002, pp. 71–80.

**Dr. Ali Aïtelhadj** received his Magister degree in Computer Science from Tizi-Ouzou University, Algeria in 2006 and Ph.D degree from Boumerdes University, Algeria in 2011. Currently, he is working as Assistant Professor (Maître de conférences) in the Department of Computer Science, Faculty of Electrical Engineering and Computer Science at the University of Tizi-Ouzou. He is the author and co-author in several international conferences and journals. His main research topics are focused on Classification, IR models and XML information retrieval. His other research areas include Advanced Data Structures and Algorithms, Evolutionary computing, and Data Mining.