THE NATIONAL UNIVERSITY
of SINGAPORE

School of Computing
Computing 1, 13 Computing Drive, Singapore 117417

TRB6/11

Edit Distance between XML and Probabilistic XML Documents

Ruiming Tang, Huayu Wu, Sadegh Nobari and Stephane Bressan

June 2011

T e c h n i c a l R e p o r t

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

OOI Beng Chin
Dean of School

Edit Distance between XML and Probabilistic XML Documents

Ruiming Tang, Huayu Wu, Sadegh Nobari, and Stéphane Bressan

School of Computing, National University of Singapore
{tangruiming, wuhuayu, snobari, steph}@comp.nus.edu.sg

Abstract. We propose an efficient algorithm for computing of the edit distance between an XML document and a probabilistic XML document. Probabilistic XML is a hierarchical data model capturing uncertainty of both value and structure. It is suitable to many modern applications such as information extraction, scientific data management and data integration. The computation of similarity is an essential building block for the comparison, alignment, clustering and classification of data in these applications. Several algorithms exist for measuring the structural similarity between XML documents among themselves or XML documents and XML document type definitions and schemas. The new challenge in efficiently computing the similarity between an XML document and a probabilistic XML document is the multiplicity of the possible worlds that a probabilistic XML document represents. In this paper, we devise and discuss algorithms for computing the similarity between an XML document and a probabilistic XML document. We empirically and comparatively evaluate their performance. In the absence of established corpora and benchmarks for probabilistic XML, we also propose and use random probabilistic XML models together with the associated random generation algorithms.

1 Introduction

1.1 Motivation

Modern applications wanting to exploit the now available numerous data sources face the challenge posed by uncertainty of information. Unfortunately there is little room for uncertainty in traditional database models and management systems. New models, new tools and techniques are needed.

Several probabilistic extensions to the relational models have been proposed (see [10, 14, 3, 28, 15]). However hierarchical models in general and XML in particular provide further features, such as semi-structured-ness and schemalessness, that are crucial to the applications that we are discussing.

Probabilistic XML is a hierarchical data model capturing uncertainty of both value and structure. There exist several variants of the model (see [24, 1]). Probabilistic XML is suitable to applications such as sensor networks, information extraction, scientific data management and data integration (see [36] and [45], for instance).

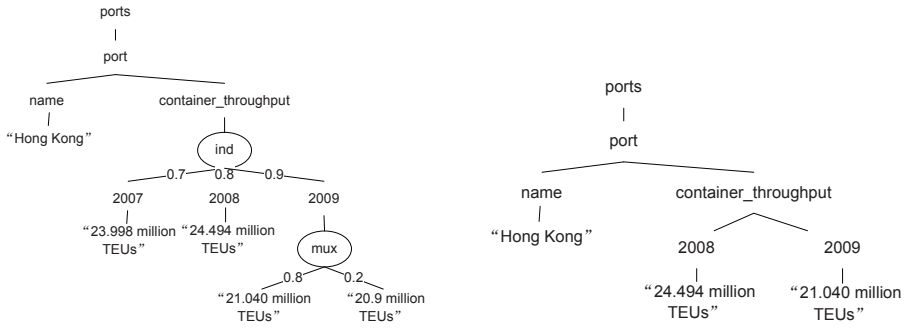
The computation of similarity is an essential building block for the tasks at hand, namely comparison, alignment, clustering and classification of data. Several algorithms exist for measuring the structural similarity between XML documents among themselves or XML documents and XML document type definitions and schemas. The new challenge in efficiently computing the similarity between an XML document and a probabilistic XML document is the multiplicity of the possible worlds that a probabilistic XML document represents.

In this paper, we devise and discuss algorithms for computing the similarity between an XML document and a probabilistic XML document. The algorithms implement the expected value of the edit distance between an XML document and the documents in the set defined by the probabilistic XML document. We empirically and comparatively evaluate their performance.

In the absence of established corpora and benchmarks for probabilistic XML, we also propose and use several random probabilistic XML models together with the associated random generation algorithms.

1.2 Probabilistic XML

Figure 1 shows a probabilistic XML document in the model of [36]. A probabilistic XML document is an XML document with special nodes. In the figure the special nodes are labeled "mux" and "ind". These nodes are called *distributional nodes*. They have a special interpretation.



(a) one probabilistic XML document (b) one possible world

Fig. 1. A Probabilistic XML Document and one of its Possible World

A distributional node labeled "mux" denotes a mutually exclusive choice among its children (or none of the children if the sum of probabilities is less than one) with a probability associated to each child. We refer to these distributional nodes as "mutually exclusive distributional nodes". A distributional node labeled "ind" denotes an independent choice among its children with a probability associated to each child. It denotes the mutually exclusive choice of one subset of the children (possibly empty). We refer to these distributional nodes as "independent distributional nodes".

In this paper we only consider probabilistic XML documents with mutually exclusive and independent distributional nodes. Other models introduce two

additional distributional nodes. A distributional node labeled "exp" denotes a mutually exclusive choice among selected subsets of its children with a probability associated to these subsets as indicated in an ancillary data structure. A distributional node labeled "cie" denotes an independent choice among its children according to a Boolean expression associated to each child. The expression is the conjunction of positive and negative literals called events. The truth value of each event is associated to a probability, globally in the document. The distributional node with its Boolean expression denotes the mutually exclusive choice of one subset of the children (possibly empty); yet, this choice is coordinated among the distributional nodes referring to the same events.

A probabilistic XML document denotes the set of XML documents corresponding to the possible replacement of the distributional nodes by one of the choice that they denote. Each of this XML document is associated with the induced probability of the corresponding choices.

In the example, the mutually exclusive distributional node below the node labeled "2009" indicates a choice between a child "21.040 million TEUs" with probability 0.8 and "20.9 million TEUs" with probability 0.2. The independent distributional node below the node labeled 'container_throughput" indicates an independent choice between zero to three children among "2007", "2008" and "2009" with the respective probabilities.

In general, a probabilistic XML document U is a random model of XML documents. It is a probability space that defines an assignment of probabilities to all ordinary XML documents or, in other words, a probability distribution over XML documents. Each XML document with a non zero probability, $Pr(d) > 0$, is called a *possible world* of the probabilistic document U. We say that a possible world d, has a probability $Pr(d)$ to belong to $pwd(U)$, representing the set of possible worlds defined by the probabilistic XML document U. Clearly $\sum_{d \in pwd(U)} Pr(d) = 1$.

The XML document in Figure 1.b is one possible world of the probabilistic XML document in Figure 1.a. The probability of this possible world to belong to the model is $0.8 \times 0.9 \times (1 - 0.7) \times 0.8 = 0.1728$. There are 12 possible worlds in the model of the XML document in Figure 1.a.

1.3 Plan

In the following, related work is synthesized in Section 2. We define similarity and present the algorithms for its computation in Section 3. Section 4 presents and discusses the results of the empirical and comparative performance evaluation of our algorithms. It also discusses the generation of synthetic probabilistic XML data. Finally, we conclude in Section 5.

2 Related Work

2.1 Uncertain Data Models

Uncertainty is introduced for the relational model by the authors of [10]). In [10] and [14], the authors introduce a data model in which point-valued probability

measures are assigned to tuples. The probability measure indicates the joint probability of all the attribute values in the tuple. In [3], the authors propose a model in which point-valued probability measures are assigned to individual attribute values or to sets of attribute values resulting in a nested relational model. [28] presents interval-valued probability measures are assigned to sets of attribute values, which is also a nested relational model. The choice of interval-valued probabilities helps capture the error in measurement approaches. In the data model of [15] interval-valued probability measures are assigned to tuples.

[1] surveys the different probabilistic XML data models that have been proposed in the literature. These models differ in the distributional nodes that they consider: mutually exclusive choices, independent choices, explicit choices and conjunction of independent events. Following [1], we refer to these models according to the distributional nodes that they consider. The mostly used models are $PrXML^{mux}$, $PrXML^{mux,ind}$, $PrXML^{exp}$ and $PrXML^{cie}$, respectively.

Table 1 summarizes the introduction and adoption of these models by the different papers that we discuss in this subsection.

models \ papers	[24]	[36]	[2]	[13]	[20]	[19]	[45]	[26]	[25]
$PrXML^{mux}$	$\surd$	$*$							$\surd$
$PrXML^{mux,ind}$		$*$	$\surd$	$\surd$			$\surd$	$\surd$	$\surd$
$PrXML^{exp}$					$*$	$\surd$			$\surd$
$PrXML^{cie}$			$*$						

Table 1. Introduction ($*$) and Adoption ($\surd$) of Models

The authors of [36] introduce the $PrXML^{mux}$ and $PrXML^{mux,ind}$ models, the latter being more expressive than the former. They propose query processing techniques for these models. [24] uses existing query processing techniques for uncertain XML to mine uncertain XML data. They use the $PrXML^{mux}$ model. The authors of [2] introduce the notion of independent events and the $PrXML^{cie}$ model. They show the added expressiveness of $PrXML^{cie}$ over $PrXML^{mux,ind}$. They define query evaluation and update techniques for both $PrXML^{mux,ind}$ and $PrXML^{cie}$. In [13] a notion of constraint in the $PrXML^{mux,ind}$ model is discussed . The authors shows the tractability of query evaluation including aggregate queries in this model. The authors of [20] introduce the $PrXML^{exp}$ model together with semantics, algebra and efficient algorithms for query processing. They look at extending the model to interval probability values in [19]. The model proposed in [45] is similar to $PrXML^{mux,ind}$. It is used to integrate disparate data and comes with a naïve query processing algorithm. The authors of [26] study the evaluation of twig queries for probabilistic XML in the $PrXML^{mux,ind}$ model. They broaden their study to $PrXML^{mux}$, $PrXML^{mux,ind}$, $PrXML^{exp}$, and $PrXML^{cie}$ in [25].

There seems to be a natural tradeoff between the expressiveness of a model and the efficient of query evaluation. $PrXML^{mux}$ is the least expressive model while the most expressive are $PrXML^{cie}$ and $PrXML^{exp}$. Their combination $PrXML^{cie,exp}$ subsumes all the other models. We believe that $PrXML^{mux,ind}$

strikes a good balance between the expressiveness and efficiency. This model is also the most commonly adapted.

One question remains critically unanswered in the above research publications: where are the probabilistic XML documents? We believe indeed that until the tools and techniques for probabilistic XML are developed to a satisfactory level we will not see applications adopting probabilistic XML models and generating probabilistic XML documents. It is therefore difficult to empirically evaluate the tools and techniques in the absence of corpora from these applications. This is a chicken and egg situation. A closer look at the papers that we have reviewed above reveals that the experimental data used for the performance evaluation and analysis are mostly ad hoc. Only the authors of [45] consider a practical information integration scenario and propose to derive the probabilities from weights assigned to the different data sources and from statistics computed from the data; yet they use two makeshift documents for the evaluation of their proposal.

We therefore turn to the random generation of probabilistic XML documents for the evaluation of the performance of our proposed algorithms. XML application benchmarks such as [41], [8], [7], and [48], as well as the [39] micro benchmark come with synthetic data generators. ToXGene [4] is a template-based generator for synthetic XML documents. It allows the controlled generation of XML documents of all size and shapes. Our approach is one of random generation as in the data generators of the benchmarks and of ToXGene.

2.2 Similarity

Similarity metrics and distances attempt to quantify the resemblance of different objects. The edit distance, introduced for strings in [32] and [46], quantifies this resemblance as the minimum cost of transforming one object into another. The lower the cost is, the more similar the objects. For instance, in the case of strings, the string edit distance is the minimum number of insertion, deletion or replacement of characters that can rewrite one string into another. Different edit distances can be defined by allowing different sets of elementary operations and giving them different individual costs.

The idea of a tree edit distance is introduced by Tai in [43] in 1979. Zhang and Shasha [49, 42] propose several definitions and algorithms for edit distances of ordered and unordered trees. In this paper we consider the tree edit distance proposed in [11] that allows update of any vertex in the tree but restricts deletions and insertions to leaves. The tree alignment metrics of the family of the one proposed by Jiang et al. in [21] are alternatives to the edit distance.

The different tree edit distances and tree alignment metrics have naturally been applied to define and compute the similarity of XML documents [35, 12]. However other distances and similarity metrics have been proposed and used for XML documents that are based on edge matching and counting [27, 33], path matching and counting [9, 38, 22], entropy [40, 18] and Fourier transform [16]. More creative and application specific metrics have been proposed in [34, 23, 47, 30]. Research on information retrieval for XML documents(e.g. [17]) has defined

similarity metrics inspired by those commonly used in information retrieval. They have, for instance, adapted term and document frequencies and the vector space model to XML. In addition, one can remark that XML structured queries (in Xpath or XQuery, for instance), DTDs and XML schemas denote sets of XML documents: the answers to a query, the valid documents. Therefore different authors have studied the similarity between XML documents and queries, XML documents and DTDs or schemas, as well as among queries or DTDs or schemas themselves. The similarity to a set of XML documents can be defined as the minimum, maximum, average or any other aggregate of the individual similarities.

The authors of [44] use tree edit distance to measure the similarity between an XML document and a DTD. In [6], the authors define an alignment-based ranked matching between an XML document and a DTD. [5, 31, 29] propose methods to measure the similarity among DTDs.

The above similarity definitions and the algorithms to effectively and efficiently compute them have been used for approximate query answering(e.g. [17]), data mining, classification and clustering [35, 27, 33, 9, 38, 22, 40, 18, 16, 44, 6], change detection [12], data integration [34, 23, 47, 30] and schema integration [5, 31, 29] among many others applications.

3 Proposal

3.1 Membership

Before we discuss the computation of the edit distance between an XML document and a probabilistic XML document, let us first introduce a different but somehow preliminary problem. We refer to this problem as the "membership problem". Given an XML document d and a probabilistic XML document U, the membership problem consists in computing $Pr(d)$ in the probability space defined by U. The naïve way to compute $Pr(d)$ is to enumerate $pwd(U)$, the set of possible worlds denoted by U together with their associated probabilities and to look up for d and $Pr(d)$ in $pwd(U)$. However the number of possible worlds can potentially combinatorially explode. Algorithm 1 is instead a level by level matching algorithm. It is a recognizer for the documents in the set of possible worlds.

3.2 Edit Distance

The edit distance between two trees is the minimum cost of the operations needed to transform one tree into another. In this paper we consider the definition of [11] in which three unit cost operations are considered: deletion of a leaf, insertion of a leaf and update of a node anywhere in the tree. In an XML document we define the elements to be the nodes and leaves. We refer to this edit distance between two XML document as δ_d. δ_d can be relatively efficiently computed using dynamic programming.

Algorithm 1: Membership algorithm

Data: U : one probabilistic XML document, *doc* : one XML document
Result: the probability Pr

```
1  generate all possible mappings in the current level;
2  if no such mapping then
3  |  stop;
4  for each mapping_i do
5  |  p_i=1;
6  |  for each distributional node in this level do
7  |  |  p_i=p_i×distributional-node-probability
8  |  for each correspondence of mapping_i do
9  |  |  recursively call this procedure;
10 Pr=Pr+p_i;
11 Return Pr;
```

A probabilistic XML document is a probability space. The edit distance between an XML document d and a probabilistic XML document U can be defined as the minimum, maximum or average distance between the XML document and the possible worlds (recall that a possible world is an XML document). In this paper we choose to define it as the expected value of the edit distance between the XML document and the possible worlds. We refer to this edit distance as δ_p. It is the sum of the tree edit distances between the XML document and the possible worlds weighted with the probabilities of the possible worlds, as given in equation 1.

$$\delta_p(d, U) = \sum_{w \in pwd(U)} Pr(w) \times \delta_d(d, w) \tag{1}$$

In this paper we use and adapt the tree edit distance and the algorithm of [11]. With the operations that we consider two trees can be compared using the respective sequences of their nodes in preorder while carefully forbidding the insertion and deletion otherwise than at the leaves. The technique and the data structure, called the "edit graph" in [11], is itself an extension to trees to the original dynamic programming approach and the matrix data structure used in [46] for string edit distance.

Next we present three algorithms to compute the edit distance between an XML document and a probabilistic XML document.

3.3 Enumeration algorithm

As with membership edit distance can be computed by enumeration of the possible worlds. The enumeration algorithm, E,presented in Algorithm 2, generates the possible worlds for the probabilistic XML document U with their respective probabilities (as in the naïve membership algorithm). Then, it computes the edit distance between the XML document and each of the possible world. Finally it sums the products of the distances with the respective probabilities. However, as

Algorithm 2: Enumeration algorithm

Data: *doc* : an XML document, U : one probabilistic XML document
Result: expected edit distance between U and *doc*

```
1 generate possible worlds d_i with their probability value p_i from p ;
2 for each possible world d_i do
3 |   compute edit distance between d_i and doc with probability as the weight:
  |   TED_i = TED(d_i, doc) × p_i
4 sum up all the weighted edit distance ∑_i TED_i;
```

we shall see this approach is as prohibitive here as it is for membership. Unfortunately, all the pairwise distances between the XML document and the possible worlds are needed.

3.4 Multidimensional Algorithm

The main idea for an efficient edit distance algorithm for probabilistic XML is to share common computations. Indeed, while distributional nodes correspond to branching to possible worlds, all other nodes in a probabilistic XML are normal XML nodes and can be processed as such. The algorithm, that we call the "multidimensional algorithm" or MA for short, copies the dynamic programming sub matrices when it encounters distributional nodes. It branches the computation to the possible worlds or dimensions. This allows the sharing of computation until the next distributional node.

The algorithm is further improved by noticing that the dynamic programming approach does not require to memorizing the entire matrix but rather one column at a time. The MA algorithm is presented in Algorithm 3. The matrix is initialized at line 1. The special case of distributional nodes is handled from line 3 to line 7 while line 8 to line 10 handle ordinary XML nodes.

Algorithm 3: MA algorithm

Data: *doc* : an XML document, U : one probabilistic XML document
Result: expected tree edit distance between U and *doc*

```
1  construct one matrix to store information for computation;
2  for each node of U in preorder traverse do
3  |   if distributional node then
4  |   |   compute all the cases of this distributional node;
5  |   |   for each case do
6  |   |   |   keep one copy of the current matrix;
7  |   |   |   process the next node;
8  |   else
9  |   |   update the matrix only with the latest column;
10 |   |   process the next node;
11 get all the produced matrix and compute final result;
```

3.5 Stack algorithm

Figure 2 illustrates the successive data structures needed in the computation of the edit distance between an XML document and a probabilistic XML document.

The figure immediately suggests reconsidering the strategy of the MA algorithm. Instead of memorizing the possible worlds and copying the columns of the dynamic programming matrix we can compute the distance by a depth first strategy that only requires remembering the data structures at the branching points in the tree. This is classically done with a stack. Algorithm 4 illustrate this algorithm that we call MAS.

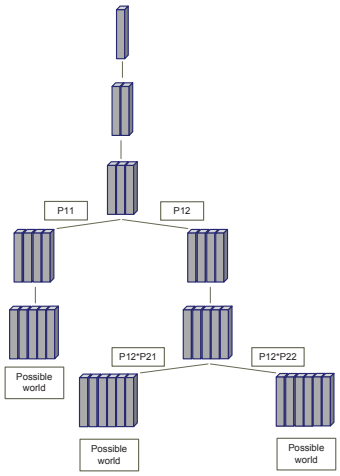


Fig. 2. MA and MAS Data Structure

The matrix is initialized at line 1. The special case of distributional nodes is handled from line 4 to line 8 while line 9 to line 13 handle ordinary XML nodes. The algorithm visits the probabilistic XML document in the preorder. If the current node is a distributional node it pushes the current data structures into a stack (line 5), and continue processing each of the possible worlds (line 6 to line 7). Otherwise (handled in line 9 to line 13) it normally compute the edit distance for the current possible world.

4 Performance evaluation

4.1 Experimental setup

The three algorithms are implemented in Java. All experiments are run on a Centos 5.5 2 x Quad-Core Xeon E5520 2.2GHz with 24.0GB RAM and 3 x 500GB SATA hard disk.

The experiments are conducted with synthetic data. In order to control the generation of synthetic data we propose three random models for probabilistic XML: $P_X(doc, n)$, $P_X(doc, p)$ and $P_X(doc, p, f)$. These random probabilistic

Algorithm 4: MAS algorithm

Data: *doc* : an XML document, U : one probabilistic XML document
Result: expected tree edit distance between U and *doc*

```
construct one matrix to store information for computation;
for each node of U in preorder traverse do
    if not reaching the end then
        if distributional node then
            push the current matrix, all possible cases into stack M, stack Poss;
            for each case do
                process the first child in the case and pop Poss.top;
            pop M.top;
        else
            if x ∈ Poss.top then
                update the matrix and process the next node;
            else
                process the next non-descendant node;
    else
        record the result for this finished instance
```

XML models are probability spaces of XML documents (probability spaces of probability spaces).

A probabilistic XML document in the $P_X(doc, n)$ model is derived from the original XML document *doc* by inserting independently n internal distributional nodes.

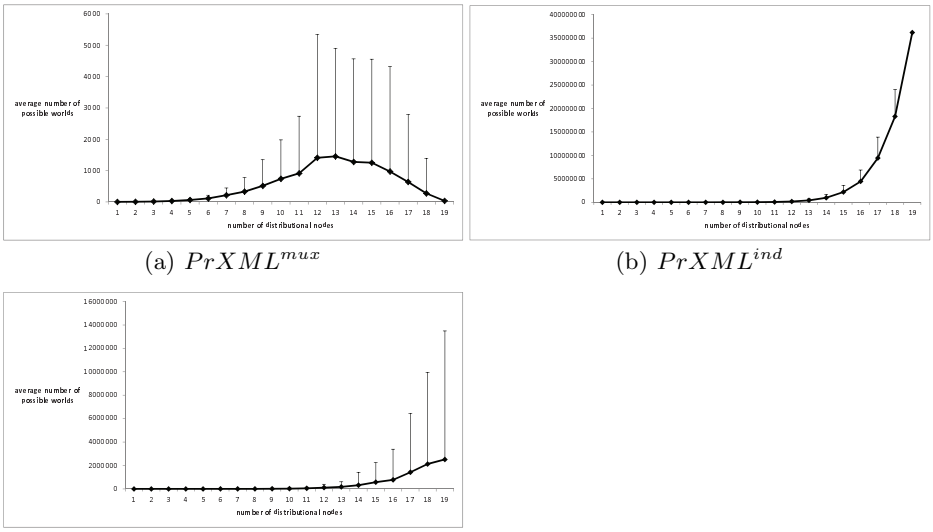
A probabilistic XML document in the $P_X(doc, p)$ model is derived from the original XML document *doc* by inserting independently distributional nodes below each internal node with probability p.

The reader has noticed the resemblance of these random models with their graph counter parts: the Erdös-Rényi models [37].

Finally, in order to control the depth at which distributional nodes appear, we propose the $P_X(doc, p, f)$ model in which a probabilistic XML document is derived from the original XML document *doc* by inserting independently distributional nodes below each internal node with probability $p \times f(q)$, where q is the level of the node in the XML tree. For each of these models we devise an algorithm that generates uniformly at random a probabilistic XML document in the model.

We use these three models and the corresponding algorithms for our experiments. In the experiments *doc* is a synthetic XML document (s.xml) generated using ToXGene [4]. We could use a real XML document but preferred a synthetic one in order to better control its features. The features of the generated document are as follows. It has 43 nodes, 19 internal nodes, a maximum depth of 4, an average depth of 2.558, a maximum number of children for the internal nodes of 4, and an average number of children for the leaf nodes of 2.211.

4.2 Experiments



(a) $PrXML^{mux}$ (b) $PrXML^{ind}$

(c) $PrXML^{mux,ind}$

Fig. 3. number of possible worlds as a function of the distributional nodes

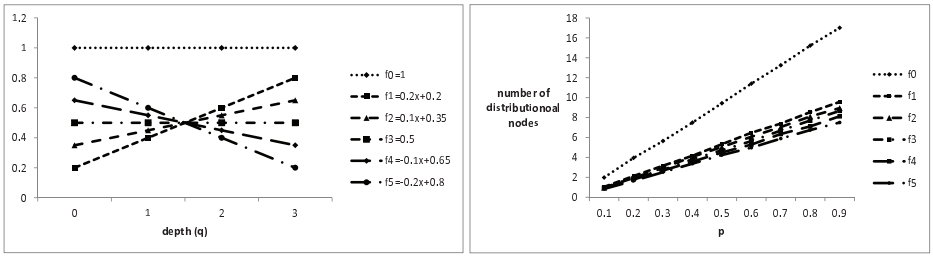
Possible Worlds In this first experiment we measure the number of possible worlds as a function of the distributional nodes. We consider the three $PrXML^{mux}$, $PrXML^{ind}$ and $PrXML^{mux,ind}$ models.

We generate 1000 probabilistic XML documents (there may not be so many different possible probabilistic XML documents, however by generating so many we get a better uniformity of the sample) in the $P_X(doc, n)$ model for n varying from 1 to 19. We plot the average value and its standard deviation.

Figure 3.a shows that, in the case of $PrXML^{mux}$, the number of possible words increases until it reaches a maximum ($n = 13$) and decreases again. This is because as the number of distributional nodes increases beyond 13 these modes are more likely to be nested and generate less possible worlds. Figure 3.b shows that, in the case of $PrXML^{ind}$, the number of possible worlds increases. There is no peak. This is because even if distributional nodes of type "ind" are nested the number of possible worlds that they generate is still combinatorial. Figure 3.c shows that, in the case of $PrXML^{ind,mux}$, the number of possible worlds increases. There is no peak. This is because the effect of distributional nodes of type "ind" dominates the effect of distributional nodes of type "mux".

Number of Distributional Nodes In this second experiment we measure the number of distributional nodes as a function of the probability of the nodes to be inserted at certain levels.

We generate 1000 probabilistic XML documents in the $P_X(doc, p, f)$ model for p varying from 0.1 to 0.9. We plot the average value for the six functions



(a) six different functions

(b) number of distributional nodes against p with different functions

Fig. 4. number of distributional nodes as a function of the probability

f_0, f_1, f_2, f_3, f_4 and f_5 of Figure 4.a. Notice that in the cases of f being a constant function ($f_0(q) = 1$ and $f_3(q) = 0.5$) $P_X(doc, p, f)$ is $P_X(doc, p \times f)$ and, therefore is in the $P_X(doc, p)$ model.

Figure 4.b shows that for the same compound probability (sum of the probabilities for each level: f_1 to f_5) the larger the slope the more distributional nodes. The case of f_0 also shows that the larger the probabilities the more distributional nodes.

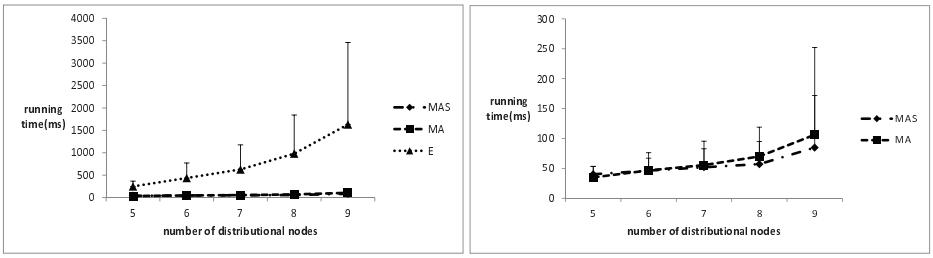
Running Time In this third experiment we measure the running time as a function of the number of distributional nodes. We consider the three $PrXML^{mux}$, $PrXML^{ind}$ and $PrXML^{mux,ind}$ models.

We generate 1000 probabilistic XML documents in the $P_X(doc, n)$ for n varying from 5 to 9. We plot the average value and its standard deviation for the three algorithms: enumeration algorithm, E, multidimensional algorithm, MA, and the stack algorithm, MAS.

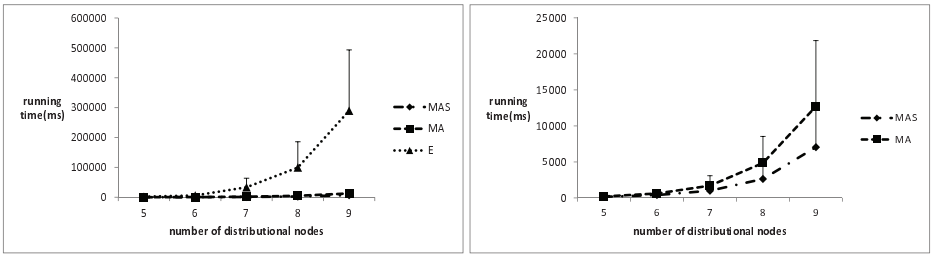
The average running time is increasing with the number of distributional nodes. On figure 5.a,c,e we see that the MA and MAS are significantly faster than E. On figure 5.b,d,f, zooming in the performance of MA and MAS, we see that the stack algorithm is faster. Although MA and MAS are just a breadth-first and depth-first traversal of the same tree, respectively, there is overhead in copying and memorizing the data structures in MA as opposed to pushing and popping the strictly necessary ones to and from the stack in MAS. MAS is the most efficient algorithm for computing the edit distance between an XML document and a probabilistic XML document.

5 Conclusion

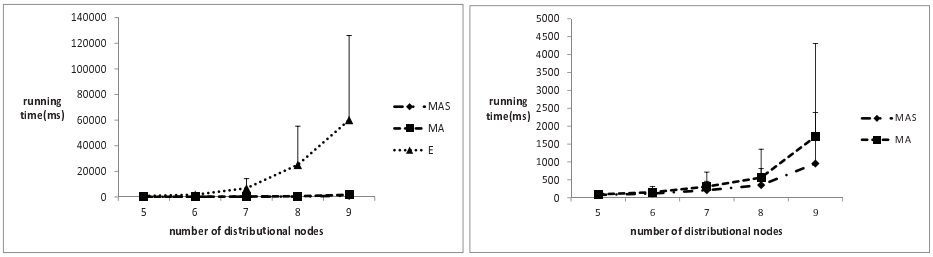
We have defined the similarity between an XML and a probabilistic XML document as the expected value of the edit distance between the XML document and the set of XML documents denoted by the probabilistic XML document. We have devised an original dynamic programming algorithm, MAS, and empirically



(a) running time of $PrXML^{mux}$ (b) zooming in (a)



(c) running time of $PrXML^{ind}$ (d) zooming in (c)



(e) running time of $PrXML^{mux,ind}$ (f) zooming in (e)

Fig. 5. running time as a function of the number of distributional nodes

shown that it outperforms the baseline approaches. In the absence of established corpora and benchmarks for probabilistic XML, we have also proposed several random probabilistic XML models and their corresponding random generation algorithms for the generation of synthetic probabilistic XML data.

References

1. S. Abiteboul, B. Kimelfeld, Y. Sagiv, and P. Senellart. On the Expressiveness of Probabilistic XML models. *VLDB J.*, 18(5):1041–1064, 2009.
2. S. Abiteboul and P. Senellart. Querying and Updating Probabilistic Information in XML. In *EDBT*, pages 1059–1068, 2006.
3. D. Barbará, H. Garcia-Molina, and D. Porter. The Management of Probabilistic Data. *IEEE Trans. Knowl. Data Eng.*, 4(5):487–502, 1992.

4. D. Barbosa, A. O. Mendelzon, J. Keenleyside, and K. A. Lyons. ToXgene: An Extensible Template-Based Data Generator for XML. In *WebDB*, pages 49–54, 2002.
5. D. Basci and S. Misra. Entropy Metric for XML DTD Documents. *ACM SIGSOFT Software Engineering Notes*, 33(4), 2008.
6. E. Bertino, G. Guerrini, and M. Mesiti. A Matching Algorithm for Measuring the Structural Similarity between an XML Document and a dtd and its applications. *Inf. Syst.*, 29(1):23–46, 2004.
7. T. Böhme and E. Rahm. Multi-user Evaluation of XML Data Management Systems with XMach-1. In *EEXTT*, pages 148–158, 2002.
8. S. Bressan, M.-L. Lee, Y. G. Li, Z. Lacroix, and U. Nambiar. The XOO7 Benchmark. In *EEXTT*, pages 146–147, 2002.
9. D. Buttler. A Short Survey of Document Structure Similarity Algorithms. In *International Conference on Internet Computing*, pages 3–9, 2004.
10. R. Cavallo and M. Pittarelli. The Theory of Probabilistic Databases. In *VLDB*, pages 71–81, 1987.
11. S. S. Chawathe. Comparing Hierarchical Data in External Memory. In *VLDB*, pages 90–101, 1999.
12. G. Cobena, S. Abiteboul, and A. Marian. Detecting Changes in XML Documents. In *ICDE*, pages 41–52, 2002.
13. S. Cohen, B. Kimelfeld, and Y. Sagiv. Incorporating Constraints in Probabilistic XML. In *PODS*, pages 109–118, 2008.
14. D. Dey and S. Sarkar. A Probabilistic Relational Model and Algebra. *ACM Trans. Database Syst.*, 21(3):339–369, 1996.
15. T. Eiter, T. Lukasiewicz, and M. Walter. A Data Model and Algebra for Probabilistic Complex Values. *Ann. Math. Artif. Intell.*, 33(2-4):205–252, 2001.
16. S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese. Detecting Structural Similarities between XML Documents. In *WebDB*, pages 55–60, 2002.
17. N. Fuhr and K. Großjohann. XIRQL: An XML Query Language Based on Information Retrieval Concepts. *ACM Trans. Inf. Syst.*, 22(2):313–356, 2004.
18. S. Helmer. Measuring the Structural Similarity of Semistructured Documents Using Entropy. In *VLDB*, pages 1022–1032, 2007.
19. E. Hung, L. Getoor, and V. S. Subrahmanian. Probabilistic Interval XML. In *ICDT*, pages 358–374, 2003.
20. E. Hung, L. Getoor, and V. S. Subrahmanian. PXML: A Probabilistic Semistructured Data Model and Algebra. In *ICDE*, pages 467–479, 2003.
21. T. Jiang, L. Wang, and K. Zhang. Alignment of Trees - An Alternative to Tree Edit. *Theor. Comput. Sci.*, 143(1):137–148, 1995.
22. S. Joshi, N. Agrawal, R. Krishnapuram, and S. Negi. A Bag of Paths Model for Measuring Structural Similarity in Web Documents. In *KDD*, pages 577–582, 2003.
23. A. M. Kade and C. A. Heuser. Matching XML Documents in Highly Dynamic Applications. In *ACM Symposium on Document Engineering*, pages 191–198, 2008.
24. E. Kharlamov and P. Senellart. Modeling, Querying, and Mining Uncertain XML Data. In A. Tagarelli, editor, *XML Data Mining: Models, Methods, and Applications*. IGI Global, 2011.
25. B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query Efficiency in Probabilistic XML models. In *SIGMOD Conference*, pages 701–714, 2008.
26. B. Kimelfeld and Y. Sagiv. Matching Twigs in Probabilistic XML. In *VLDB*, pages 27–38, 2007.
27. H.-P. Kriegel and S. Schönauer. Similarity Search in Structured Data. In *DaWaK*, pages 309–319, 2003.

28. L. V. S. Lakshmanan, N. Leone, R. B. Ross, and V. S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. *ACM Trans. Database Syst.*, 22(3):419–469, 1997.
29. M.-L. Lee, L. H. Yang, W. Hsu, and X. Yang. XClust: Clustering XML Schemas for Effective Integration. In *CIKM*, pages 292–299, 2002.
30. L. Leitão, P. Calado, and M. Weis. Structure-Based Inference of XML Similarity for Fuzzy Duplicate Detection. In *CIKM*, pages 293–302, 2007.
31. E. Leonardi, T. T. Hoai, S. S. Bhowmick, and S. K. Madria. DTD-Diff: A Change Detection Algorithm for DTDs. *Data Knowl. Eng.*, 61(2):384–402, 2007.
32. V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.
33. W. Lian, D. W.-L. Cheung, N. Mamoulis, and S.-M. Yiu. An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. *IEEE Trans. Knowl. Data Eng.*, 16(1):82–96, 2004.
34. W. Liang and H. Yokota. LAX: An Efficient Approximate XML Join Based on Clustered Leaf Nodes for XML Data Integration. In *BNCOD*, pages 82–97, 2005.
35. A. Nierman and H. V. Jagadish. Evaluating Structural Similarity in XML Documents. In *WebDB*, pages 61–66, 2002.
36. A. Nierman and H. V. Jagadish. ProTDB: Probabilistic Data in XML. In *VLDB*, pages 646–657, 2002.
37. S. H. Nobari, X. Lu, P. Karras, and S. Bressan. Fast Random Graph Generation. In *EDBT*, 2011.
38. D. Rafiei, D. L. Moise, and D. Sun. Finding Syntactic Similarities Between XML Documents. In *DEXA Workshops*, pages 512–516, 2006.
39. K. Runapongsa, J. M. Patel, H. V. Jagadish, and S. Al-Khalifa. The Michigan Benchmark: A Microbenchmark for XML Query Processing Systems. In *EEXTT*, pages 160–161, 2002.
40. I. Sanz, M. Mesiti, G. Guerrini, and R. B. Llavori. An Entropy-Based Characterization of the Heterogeneity of XML Collections. In *DEXA Workshops*, pages 238–242, 2008.
41. A. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, and R. Busse. Assessing XML Data Management with XMark. In *EEXTT*, pages 144–145, 2002.
42. D. Shasha and K. Zhang. Fast Algorithms for the Unit Cost Editing Distance Between Trees. *J. Algorithms*, 11(4):581–621, 1990.
43. K.-C. Tai. The Tree-to-Tree Correction Problem. *J. ACM*, 26(3):422–433, 1979.
44. J. Tekli, R. Chbeir, and K. Yétongnon. Structural Similarity Evaluation Between XML Documents and DTDs. In *WISE*, pages 196–211, 2007.
45. M. van Keulen, A. de Keijzer, and W. Alink. A Probabilistic XML Approach to Data Integration. In *ICDE*, pages 459–470, 2005.
46. R. A. Wagner and M. J. Fischer. The String-to-String Correction Problem. *J. ACM*, 21(1):168–173, 1974.
47. M. Weis and F. Naumann. DogmatiX Tracks down Duplicates in XML. In *SIGMOD Conference*, pages 431–442, 2005.
48. B. B. Yao, M. T. Özsu, and J. Keenleyside. XBench - A Family of Benchmarks for XML DBMSs. In *EEXTT*, pages 162–164, 2002.
49. K. Zhang and D. Shasha. Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.