論文 / 著書情報
Article / Book Information

論題(和文)	
Title(English)	SLAX: An Improved Leaf-Clustering Based Approximate XML Join Algorithm for Integrating XML Data at Subtree Classes
著者(和文)	梁 文新, 横田 治夫
Authors(English)	Wenxin Liang, Haruo Yokota
出典(和文)	情報処理学会論文誌データベース, Vol. 47, No. SIG 8 (TOD30), pp. 47-57
Citation(English)	IPSJ Transaction on Database, Vol. 47, No. SIG 8 (TOD30), pp. 47-57
発行日 / Pub. date	2006, 6
権利情報 / Copyright	

Regular Paper

SLAX: An Improved Leaf-Clustering Based Approximate XML Join Algorithm for Integrating XML Data at Subtree Classes

Wenxin Liang[†] and Haruo Yokota[†,††]

XML is widely applied to represent and exchange data on the Internet. However, XML documents from different sources may convey nearly or exactly the same information but may be different on structures. In previous work, we have proposed *LAX* (**L**eaf-clustering based **A**pproximate **X**ML join algorithm), in which the two XML document trees are divided into independent subtrees and the approximate similarity between them are determined by the tree similarity degree based on the mean value of the similarity degrees of matched subtrees. Our previous experimental results show that *LAX*, comparing with the tree edit distance, is more efficient in performance and more effective for measuring the approximate similarity between XML documents. However, because the tree edit distance is extremely time-consuming, we only used bibliography data of very small sizes to compare the performance of *LAX* with that of the tree edit distance in our previous experiments. Besides, in *LAX*, the output is oriented to the pair of documents that have larger tree similarity degree than the threshold. Therefore, when *LAX* is applied to the fragments divided from large XML documents, the hit subtree selected from the output pair of fragment documents that has large tree similarity degree might not be the proper one that should be integrated. In this paper, we propose *SLAX* (**S**ubtree-class **L**eaf-clustering based **A**pproximate **X**ML join algorithm) for integrating the fragments divided from large XML documents by using the maximum match value at subtree classes. And we conduct further experiments to evaluate *SLAX*, comparing with *LAX*, by using both real large bibliography and bioinformatics data. The experimental results show that *SLAX* is more effective than *LAX* for integrating both large bibliography and bioinformatics data at subtree classes.

1. Introduction

The Extensible Markup Language (XML) is widely applied to represent and exchange data on the Internet, because it can represent different kinds of data from multiple sources. Nowadays more and more data, especially bioinformatics such as Swiss-Prot [15], TrEMBL [17] and bibliography data such as DBLP [22] and ACM SIGMOD Record [1], are published and exchanged by XML on the Internet. However, XML documents from different data sources might contain nearly or exactly the same information but might be constructed by different structures. Besides, even the two XML documents convey the same contents, both of them may have some extra information what the other does not do. Therefore, it becomes important that an algorithm can efficiently measure the similarity between XML documents for integrating such data sources so that the users can conveniently access and acquire more complete information.

The Document Type Descriptor (DTD) is recognized as a helpful tool to detect the structural information from XML documents [2),6),7),13)]. However, even if XML documents have the same DTDs, they may not have identical tree structures because of the repeating and optional elements and attributes [9),10),13)]. **Figure 1** shows an example of two XML documents (sample fragments from the XML files of Swiss-Prot [20] and TrEMBL [21], respectively) with very similar DTDs. Although these two documents are structurally different due to the repeating and optional elements, they express very similar information. Besides, each of the document has some information what the other does not do. For instance, `comment` in Fig. 1 (a); and `evidence` in Fig. 1 (b).

The tree edit distance is widely used as an effective metric for measuring the structural similarity between XML documents [9),13)]. However, it is a very expensive operation; in the worst case, the time complexity is $O(n^4)$ for the document of size n, where n is the number of the nodes of the document.

In previous work [11)], we have proposed *LAX*

† Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology

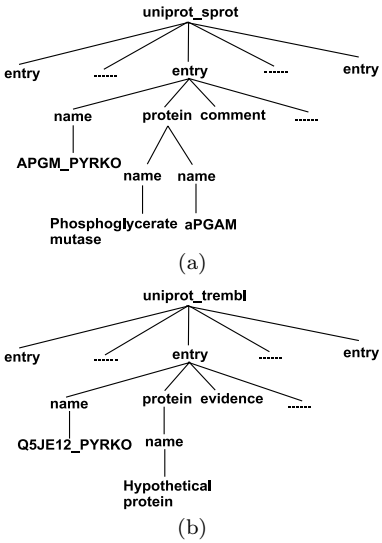


Fig. 1 Example XML documents.

(**L**eaf-clustering based **A**pproximate **X**ML join algorithm), in which the two XML document trees are divided into subtrees representing independent information units, and the output is oriented to the pair of documents that has larger tree similarity degree than the user-defined threshold. We have also proposed an effective algorithm for segmenting XML documents into independent subtrees for *LAX*. We have shown that *LAX* is more efficient in performance comparing to the tree edit distance, because it is an $O(n^2)$ operation for the document of size n in the worst case. Because the tree edit distance is extremely time-consuming, we only used bibliography data of very small sizes to compare the performance of *LAX* with that of the tree edit distance in our previous experiments. In addition, in order to compare with the work based on the tree edit distance [9),10)], the output of *LAX* is oriented to the pair of documents that have larger tree similarity degree (mean value of the similarity degrees of matched subtrees) than the threshold. The large XML documents sometimes can not be loaded into the main memory, they must be divided into small fragments. In this case, the target subtrees are distributed in each fragment document. Therefore, when *LAX* is applied to such fragment documents, the hit subtrees selected from the output pair of fragment documents that has large tree similarity degree might not be the proper one that should be integrated.

In this paper, we propose *SLAX* (**S**ubtree-class **L**eaf-clustering based **A**pproximate **X**ML join algorithm) to integrate fragments divided from large XML documents by using the maximum match value at subtree classes. We conduct further experiments to compare *SLAX* with *LAX* on the performance and effectiveness by using both large real bibliography and bioinformatics data. The experimental results indicate that the precision of subtree matching using *SLAX* is twice larger than that using *LAX* for bibliography data. And *SLAX* is useful and valuable for people to acquire similar proteins belonging to the same species, organisms, and so on, even if the two XML documents do not contain exactly the same proteins.

The rest of this paper is organized as follows. Section 2 briefly introduces the work related to the paper and the requirements for the issues of measuring approximate similarity between XML documents. In Section 3, we introduce the basic knowledge of *LAX*. Section 4 describes the problem of *LAX* when it handles fragments divided from large XML documents and proposes *SLAX* for solving this problem. In Section 5, we conduct experiments to compare *SLAX* with *LAX* by using both large real bioinformatics and bibliography data. In the end, Section 6 concludes the paper and outlines the future work.

2. Related Work and Requirements

A well formed XML document can be parsed into an ordered labeled tree [19)]. The tree structure represents the nesting of its elements, and node labels record the contents of the elements by element tags, attribute names, attribute values and PCDATA values.

Many researches have been done to solve the problem of measuring the edit distance between ordered labeled trees [3)~5),8),12),14),16),18),23)~25)]. The edit distance between two ordered labeled trees is defined as the minimum cost edit operations (insertions, deletions and substitutions) required to transform one tree to another. It can be figured out by a mapping between the nodes of the two trees [25)]. The tree edit distance is regarded as an effective metric for measuring the structural similarity in XML documents [8),9),13)]. However, its computational cost is extremely expensive; in the worst case, it is an $O(n^4)$ operation for the document of size n. Thus, it is of difficulty for the tree edit distance to handle the XML documents of large sizes.

In order to avoid the expensive tree edit

distance operation as much as possible, S. Guha, et al. proposed the lower and upper bound as inexpensive substitutions for the tree edit distance operation [9]. However, when the upper bound is greater than the threshold distance τ and meanwhile the lower bound is less than τ, the tree edit distance will still must be calculated. Therefore, we need a more efficient metric for measuring the approximate similarity between XML documents. As the matter of fact, many real XML documents are constructed by repeating elements, `entry` in the `uniprot_sprot.xml` [20] and `uniprot_tremb.xml` [21] for example. Such kind of XML documents can be segmented into subtrees representing independent units by rooting the subtrees at the repeating elements. The approximate similarity between the well-segmented documents can be effectively determined by computing the similarity degree based on the clustered leaf nodes of each pair of subtrees even without considering the structural and semantic heterogeneity.

3. Introduction of LAX

3.1 Overview

In previous work [11], we have proposed *LAX* for evaluating the approximate similarity between XML documents. In *LAX*, the two XML documents to be joined are segmented into subtrees representing independent information units. And the approximate similarity between them are determined by the tree similarity degree that is the mean value of the similarity degrees of the matched subtrees.

Notation. Let T_1 and T_2 be two XML document trees. Let T_1 be the base tree, and T_2 be the target one. Assume T_1 and T_2 are segmented into k_1 and k_2 sub-trees t_{1i} $(1 \leq i \leq k_1)$ and t_{2j} $(1 \leq j \leq k_2)$, respectively.

The *subtree similarity degree* between t_{1i} and t_{2j}, $S_S(t_{1i}, t_{2j})$ is defined by Equation (1) as the percentage of the number of matched leaf nodes (the pair of leaf nodes that has the same PCDATA value) out of the number of leaf nodes in the base subtree t_{1i}, where n and n_{1i} denote the number of matched leaf nodes and the number of leaf nodes in the base subtree t_{1i}.

$$S_S(t_{1i}, t_{2j}) = \frac{n}{n_{1i}} \times 100\ (\%) \qquad (1)$$

The *matched subtree*, $T_M[i]$ is defined as the pair of subtrees that has the maximum subtree similarity degree in one join loop; that is, the similarity degree of $T_M[i]$, $S_M[i]$ can be calculated as follows.

$$S_M[i] = Max_{j=1}^{k_2}(S_S(t_{1i}, t_{2j})) \qquad (2)$$

In the ith join loop, the matched subtree $T_M[i]$ is a *hit subtree*, iff $S_M[i] \geq \mathcal{T}$ $(0 < \mathcal{T} \leq 1)$, where $\mathcal{T}$ is the user defined threshold for the output subtrees.

The *tree similarity degree* between T_1 and T_2, $S_T(T_1, T_2)$ is determined by Equation (3) based on the mean value of the similarity degrees of matched subtrees.

$$S_T(T_1, T_2) = \frac{\sum_{i=1}^{k_1} S_M[i]}{k_1} \times 100\ (\%) \qquad (3)$$

3.2 Segmentation Algorithm

Many real XML documents are constructed by repeating elements, and they can be divided into independent subtrees at the repeating elements. However, there might be many different repeating elements with the same tag names in different levels. For instance, `entry` and `name` in Fig. 1 (a). Therefore, it is not an easy task to segment the XML document tree into subtrees at the proper positions. In previous work [11], we have proposed an effective algorithm for segmenting XML document trees, in which the spot for segmentation is determined by the *weighting factor* w. For a *candidate element* $E(n, d)$, where n denotes the number of candidate elements among its children, and d represents the the distance to its furthest descendant. The weighting factor w can be calculated by the following equation,

$$w = n \times d^{\theta} \qquad (0 < \theta \leq 1) \qquad (4)$$

where, θ is an adjustable constant.

3.3 Join Process

Let S_1 and S_2 be two XML data sources. Assume each document $d_1 \in S_1$ and $d_2 \in S_2$ are parsed into XML document trees T_1 and T_2. Let T_1 and T_2 be segmented into k_1 and k_2 subtrees t_{1i} and t_{2j}. Given a user-defined threshold τ, the join process of *LAX* is illustrated by *Algorithm LAX* shown in **Fig. 2**.

3.4 Comparison with Tree Edit Distance

For the document of size n, in the worst case, *LAX* is an $O(n^2)$ operation while the tree edit distance is an $O(n^4)$ one. Our previous experimental results indicate that as the size of document increases, *LAX* becomes overwhelmingly faster relative to the tree edit distance [11]. As for two XML documents with different DTDs that have the same number of nodes, the tree

```
Algorithm LAX {
Input: XML data source S1 and S2
Output: Pairs of XML documents (d1, d2)
  for each d1 ∈ S1 {
    parse d1 into T1;
    segment T1 into k1 subtrees;
    for each d2 ∈ S2 {
      parse d2 into T2;
      segment T2 into k2 subtrees;
      calculate ST(T1, T2);
    }
    if(ST(T1, T2) ≥ τ) {
      output (d1, d2);
    }
  }
}
```

Fig. 2 Algorithm *LAX*.

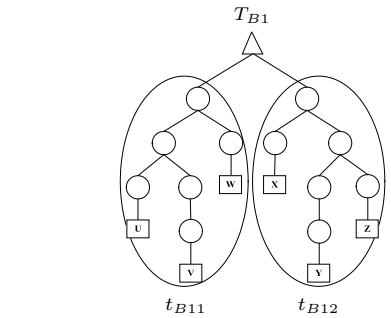
edit distance of them will not increase much when the PCDATA values of the leaf nodes change. While in *LAX*, the tree similarity degree will change in a large scale as the values of the leaf nodes change. Our previous experimental results show that *LAX* can effectively distinguish the similarity differences between XML documents even the tree edit distances of them are almost the same [11].

4. SLAX

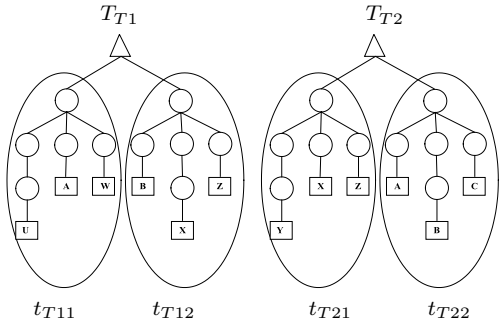
4.1 Problem of LAX

In this paper, we just use the original real large XML documents without storing them into RDBs. When the XML documents are too large to be loaded into the main memory, they must be divided into small fragments. In this case, the target subtrees are distributed in each fragment document. Besides, in *LAX*, the output is oriented to the pair of documents that have larger tree similarity degree (mean value of the similarity degrees of matched subtrees) than the threshold. Therefore, when *LAX* is applied to such fragment documents, the hit subtrees selected from the output pair of fragment documents that has large tree similarity degree might not be the proper one that should be integrated.

Example 1. *For the base document tree T_{B1}, and the target ones T_{T1} and T_{T2} divided from a large XML document tree T_T in* **Fig. 3**, *the tree similarity degrees $S_T(T_{B1}, T_{T1}) = 66.7\%$ and $S_T(T_{B1}, T_{T2}) = 50\%$. Assume the threshold for the output $0.5 < \tau \leq 0.667$, and the threshold for the hit subtree $\mathcal{T} \leq 0.667$. Because $S_T(T_{B1}, T_{T1}) = 66.7\% > \tau > S_T(T_{B1}, T_{T2}) = 0.5$, the document tree pair (T_{B1}, T_{T1}) will be output. Therefore, the subtree pair (t_{B12}, t_{T12}) $(S_S(t_{B12}, t_{T12}) = 66.7\% > \mathcal{T})$ will be selected as the hit subtree by LAX. However, the most proper subtree pair is actually the (t_{B12}, t_{T21}), because $S_S(t_{B12}, t_{T21}) = 100\% > S_S(t_{B12}, t_{T12}) = 66.7\%$.*



(a) Base document tree



(b) Target document tree

Fig. 3 Problem of *LAX*.

4.2 Key Definitions for SLAX

Before we propose the improved algorithm *SLAX* to solve the problem mentioned in Section 4.1, we present the following notations and definitions.

Notation. Let S_B and S_T be two XML data sources, where B denotes *base* and T denotes *target*. Assume each document $d_{Bm} \in S_B$ $(1 \leq m \leq K)$ and $d_{Tn} \in S_T$ $(1 \leq n \leq L)$ are parsed into XML document trees T_{Bm} $(1 \leq m \leq K)$ and T_{Tn} $(1 \leq n \leq L)$. Let T_{Bm} and T_{Tn} be segmented into k_B and k_T subtrees t_{Bmi} $(1 \leq i \leq k_B)$ and t_{Tnj} $(1 \leq j \leq k_T)$.

Definition 1 (Match Value). *The match value M[n] for the subtree t_{Bmi} and each target tree T_{Tn} $(1 \leq n \leq L)$ is defined as the the following equation.*

$$M[n] = Max_{j=0}^{k_T}(S_S(t_{Bmi}, t_{Tnj})) \qquad (5)$$

Definition 2 (Maximum Match Value). *The maximum match value $M_M[i]$ for the sub-*

```
Algorithm SLAX {
Input: XML data source S_B and S_T
Output: Pairs of subtrees (t_B, t_T)
  parse S_B into T_B;
  parse S_T into T_T;
  divide T_B into K fragment trees T_Bm;
  for (m = 1 to K) {
    segment T_Bm into k_B subtrees;
    for (i = 1 to k_B) {
      M_M[i] = 0;//Maximum match value for each base subtree t_Bmi
      divide T_T into L fragment trees T_Tn;
      for (n = 1 to L) {
        segment T_Tn into k_T subtrees;
        M[n]=0;//Match value for each target fragment tree T_Tn
        for (j = 1 to k_T) {
          calculate S_S(t_Bmi, t_Tnj);
          M[n] = Max(S_S(t_Bmi, t_Tnj), M[n]);
        }
        if(M[n] ≥ M_M[i]) {
          M_M[i] = Max(M[n], M_M[i]);
          N = n;//Record the index of the target fragment tree
        }
      }
      if (M_M[i] ≥ T){
        output(t_Bmi,t_TNj);
      }
    }
  }
}
```

Fig. 4 Algorithm *SLAX*.

tree t_{Bmi} is defined as following equation.

$$M_M[i] = Max_{n=0}^{L}(M[n]) \quad (6)$$

Definition 3 (Matched Tree). *The matched tree T_{Mi} for each subtree t_{Bmi} is defined as the target tree T_{Tn} that has the maximum $M[n]$.*

Definition 4 (Matched Pair). *The matched pair P_i for the subtree t_{Bmi} is defined as the pair of subtrees that has the maximum match value.*

Definition 5 (Hit Pair). *Given a threshold $\mathcal{T}$ ($0 < \mathcal{T} \leq 1$), the matched pair P_i is a hit pair P_{Hi}, if the maximum match value of P_i, $M_M[i] \geq \mathcal{T}$. The hit pair P_{Hi} should be output as the final result.*

4.3 Algorithm SLAX

Let S_B and S_T be two XML data sources, and each $d_{Bm} \in S_B$ and $d_{Tn} \in S_T$ be parsed into XML document trees T_{Bm} and T_{Tn}. Assume T_{Bm} and T_{Tn} are segmented into k_B and k_T subtrees t_{Bmi} and t_{Tnj}. Given a user-defined threshold $\mathcal{T}$, the algorithm *SLAX* is illustrated by **Fig. 4**.

Example 2. *For the base tree T_{B1} and the target ones T_{T1} and T_{T2} in Fig. 3, The matched trees T_{M1} and T_{M2} for the subtree T_{B11} and T_{B12} are T_{T1} and T_{T2}, respectively. And the matched pairs P_1 and P_2 of the subtree T_{B11} and T_{B12} are (t_{B11}, t_{T11}) and (t_{B12}, t_{T21}), respectively. Assume $\mathcal{T} < 0.667$, both of the matched P_1 and P_2 are the hit pairs and should be output as the final results.*

4.4 Comparison with *LAX*

The main differences between the improved algorithm *SLAX* and the original one *LAX* are summarized as follows:

- **Application Object** The application object for *LAX* is to measure the approximate similarity between XML documents, while that for *SLAX* is to detect the subtrees that represent the same or similar information in XML documents from different sources, so that the XML documents can be integrated at subtree classes.
- **Join Base and Target** The join base and target for *LAX* are oriented to XML documents, while those for *SLAX* are subtree-oriented.
- **Basis for Subtree Matching** In *LAX*, the subtree matching must be selected from the pair of XML documents that has larger tree similarity degree than the threshold. While in *SLAX*, the subtree matching is directly based on the maximum match value for the base subtree and the target one.

5. Experimental Evaluation

In our previous experiments [11], we have compared the performance of *LAX* with the tree edit distance for measuring the approximate similarity between XML documents. Because the tree edit distance is extremely time-consuming, we only used very small bibliography XML documents; the mean size of the fragment files of SIGMOD Record [1] and DBLP [22] we used was 4.32 KB (about 200 nodes). In this paper, we conduct experiments to compare *SLAX* with *LAX* by using different types of large XML data. We should have compared *SLAX* and *LAX* with the tree edit distance. However, the tree edit distance is too time-consuming to be applied to the large data. But anyway, our previous experimental results have shown that *LAX* is more efficient and effective for measuring the approximate similarity between XML documents than the tree edit distance even for small data [11].

In order to observe and compare how effectively *SLAX* and *LAX* determine the matched subtree by using the maximum match value and the tree similarity degree, respectively, we define the *precision of subtree matching*, P as follows.

Definition 6 (Precision of Subtree Matching). *The precision of subtree matching (P) is the percentage of the number of correctly matched subtree (N_c) out of the total number of subtrees (N)in the base document as the following equation.*

$$P = \frac{N_c}{N} \times 100(\%) \quad (7)$$

We use both real bibliography and bioinformatics data that are more than 100 times larger of sizes to evaluate *SLAX* and *LAX* in the following aspects.

- How do the document size and the number of the segmented subtrees impact the execution time of *SLAX* and *LAX* for different types of large XML data?
- What is the difference in the characteristic and performance for *SLAX* and *LAX* to integrate different types of large XML data?
- What is the difference in the precision of subtree matching for bibliography documents using the maximum match value and the tree similarity degree, respectively?

5.1 Data Used

5.1.1 Bibliography Data

The main characteristic of the bibliography document is that the number of leaf nodes of each segmented subtree is small. A bibliography XML document can be generally divided into a large number of subtrees representing a literature item such as an article and a book. The size of each subtree in the bibliography document is quite small. For example, the mean number of leaf nodes of an segmented subtree in SigmodRecord.xml [1] is only 5.8. In our experiments, we use SigmodRecord.xml (482 KB, about 20,000 nodes) and 955 fragments divided from DBLP.xml [22]. The size of each fragment is 300 KB (about 15,000 nodes).

5.1.2 Bioinformatics Data

Comparing with the bibliography data, bioinformatics data, such as protein data, contains a lot of information in each entry. Therefore, the size of each segmented subtree becomes much larger in the bioinformatics data. For instance, the mean number of leaf nodes of a 3 MB fragment of uniprot_sprot.xml [20] is 42.4.

Because the original protein XML documents are too big to be loaded into the main memory, we divide the uniprot_sprot.xml [20] and uniprot_trembl.xml [21] into fragment files. The size of each fragment of uniprot_sprot.xml [20] is 3 MB (about 80,000 nodes), and that of uniprot_trembl.xml [21] is 1 MB (about 25,000 nodes).

Table 1 Experimental environment.

CPU	AMD Athlon 64 3500+ 2.20 GHz
Memory	1.0 GB
OS	MS Windows XP Professional
Programming Environment	Sun JDK 1.4.2

Table 2 Precision of subtree mathcing for bibliography data.

	N_c	N_i	P
LAX	33	67	33%
SLAX	71	29	71%

5.2 Experimental Environment

The experiments have been done under the environment shown in **Table 1**.

5.3 Experimental Results

5.3.1 Bibliography Data

For the bibliography data, we take the SigmodRecord.xml as the base document, and all the 955 fragments of DBLP.xml, named DBLP1.xml-DBLP955.xml as the target ones. The number of segmented subtrees of the SigmodRecord.xml is 1504, and the mean number of those of the DBLP fragment files is 640. The mean execution time for handling each pair of documents by *SLAX* is 3.77 seconds.

We randomly sample 100 base subtrees from SigmodRecord.xml to compose a document, named sigmod.xml, using the same DTD as SigmodRecord.xml. Then, we join the sigmod.xml with the 955 framents of DBLP.xml by using the tree similarity degree and maximum match value, respectively. The number of correctly matched subtrees (N_c), the number of incorrectly matched subtrees (N_i) and the precision of subtree mathcing (P) using *LAX* and *SLAX* are shown in **Table 2**.

Here we show a real example of mismatching using *LAX*. We use SigmodRecord.xml as the base document, and randomly choose 40 fragments of DBLP.xml as the target ones. **Figure 5** shows the match values for the sample subtree and each fragment of DBLP.xml, and **Fig. 6** shows the source code of a sub-

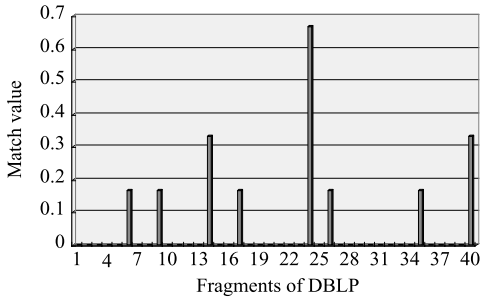


Fig. 5 Match values for the sample subtree and dblp1.xml-dblp40.xml.

```
<article>
<title>The IDEA Web Lab.</title>
<initPage>587</initPage>
<endPage>589</endPage>
<authors>
<author position="00">Stefano Ceri</author>
<author position="01">Piero Fraternali</author>
<author position="02">Stefano Paraboschi</author>
</authors>
</article>
```

Fig. 6 Sample subtree from sigmod.xml.

```
<inproceedings mdate="2002-01-03" key="conf/sigmod/CeriFP98">
<crossref>conf/sigmod/98</crossref>
<author>Stefano Ceri</author>
<author>Piero Fraternali</author>
<author>Stefano Paraboschi</author>
<title>The IDEA Web Lab.</title>
<pages>587-589</pages>
<year>1998</year>
<booktitle>SIGMOD Conference</booktitle>
<url>db/conf/sigmod/sigmod98.html#CeriFP98</url>
<ee>db/conf/sigmod/CeriFP98.html</ee>
<cdrom>SIGMOD98/P587.PDF</cdrom>
<cite>journals/jiis/BaralisCFP96</cite>
<cite>...</cite>
<cite>conf/dood/CeriFPP97</cite>
</inproceedings>
```

Fig. 7 The matched subtree selected by *SLAX* from dblp24.xml.

tree sampled from SigmodRecord.xml. The matched subtree for the sample subtree is selected by *SLAX* from dblp24.xml because of the maximum match value, and its source code is shown in **Fig. 7**. It is evident that the matched subtree determined by *SLAX* represents exactly the same article as the sample subtree. **Figure 8** shows the tree similarity degrees for SigmodRecord.xml and each fragment of DBLP.xml. Because the tree similarity degree between dblp14.xml and SigmodRecord is the largest among the 40 fragments, the number of hit pairs in dblp14.xml is larger than those in dblp24.xml; that is the number of similar subtrees in dblp14.xml is larger than that in dblp24.xml. However, the source code in **Fig. 9** shows that the matched subtree determined by *LAX* from dblp14.xml is not exactly the same

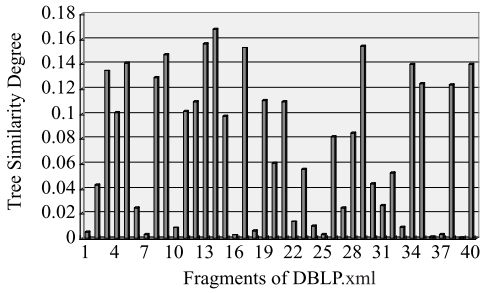


Fig. 8 Tree similarity degrees for SigmodRecord.xml and dblp1.xml-dblp40.xml.

```
<article mdate="2003-03-17" key="journals/cn/BonifatiCP02">
<author>Angela Bonifati</author>
<author>Stefano Ceri</author>
<author>Stefano Paraboschi</author>
<title>Pushing reactive services to XML repositories using active rules.</title>
<pages>645-660</pages>
<year>2002</year>
<volume>39</volume>
<journal>Computer Networks</journal>
<number>5</number>
<url>db/journals/cn/cn39.html#BonifatiCP02</url>
<ee>http://dx.doi.org/10.1016/S1389-1286(02)00226-8</ee>
</article>
```

Fig. 9 The matched subtree selected by *LAX* from dblp14.xml.

Table 3 The number of subtrees and the mean execution time.

	trembl1	trembl2	trembl3	trembl4	trembl5
N	324	349	389	414	327
T_S	3.53s	3.61s	3.70s	3.74s	3.55s
T_L	3.57s	3.69s	3.72s	3.75s	3.60s

article as the sample one but written by the same authors.

5.3.2 Bioinformatics Data

In order to learn how the document size and the number of segmented subtrees impact the excution time for *SLAX* and *LAX*, we randomly choose 5 fragments of uniprot_trebml.xml, named trembl1.xml-trembl5.xml, and join them with 40 fragments of uniprot_sprot.xml, named sprot1.xml-sport40.xml. N, T_S and $T_L$in **Table 3** indicate the number of subtrees in trembl1-5.xml and the mean execution time to join a pair of fragment files by *SLAX* and *LAX*, respectively.

Because the documents from TrEMBL and Swiss-Prot do not contain exactly the same protein information, it is of difficulty to quantitively define the precision of subtree matching for the protein data without expert knowledge. Therefore, in this paper we only conduct experiments to verify if our algorithm can find the similar protein from the target fragments for a

```
<entry dataset="TrEMBL" created="2005-05-10" modified="2005-05-10">
 <accession>Q5JE11</accession>
 <name>Q5JE11_PYRKO</name>
 <protein evidence="EI1">
  <name>Hypothetical protein</name>
 </protein>
 <gene>
  <name type="ordered locus" evidence="EI3">TK1070</name>
 </gene>
 <organism key="1">
  <name type="scientific">Pyrococcus kodakaraensis</name>
  <name type="synonym">Thermococcus kodakaraensis</name>
  <dbReference type="NCBI Taxonomy" id="69014" evidence="EI1" key="2"/>
  <lineage>
   <taxon>Archaea</taxon>
      .....
   <taxon>Thermococcus</taxon>
  </lineage>
 </organism>
 <reference evidence="EI1 EI3" key="3">
  .....
 </reference>
 <dbReference type="EMBL" id="AP006878" evidence="EI1" key="6">
  .....
 </dbReference>
  .....
</entry>
```

Fig. 10 Base subtree sampled from trembl4.xml.

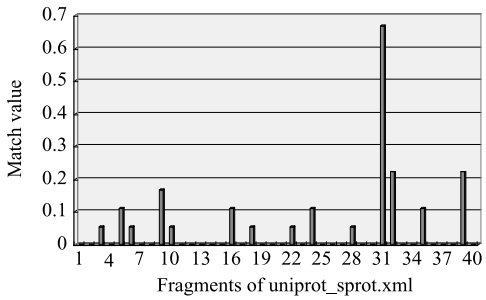


Fig. 11 Match values for trembl4.xml and the sprot1.xml-sprot40.xml.

sample protein in the base fragment.

We choose trembl4.xml as the base document and sprot1.xml-sprot40.xml as the target ones. The number of segmented subtrees of trembl4.xml is 414 and the mean number of those of sprot1-40.xml is 387. **Figure 10** shows the source code of the base subtree sampled from trembl4.xml, and **Fig. 11** indicates the match values for the sample subtree and sprot1-40.xml. The matched subtree for the sample one will be selected from sprot31.xml by *SLAX*. **Figure 12** shows the source code of the matched subtree determined by *SLAX*. From Fig. 10 and Fig. 12, we can observe that the matched subtree pair determined by *SLAX* represents very similar proteins belonging to the same organism. In respect of *LAX*, the tree similarity degrees for trembl4.xml and sprot1-40.xml are shown in **Fig. 13**. **Figure 14** shows the source code of the matched subtree for the sample one selected by *LAX* from sprot22.xml. It is apparent that the matched subtree determined by *LAX* is totally a different protein corresponding to the base one. Although

```
<entry dataset="Swiss-Prot" created="2005-05-10" modified="2005-09-13">
 <accession>Q5JI21</accession>
 <name>APGM_PYRKO</name>
 <protein ref="1">
   .....
  <name>aPGAM</name>
 </protein>
 <gene>
  <name type="primary">apgM</name>
  <name type="ordered locus">TK0866</name>
 </gene>
 <organism key="2">
  <name type="scientific">Pyrococcus kodakaraensis</name>
  <name type="synonym">Thermococcus kodakaraensis</name>
  <dbReference type="NCBI Taxonomy" id="69014" key="3"/>
  <lineage>
   <taxon>Archaea</taxon>
    .....
   <taxon>Thermococcus</taxon>
  </lineage>
 </organism>
 <reference key="4">
  .....
 </reference>
 <comment type="function" status="By similarity">
  .....
 </comment>
 <dbReference type="EMBL" id="AP006878" key="7">
  <property type="protein sequence ID" value="BAD85055.1"/>
 .....
 </dbReference>
 .....
</entry>
```

Fig. 12 The Matched subtree selected by *SLAX* from sprot31.xml.

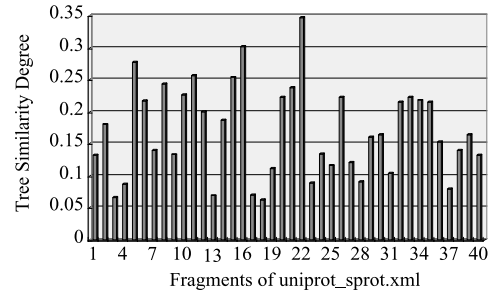


Fig. 13 Tree similarity degree for trembl4.xml and sprot1.xml-sprot40.xml.

the matched pairs for protein data selected by *SLAX* sometimes may not denote exactly the same proteins, it is still helpful and valuable for people to obtain the group of proteins belonging to the same species, organisms, and so on.

5.4 Discussion and Comparison

According to the results of the experiments, we outline the following discussions and comparisons to summarize the characteristics and the differences of *SLAX* and *LAX* when they handle different types of large XML documents.

- From Table 3 and **Fig. 15**, we can observe that *SLAX* is slightly faster than *LAX*. Besides, the execution time for joining a pair of documents for both *SLAX* and *LAX* increases tardily as the number of segmented subtrees increases. With regard to the impact of document sizes, the integration of large documents might be faster than

```
<entry dataset="Swiss-Prot" created="1998-12-15" modified="2005-05-10">
 <accession>O59647</accession>
 <name>MTMW_METWO</name>
 <protein ref="1">
  ...
  <name>M.MwoI</name>
 </protein>
 <gene>
  <name type="primary">mwoIM</name>
 </gene>
 <organism key="2">
  <name type="scientific">Methanobacterium wolfei</name>
  <dbReference type="NCBI Taxonomy" id="145261" key="3"/>
  <lineage>
   <taxon>Archaea</taxon>
   <taxon>Euryarchaeota</taxon>
   ...
  </lineage>
 </organism>
 <reference key="4">
 ...
 </reference>
 ...
 <comment type="function">
 ...
 </comment>
 <dbReference type="EMBL" id="AF051376" key="9">
 ...
 </dbReference>
 ...
</sequence>
```

Fig. 14 The Matched subtree selected by *LAX* from sprot22.xml.

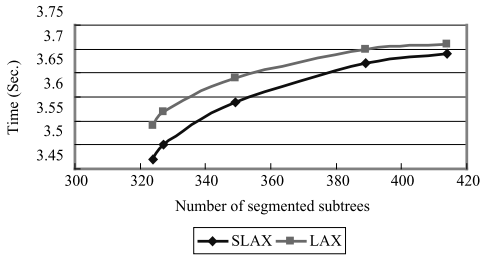


Fig. 15 Execution time for joining a pair of bioinformatics fragments.

that of small ones. For example, the total mean size of SigmodRecord.xml and DBLP fragment files is about 0.8 MB and that of the protein fragment files is about 4.0 MB. However, the mean time for handling the protein data by *SLAX* is 3.63 seconds while that for processing the bibliography data is 3.77 seconds. It takes less time for *SLAX* to handle the protein data, because the number of segmented subtrees of bibliography data is much larger than that of the protein data.

- We can easily find the maximum peak (maximum match value) from Fig. 5 and Fig. 11 for both bibliography and bioinformatics data. While it is of difficulty to discriminate the max peak from Fig. 8 and Fig. 13, because there are many peaks that have very close tree similarity degrees with the maximum one. Therefore, the most proper matched subtree sometimes does not exist in the maximum peak but other peaks having the close value to the maximum one.
- For the bibliography data, the similarity degree of subtrees with smaller number of leaf nodes are easier to be the same because of the smaller n_{1i} in Eq. (1). Therefore, it may happen to get multiple subtrees that have the same similarity degree. On the other hand, the maximum match value for the hit subtrees for bioinformatics data might be small because of the large number of leaf nodes in each subtree.
- *SLAX* can more precisely detect the proper matched subtree for integrating the fragments divided from large XML documents than *LAX*. For the bibliography data, *SLAX* can effectively detect the matched subtree that contains exactly the same information as the base one. The precision of subtree matching using *SLAX* is twice larger than that using *LAX*. For the protein data, *SLAX* is useful and valuable for people to acquire similar proteins belonging to the same species, organisms, and so on, even the two XML documents do not contain exactly the same proteins. Therefore, we consider that *SLAX* is applicable and effective for integrating both large bibliography and bioinformatics data at subtree classes.

6. Conclusions and Future Work

As more and more data are increasingly represented and exchanged by XML on the Internet, a method that can efficiently measure the approximate similarity between XML documents for integrating multiple XML data sources becomes more important. We have proposed *LAX* (**L**eaf-clustering based **A**pproximate **X**ML join algorithm) in previous work, in which the two XML document trees are segmented into subtrees representing independent units, and the output is oriented to the pair of documents that has larger tree similarity degree than the user-defined threshold.

In this paper, we have proposed *SLAX* by using the maximum match value for integrating the fragments divided from large XML documents at subtree classes. We have done experiments to evaluate *SLAX*, comparing with *LAX*, by using both real large bibliography and bioinformatics data. Our experimental results indicate that the precision of subtree matching us-

ing *SLAX* is twice larger than that using *LAX* for bibliography data. And *SLAX* is useful and valuable for people to acquire similar proteins belonging to the same species, organisms, and so on, even if the two XML documents do not contain exactly the same proteins. Therefore, We consider that *SLAX* performs more effectively than *LAX* for integrating both large bibliography and bioinformatics data at subtree classes.

Due to the limitation of the main memory, we plan to do further experiments for evaluating *LAX* and *SLAX* by using different types of large XML data stored RDBs. We are also going to improve the segmentation algorithm for handling more complex XML data. Besides, the semantic heterogeneity is to be taken into account to improve the precision of our algorithms.

Acknowledgments This work was partially supported by the Grant-in-Aid for Scientific Research of MEXT Japan #16016232, by the CREST of JST (Japan Science and Technology Agency), and by the TokyoTech 21COE Program "Framework for Systematization and Application of Large-Scale Knowledge Resources".

References

1) ACM SIGMOD Record in XML. Available at http://www.acm.org/sigmod/record/xml/
2) Arenas, M. and Libkin, L.: A Normal Form for XML Documents, *ACM Transactions on Database Systems*, Vol.29, No.1, pp.195–232 (March 2004).
3) Chawathe, S. and Garacia-Molina, H.: Meaningful Change Detection in Structured Data, *Proc. ACM SIGMOD 1997*, pp.26–37 (1997).
4) Chawathe, S., Tajaraman, A., Garacia-Molina, H. and Widom, J.: Change Detection in Hierarchically Structured Information, *Proc. ACM SIGMOD 1996*, pp.493–504 (1996).
5) Cobena, G., Abiteboul, S. and Marian, A.: Detecting Changes in XML Documents, *Proc. ICDE 2002*, pp.41–52 (2002).
6) Ethier, K. and Abel, S.: Freely Available Structures: XML Document Type Definitions You Can Use Today, *Free Software Magazine*, Issue 6, pp.1–4 (July 2005).
7) Fan, W. and Libkin, L.: On XML Integrity Constraints in the Presence of DTDs, *Proc. PODS'01*, pp.114–125 (2001).
8) Garofalakis, M. and Kumar, A.: Correlating XML data streams using tree-edit distance embeddings, *Proc. PODS'03*, pp.143–154 (2003).
9) Guha, S., Jagadish, H.V., Koudas, N., Srivastava, D. and Yu, T.: Approximate XML Joins, *Proc. ACM SIGMOD 2002*, pp.287–298 (2002).
10) Guha, S., Koudas, N., Srivastava, D. and Yu, T.: Index-Based Approximate XML Joins, *Proc. ICDE 2003*, pp.708–710 (2003).
11) Liang, W. and Yokota, H.: *LAX*: An Efficient Approximate XML Join Based on Clustered Leaf Nodes for XML Data Integration, *Proc. BNCOD 2005*, Springer LNCS 3567, pp.82–97 (July 2005).
12) Marian, A., Abiteboul, S., Cobena, G. and Mignet, L.: Change-Centric Management of Versions in an XML Warehouse, *Proc. 27th VLDB*, pp.581–590 (2001).
13) Nierman, A. and Jagadish, H.V.: Evaluating Structural Similarity in XML Documents, *Proc. WebDB 2002*, pp.61–66 (2002).
14) Selkow, S.: The Tree-to-tree Editing Problem, *Information Processing Letters*, Vol.6, No.6, pp.184–186 (Dec. 1977).
15) Swiss-Prot. http://www.ebi.ac.uk/swissprot/
16) Tai, K.-C.: The Tree-to-Tree Correction Problem, *J. ACM*, Vol.26, No.3, pp.422–433 (1979).
17) TrEMBL. http://www.ebi.ac.uk/trembl/
18) Wang, Y., DeWitt, D.J. and Cai, J.: X-Diff: An Effective Change Detection Algorithm for XML Documents, *Proc. ICDE 2003*, pp.519–530 (March 2003).
19) World Wide Web Consortium (W3C). The Document Object Model (DOM). http://www.w3.org/DOM/
20) XML Version of Swiss-Prot. Avaibable at ftp://www.ebi.ac.uk/pub/databases/uniprot/current_release/knowledgebase/complete/uniprot_sprot.xml.gz
21) XML Version of TrEMBL. Avaibable at ftp://www.ebi.ac.uk/pub/databases/uniprot/current_release/knowledgebase/complete/uniprot_trembl.xml.gz
22) XML Version of DBLP. Available at http://dblp.uni-trier.de/xml/
23) Zhang, K.: Algorithms for the constrained editing distance between ordered labeled trees and related problems, *Pattern Recognition*, Vol.28, No.3, pp.463–474 (1995).
24) Zhang, K. and Shasha, D.: Simple Fast Algorithm for the Editing Distance Between Trees and Related Problems, *SIAM Journal of Computing*, Vol.18, No.6, pp.1245–1262 (Dec. 1989).
25) Zhang, K. and Shasha, D.: Tree Pattern Matching, *Pattern Matching Algorithms*, chapter 11. Oxford University Press (1997).

(Received September 19, 2005)
(Accepted March 13, 2006)

(Editor in Charge: *Hiroshi Ishikawa, Masayoshi Aritsugi, Kaoru Katayama, Yutaka Kidawara, Masashi Tsuchida*)

Wenxin Liang received his B.E. and M.E. degrees in biomedical electronic engineering from Xi'an Jiaotong University, China in 1998 and 2001, respectively. He is currently a Ph.D. student at the Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology, Japan. His current research interests include XML data integration and management, XML storage, indexing, labeling and querying techniques. He is a student member of IPSJ, DBSJ and SIGMOD-J.

Haruo Yokota received his B.E., M.E., and Dr.Eng. degrees from Tokyo Institute of Technology in 1980, 1982, and 1991, respectively. He joined Fujitsu Ltd. in 1982, and was a researcher at the ICOT for the Japanese 5th Generation Computer Project from 1982 to 1986, and at Fujitsu Laboratories Ltd. from 1986 to 1992. From 1992 to 1998, he was an associate professor in Japan Advanced Institute of Science and Technology (JAIST). He is currently a professor at Global Scientific Information and Computing Center, and Department of Computer Science in Graduate School of Information Science and Engineering in Tokyo Institute of Technology. His current research interests include general research area of data engineering, information storage systems and dependable computing. He is a member of DBSJ, IPSJ, IEICE, JSAI, IEEE, IEEE-CS, ACM, ACM-SIGMOD, and ACM-SIGARCH.