Semantic-based Optimal XML Schema Matching:

A Mathematical Programming Approach

Jaewook Kim and Yun Peng
Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County
Baltimore, MD USA
jaewook2@umbc.edu and ypeng@umbc.edu

Nenad Ivezic and Junho Shin
Manufacturing Systems Integration Division
National Institute of Standards and Technology
Gaithersburg, MD USA
nivezic@nist.gov and junho.shin@nist.gov

***Abstract*—We propose a novel solution for semantic-based XML schema matching, taking a mathematical programming approach. This method identifies the globally optimal solution for the problem of matching two XML schemas by reducing the tree-to-tree matching problem to simpler problems of path-to-path, node-to-node, and word-to-word matching. We formulate these matching problems as maximum weighted bipartite graph matching problems with different constraints, which are solved by different mathematical programming techniques, including integer programming and dynamic programming. Solutions to simpler problems provide weights for the next stage until the optimal tree-to-tree matching solution is obtained. The effectiveness of this approach has been verified and demonstrated by computer experiments.**

Keywords-E-business; XML schema matching; optimization; maximum weighted bipartite graph; semantic similarity; integer programming; dynamic programming

I. INTRODUCTION

XML [1] and XML schemas [2] have been widely used in the e-Business transactions among enterprises that exchange business documents with their partners (e.g., suppliers and customers in the supply chain) [3,4,5]. Many enterprises and organizations have defined their own XML schemas to describe the structure and content of their business documents (i.e., XML instances) to be used in the transactions. Many organizations have also published standard XML schemas to be shared in the transactions within specific industry domains [6,7,8] (e.g., e-manufacturing, e-government, and e-health industries).

The popularity of XML leads to an integration problem as different enterprises or organizations often choose different XML representations for the same or similar concepts [4,5]. One of the most critical steps to achieving the seamless exchange of information between heterogeneous e-Business systems is schema matching, which is known to be costly and error-prone [9,10]. Schema matching takes as input two schemas, each consisting of a set of discrete components (elements or attributes), and determines as output the relationships between these components [11].

XML schemas or instances are typically reviewed as labeled trees (i.e., rooted acyclic graphs) where each node represents a component named by a label of English word or concatenation of words or their abbreviations. In this paper we focus on one type of schema mapping, namely, matching between all atomic components (i.e., the leaf nodes) between two schemas or instances based on their semantics (meaning of nodes). Also, we only consider pair-wise matchings of 1:1 cardinality (e.g., any atomic component in the source schema can match no more than one atomic component in the target schema). Matching between those atomic components help identify how a certain value of one XML instance can be transformed to certain value of the other for successful exchange of information.

We propose new innovative techniques to address two challenging problems in this type of schema matching. First, due to synonyms (different words meaning the same thing) and multi-senses (one word having different meanings in different context) found in natural languages, the meaning of an atomic component cannot be determined solely by the words in its label. The *semantic ambiguity* can be reduced by contextual information such as the labels of its neighboring nodes. In this paper, we concentrate on one type of context for an atomic component: the nodes along the path from the root to the leaf of the schema tree.

Second, it is difficult to correctly identify the *best set* of matching pairs for all atomic components between two trees. This is because one leaf in one tree may match more than one leaf in the other tree (with different semantic similarities) and locally identified best matching pairs do not necessarily form the globally optimal set. We propose to use mathematical programming techniques to solve this combinatorial optimization problem. To further reduce the computational complexity, we propose to decompose the global optimization into simpler matching problems such as path-to-path, node-to-node, and word-to-word matching. We formulate the sequence of matching problems as maximum weighted bipartite matching problems with different sets of constraints and solve them by different mathematical programming techniques, including integer programming [12] and dynamic programming [13]. Solutions to simpler problems provide weights for the next stage until the optimal tree-to-tree matching is obtained.

The remainder of the paper is organized as follows. Section II provides a brief survey of the related works. The detailed algorithm of the proposed approach is described in Sections III. Section IV reports the experiments and results. Section V concludes with the directions for future research.

II. Related Works

Many schema matching methods have been proposed [10,11]. Typically, these methods first attempt to identify semantic relationships between the elements of two schemas. Based on the granularity of the matching, these schema matching techniques can be separated into two classes: element-level and structure-level [11]. The element-level approaches determine the matching elements in the target schema for each element of the source schema. The structure-level approaches refer to matching combinations of elements that appear together in a structure.

For the element-level matching, string-based similarity metric is the most fundamental technique to analyze the linguistic context of names and name descriptions of schema elements. There is a variety of string-based similarity metrics, including Hamming distance [14], Jaccard similarity coefficient [15], Cosine coefficient [16], and n-gram [17]. The string-based metrics can be enhanced using natural language preprocessing techniques for the input string, such as tokenization, lemmatization, and elimination [11].

To further enhance the string-based metrics, corpus resources can be utilized for more accurate and less ambiguous results. One of the important resources is the lexical taxonomy among the words (e.g., parents, children, ancestor, and descendant relationships). Common knowledge corpora, such as WordNet [18] and domain-specific corpora can be used to determine the meaning of the words [19,20].

A corpus also provides statistical information related to the importance of words. The difference in importance of individual entities and their relationships affects the semantic similarity measurement. The information content (IC)-based metric was proposed to utilize this statistical information [21,22,23]. This approach measures the similarity between two entities (e.g., two words, two objects, or two structures), *A* and *B* based on how much information is needed to describe *common*(*A*, *B*), the commonality between them (e.g., the features or hypernyms the two words share). Applying this approach to tree-like IS-A taxonomies [22], one can measure the similarity between *A* and *B* as

$$Sim_{IC}(A,B) = \frac{2 \cdot \log P(C)}{\log P(A) + \log P(B)}, \tag{1}$$

where C is the most specific subsumer of A and B with the smallest prior probability and the probabilities can be obtained according to the frequencies in a corpus .

For structure-level matching, a variety of graph-based techniques have been proposed [10,11]. Typically, a graph-based metric quantifies the commonality between components by taking into account the lexical and structural similarities of sub-components (e.g., ancestors and descendents including leaf components). Because most schemas can be viewed as labeled trees, many matching algorithms have been developed based on either top-down or bottom-up traversal techniques [10].

As an example of the top-down approach, TransScm [24] provides a schema-based matching for data translation and conversion based on the syntactic analysis of the structures. Tess [25] is another example of a top-down algorithm, which deals with schema evolution. Tess takes definitions of the old and new types and identifies pairs of types as matching candidates. It then recursively tries to match their substructure in a top-down fashion.

Alternatively, the bottom-up approach compares all combinations of the elements and finds matches at a given level even if the intermediate and higher level structures differ considerably. Similarity flooding (SF) [26] provides bottom-up matching based on similarity propagation. Another effective bottom-up method, called S-Match [27], decomposes the tree matching problem into a set of node matching problems. Each node matching problem is translated into a propositional formula, thus transforming the matching problem to a propositional unsatisfiability problem, which can then be efficiently resolved using state of the art propositional satisfiability deciders.

Cupid [28] provides a composite matching approach combining element- and structure-level matching approaches. It computes the similarity with domain-specific thesauri as the linguistic resources and traverses the tree in a combined bottom-up and top-down manner.

Top-down approaches are less expensive but can be misled if the top-level schema structures are very different [10]. On the other hand, bottom-up approaches take a more comprehensive view [11]. However, the ways the existing bottom-up methods identify the optimal matching are typically ad hock, not based on a theoretically well founded manner.

III. Algorithm

We propose a novel schema-based matching algorithm to solve the combinatorial matching optimization problem of matching atomic components between two schemas. Our approach synthesizes the global optimal set of pair-wise matchings between atomic components in a principled manner by mathematical programming.

A. *Algorithm overview*

The matching algorithm takes as input two schemas and identifies the set of matching pairs of all atomic components of the input schemas with the highest semantic similarity among all possible sets of pair-wise matchings. Fig. 1 illustrates the matching process of our approach for two input schemas, *S1* and *S2*.

The algorithm breaks the complex combinatorial optimization problem into four matching stages: tree-to-tree, path-to-path, node-to-node, and word-to-word matchings. As can be seen in Fig. 2, each stage works on a bipartite graph, consisting of two sets of vertices and a weight matrix with the objective of finding the 1:1 matching between vertices in one set to the other with the highest overall weight. Therefore, we formulate these sub-problems as *maximum weighted bipartite graph matching* problems [29].

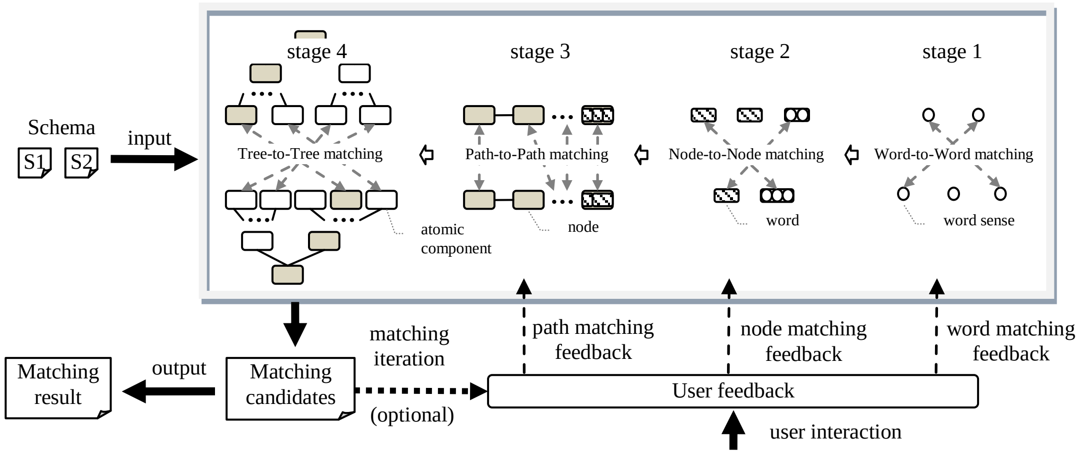


Figure 1. Matching algorithm overview.

Except for the word-to-word matching stage, the weight matrix for each stage is a similarity matrix calculated by the previous stage. For example, the similarity matrix for tree-to-tree matching stage is provided by path-to-path matching stage. The word-to-word matching stage uses WordNet to compute the semantic similarity between two words by identifying the optimal matching pairs for their respective senses.

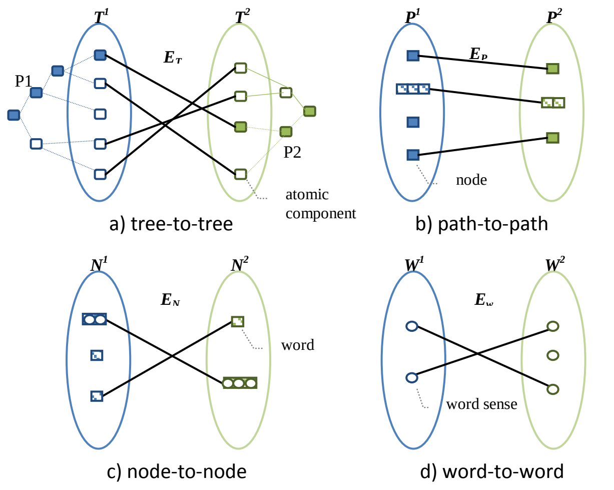


Figure 2. Weighted bipartite graph modeling for different types of components in two labeled trees.

Except for the path-to-path matching stage, optimal matching at each stage can be obtained according to the general *Maximum-weighted Bipartite Matching* algorithm (*MBM*) [30]. The path-to-path matching requires an additional *ordering criterion* [31] that path *P1* includes most of the nodes of path *P2* in the correct order, and is called *Ordered Maximum-weighted Bipartite Matching* (*OMBM*) problem. Algorithms to solve the *MBM* and the *OMBM* problems are described in the following sections.

B. Maximum-weighted bipartite matching algorithm

Tree-to-tree, node-to-node, and word-to-word matching stages can be formulated as the general *weighted bipartite graph matching* problems. Let *G* be a weighted bipartite graph with two sets of vertices, $U=\{u_1,u_2,\cdots,u_m\}$ and $V=\{v_1,v_2,\cdots,v_n\}$, and the edge set *E*. Edge e_{ij} in the graph connects the vertices u_i and v_j whose weight w_{ij} is given in the weight matrix *W*. Vertices of the same set are not connected.

A matching *M* of graph *G* is a subset of *E* such that no two edges in *M* share a common vertex. In other words, M consists of a set of pair-wise matchings of 1:1 cardinality. The *maximum-weighted bipartite matching* is a matching whose sum of the weights of the edges is the highest among all possible sets of pair-wise matchings. The optimal matching *M* can be found by integer programming defined below:

$$\text{Maximize: } \sum_{e_{ij}\in E} w_{ij}x_{ij} \qquad (2)$$

subject to:

$$\sum_{i=1}^{m} x_{ij}=1, \forall j=1,\cdots,n, \ \sum_{j=1}^{n} x_{ij}=1, \forall i=1,\cdots,m,$$

$x_{ij}\in\{0,1\}$, where x_{ij} is 1 if $e_{ij}\in M$ and 0 otherwise.

Because integer programming is typically NP-hard (i.e., harder than a nondeterministic polynomial-time problem and for worst case with running time exponential to the problem size) [32], we approximate it by a simple greedy algorithm as follows:

```
algorithm MBM-greedy (U,V,W)
   sort W;
   while (|U|>0 and |V|>0)
       Choose vertices u and v connected with an edge e that has the
       highest weight w in the weight matrix W;
       if edges in M share neither u nor v
           then M := M + {e}, U := U - {u}, V := V - {v};
       W [u,v] := 0;
       wsum := wsum + w;
   wavg := 2 * wavg / (m + n);
   return {M, wavg};
```

Figure 3. Greedy algorithm for maximum weighted bipartite matching.

The greedy algorithm simply sorts the weight matrix *W* in ascending order and at each iteration chooses an edge with the highest weight. The initial weight matrix *W* is calculated by the previous matching stage. The chosen edge would be the matching candidate if it shares no vertex with edges in *M*. This process is repeated until there is no vertex to be matched in either *U* or *V*. The algorithm returns the optimal matching set *M* and the average weight of all edges in *M* as the measure of similarity between *U* and *V*. In this greedy algorithm, the most expensive step is the sorting of the weight matrix *W* of size $m \times n$. We use a quicksort algorithm [33] that takes *O*(*k*log(*k*)) to sort *k* items. Thus, the complexity of this greedy algorithm is *O*(*mn* log(*mn*)).

C. Ordered maximum-weighted bipartite matching algorithm

Some have suggested using the longest common sequence (LCS) to address the ordering criterion in the path-to-path matching [30,34,35] . However, none of the suggestions utilizes the semantic similarities of the nodes on the two paths. To consider semantic similarities of the nodes, we have developed the *ordered maximum-weighted bipartite matching* algorithm based on dynamic programming.

Let *G* be a weighted bipartite graph with two *ordered* sets of vertices $U = \{u_1, u_2, \cdots, u_m\}$ and $V = \{v_1, v_2, \cdots, v_n\}$, and the edge set *E*. The core algorithm, *OMBM* (*U*, *V*), finds the optimal matching *M* between *U* and *V* by recursively partitioning it into smaller sub-problems until the solution becomes trivial.

For a sequence $S=s_1s_2...s_d$, a sequence shortened from the end is denoted $S_k=s_1s_2...s_k$, where $k \le d$. We call S_k the *prefix* of *S*. The *prefixes* of *U* are $U_1, U_2, ..., U_m$, and the *prefixes* of *V* are $V_1, V_2, ... V_n$. Let *OMBM* (U_i, V_j) be the function that finds the optimal matching of *prefixes* U_i and V_j. This problem can be reduced to three alternative simpler sub-problems with shortened *prefixes* and returns the one with maximum sum of weights:

1) u_i and v_j match each other. Then the optimal matching for U_i and V_j can be formed by attaching edge e_{ij} to the optimal matching of two shortend sequences U_{i-1} and V_{j-1}, denoted. $\left(OMBM\left(U_{i-1}, V_{j-1}\right), e_{ij}\right)$.

2) u_i and v_j do not match each other. Then, either of them can be removed to shorten one of the matching sequences and *OMBM* (U_i, V_j) is reduced to either *OMBM* (U_{i-1}, V_j) or *OMBM* (U_i, V_{j-1}).

Thus *OMBM* (U_i, V_j) can be computed by the following recursive function:

$$OMBM\left(U_i, V_j\right) = \begin{cases} \varnothing & \text{if } i = 0 \text{ or } j = 0 \\ max\begin{pmatrix} OMBM\left(U_{i-1}, V_j\right), \\ OMBM\left(U_i, V_{j-1}\right), \\ \left(OMBM\left(U_{i-1}, V_{j-1}\right), e_{ij}\right) \end{pmatrix} & \text{otherwise} \end{cases}, \quad (3)$$

where the function *max*() returns the optimal matching among the three matchings from the sub-problems; it returns empty if either U_i or V_j is reduced to nill (*i* = 0 or *j* = 0).

The optimal matching *M* of two sets of order vertices *U* and *V*, |*U*| = *n*, |*V*| = *m*, is then computed as:

$$M = OMBM\left(U_m, V_n\right) \cdot \quad (4)$$

The similarity score of *M*, denoted Sim_M, is the average weights of all edges in *M*:

$$Sim_M = \sum_{e_{ij} \in M} w_{ij} \cdot \frac{2}{m+n} \cdot \quad (5)$$

To efficiently execute the algorithm, we use a bottom-up approach [13]. The algorithm is as follows:

```
algorithm OMBM-arrays (U,V,W)
  for i from 1 to m
    for j from 1 to n
      A [i,j] := maximum of A[i-1,j], A[i,j-1], and A[i-1,j-1]+W [i,j];
  wavg := 2*A[m,n] / (m + n);
  return {A, wavg};
```

Figure 4. Bottom-up dynamic programming algorithm for ordered maximum weighted bipartite matching.

This algorithm starts from the simplest matching between U_1 and V_1 and continues to more complex matching problems. The calculated similarity scores for the optimal matchings (average weights by Eq. (5)) are stored in table *A*[*i*,*j*] in Fig. 4 for future use to avoid repeated calculations of smaller problems. The weights of *W* are calculated by the previous matching stage (i.e., node-to-node matching stage). The complexity is only *O*(*mn*), i.e., linear to the size of table *A*.

The following algorithm deals with a simple matter of finding the matching between components *U* and *V* that is identified by bottom-up dynamic programming algorithm of Fig. 5.

```
algorithm OMBM (U,V,W)
  OMBM-arrays (U,V,W);
  i := m, j :=n;
  while i > 0 and j > 0
    if A[i,j] equal to A[i-1,j] then i--;
      elseif A[i,j] equal to A[i,j-1] then j--;
        else M := M+{e_{i,j}}, i--, j--;
  return {M, wavg};
```

Figure 5. Dynamic programming algorithm for ordered maximum weighted bipartite matching.

Algorithm *OMBM* is further enhanced by considering the differences in importance for the individual components measured by their IC values. We collect the frequency of each component by their occurrences in the schemas and compute the IC by Eq. (1). Fig. 6 shows the modified algorithm.

Algorithm *OMBM-IC* gives more weights to higher level components because lower level components are typically generic and common entities that appear widely as the descendants of many elements and thus less discriminatory than the higher level components. In addition, it also considers the differences in importance of components at the same level. The complexity of this algorithm is still *O*(*mn*).

```
algorithm OMBM-IC (U,V,W)
  for i from 1 to m
    ic-sum := ic-sum + ic($u_i$);
  for j from 1 to n
    ic-sum := ic-sum + ic($v_j$);
  for i from 1 to m
    for j from 1 to n
      ic-weight = W [$u_i$, $v_j$]*(ic($u_i$)+ic($v_j$));
      A[i,j] := maximum of A[i-1,j], A[i,j-1], and A[i-1,j-1]+ ic-weight;
  wavg := A[m,n]/ ic-sum;
  return {A, wavg};
```

Figure 6. Algorithm enhanced by information contents.

D. Overall schema matching algorithm

Fig. 7 below gives the algorithm for overall schema matching and how each stage obtains the weight matrix by calling the optimization algorithm for the previous stage.

```
algorithm T2T-matching (T1, T2)
  for i from 1 to |T1|
    for j from 1 to |T2|
      t2t-smatix[i,j] := P2P-matching (T1's $i_{th}$ path, T2's $j_{th}$ path);
  return MBM-greedy (T1's atomics, T2's atomics, t2t- smatix);

algorithm P2P-matching (P1, P2)
  for i from 1 to |P1|
    for j from 1 to |P2|
      p2p-smatix[i,j] := N2N-matching (P1's $i_{th}$ node, P2's $j_{th}$ node);
  return OMWM-IC (P1's nodes, P2's nodes, p2p-smatix);

algorithm N2N-matching (N1, N2)
  for i from 1 to |N1|
    for j from 1 to |N2|
      n2n-smatix[i,j] := W2W-matching (N1's $i_{th}$ word, N2's $j_{th}$ word);
  return MWM-greedy (N1's words, N2's words, n2n-smatix);

algorithm W2W-matching (W1, W2)
  if wordnet definitions for W1 and W2 exists then
    for i from 1 to |W1|
      for j from 1 to |W2|
        w2w-smatix[i,j] := word-sense-sim (W1's $i_{th}$ sense, W2's $j_{th}$ sense);
    return MWM-greedy (W1's senses, W2's senses, w2w-smatix);
  else
    return word-desc-sim (W1, W2);
```

Figure 7. Overall schema matching algorithm.

The algorithm views matching two schema trees as matching two sets of atomic components with their respective path-contexts. Each path consists of a sequence of nodes along the path from the root to the leaf of the schema tree. Each node represents a component named by a label of English word or concatenation of words or their abbreviations. To compute semantics similarities between words, we analyze optimal pair-wised matchings between multiple senses of two words.

The word-to-word matching algorithm uses two semantic similarity measure functions: *word-sense-sim*() based on WordNet taxonomy and *word-desc-sim*() based on textual description. In WordNet, nouns are organized into taxonomies in which each node has a set of synonyms (a synset), each of which representing a single sense [18]. If a word has multiple senses, it will appear in multiple synsets at various locations in the taxonomy. To compute the semantic similarity between two words (two sets of senses), we use the *MBM-greedy*() algorithm with the input of two set of synsets for words *W1* and *W2*, respectively, and a similarity matrix of all possible pairs between senses in *W1* and *W2* is calculated by Eq. (1).

If a word does not exist in WordNet, we extract the textual description of a given word from the internet and then use string similarity measures, such as the cosine similarity [16], to calculate the similarity between the two textual descriptions of the two words.

IV. Experiments and Results

We have implemented a prototype system of our approach based on Java and JWNL [36] for experimental validation. In the experiments, we used five real world XML schemas for purchase orders (i.e., CIDX, Apertum, Excel, Norris, and Paragon) from [37,38]. Table I summarizes the characteristics of those XML schemas.

TABLE I. Characteristics of PO XML Schemas

Schemas	CIDX	Apertum	Excel	Norris	Paragon
max depth	4	5	4	4	6
# nodes	40	145	55	65	80
# leaf nodes	33	116	42	54	68

In the experiment, as suggested in [36], we compute the tree-to-tree similarity of the ten pairs of the five XML schemas, then for each schema, a matching to any of the other four is accepted if the similarity score is above a fixed threshold 0.6. To evaluate the quality of our match result, we used several performance metrics including Precision, Recall, F-measure, and Overall [39,40], against the results from manual matching [38]. These measures are then compared with the performances of other approaches for the same setting [28,38,41]. Note that the Overall measure, proposed by [38] to estimate the post-match efforts, varies in [-1, 1] and other three vary in [0,1].

Precision, Recall, F-measure, and Overall for our results are 0.85, 0.85, 0.85, and 0.69. To increase the precision, we used a relative threshold which is chosen as the similarity of the matching with the largest gap to the next best matching, among matching candidates with similarities ranging from 0.5 to 0.6. Fig. 8 shows the performance analysis of the matching result that our solution produced.

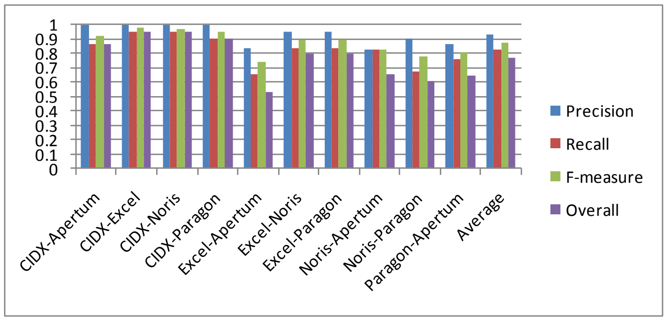
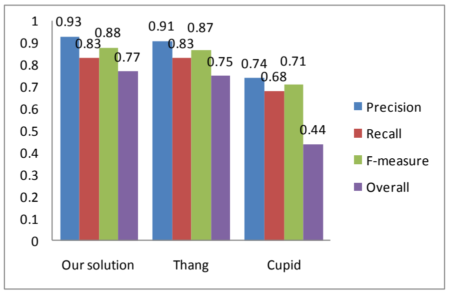


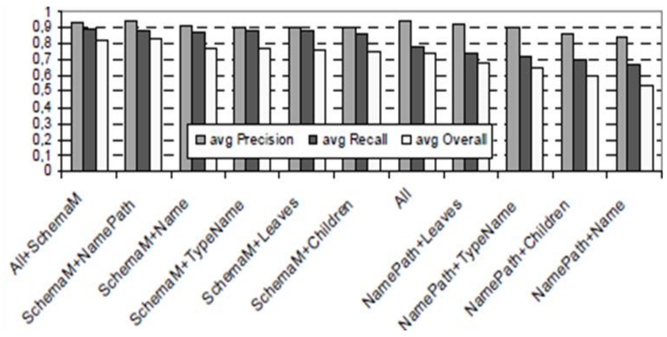
Figure 8. Performance analysis.

The experiment results show that our matching performances of Precision, Recall, F-measure, and Overall are 0.93, 0.83, 0.88, and 0.77, respectively. Comparing to the previous results that use a fixed threshold, the Recall is slightly decreased while the Precision is significantly increased. The relative threshold also helps to increase F-measure and Overall.

For comparison purpose, the average scores of performance metrics by some other methods are given in Fig. 9.



a) Thang



b) COMA: matcher combinations

Figure 9. Performance analysis.

The first comparison, as illustrated in Fig. 9a, is with Thang [41] who proposed an XML schema matching solution that combines linguistic, data type, and path-context similarity measures. He also implemented the Cupid [28] algorithm for comparison purpose. We compared our result to both algorithms'. In general, all performance metrics of our approach are slightly better than Thang's and significantly better than Cupid's.

The second comparison is with COMA [38] which used various ways for combining different matchers. Because COMA only provides performance graphs without the specific score as shown in Fig. 9b, it is difficult to exactly compare the performances with our result. However, Fig. 9b shows that our result is, in general, at least equal to or slightly better than COMA's results even if some of their matchers used the manual matching called *SchemaM* [38].

V. CONCLUSIONS

In this paper, we have described a solution to identify semantic-based optimal XML schema matching using mathematical programming. This solution identifies the optimal matching between two XML schemas on the assumption that the tree-to-tree matching problem can be globally optimized by reducing it to simpler problems, such as path-to-path, node-to-node, and word-to-word matching. We have implemented a prototype system for our solution and conducted the experiments with actual industry XML schemas. We compared our result to some other XML schema matching approaches. The results were encouraging. The average matching performances of Precision, Recall, F-measure, and Overall were 0.93, 0.83, 0.88, and 0.77, which are better than or at least equal to other approaches'.

Although our approach primarily targets the XML schema matching problem, the solution can be applied to other matching problems (e.g., XML instance matching) if the instances can be represented as labeled trees. Our solution is limited to the assumptions that only 1:1 matching cardinality is considered and that schema designers correctly use the English terminologies when labeling the elements/attributes in the schemas. These limitations call for further research. Other directions of research include methods to improve the performance by utilizing domain specific terminology and taxonomy, ontologies with formally defined concept semantics, and user feedback.

ACKNOWLEDGMENT

This work was supported in part by NIST award 70NANB9H9145.

REFERENCES

[1] W3.org "Extensible Markup Language (XML) 1.1 specification," available at: http://www.w3.org/TR/xml11/

[2] W3.org, "XML schema 1.1 specification," available at: http://www.w3.org/TR/xmlschema11-1/

[3] R. Kalakota, and M. Robinson, "E-business: roadmap for Success," Addison-Wesley, Reading, MA, 1999.

[4] J.M. Nurmilaakso and P. Kotinurmi, "A review of XML-based supply-chain integration," Production Planning and Control, vol. 15, no. 6, Sep. 2004, pp. 608–621, doi: 10.1080/09537280412331283937.

[5] R. Skinstad, "Business Process Integration through XML", available at: http://www.infoloom.com/gcaconfs/WEB/paris2000/S10-03.HTM

[6] S.Y. Shim, V.S. Pendyala, M. Sundaram, and J.Z. Gao, "Business-to-business e-commerce frameworks," IEEE Computer, vol. 33, no. 10, Oct. 2000, pp. 40-47, doi: 10.1109/2.876291.

[7] B. Medjahed, B. Benatallah, A. Bouguettaya, A. Ngu, and A. Elmagarmid, "Business-to-business interactions: issues and enabling technologies," VLDB Journal, vol. 12, no. 1, May 2003, pp. 59–85, doi: 10.1007/s00778-003-0087-z.

[8] C. Bussler, "B2B protocol standards and their role in semantic B2B integration engines," Bull Tech Comm Data Eng, vol. 24, no. 1, 2001, pp. 3–11.

[9] A. Gal, "Why is Schema Matching Tough and What Can We Do About It?," ACM Sigmod Record, vol. 35, no. 4, 2006, pp. 2-5. doi: 10.1145/1228268.1228269.

[10] E. Rahm and P.A. Bernstein, "A survey of approaches to automatic schema matching," VLDB Journal, vol. 10, no. 4, 2001 , pp. 334-350. doi: 10.1007/s007780100057.

[11] P. Shvaiko, and J. Euzenat, "A survey of schema-based matching approaches," Journal on Data Semantics IV, LNCS 3730, 2005, pp. 146-171. doi: 10.1007/11603412_5.

[12] K.H. Elster, "Modern Mathematical Methods of Optimization," Vch Pub. 1993, ISBN 3-05-501452-9.

[13] R.E. Bellman, "Dynamic Programming," Princeton University Press, Princeton, NJ, 1957, Republished 2003: Dover, ISBN 0486428095.

[14] R.W. Hamming, “Error detecting and error correcting codes,” Bell System Technical Journal, vol. 29, no. 2, 1950, pp. 147–160, MR0035935.

[15] C.J. Van Rijsbergen, “Information retrieval,” 2nd ed., London: Butterworths, 1979.

[16] H.A. Sneath, “Then application of computers to taxonomy,” Journal of General Microbiology, vol. 17, no. 1, 1957, pp. 201-226, doi: 10.1099/00221287-17-1-201.

[17] E. Ukkonen, “Approximate string matching with q-grams and maximal matches,” Theoretical Computer Science, vol. 92, no. 1, 1992, pp. 191-211.

[18] G.A. Miller, “WORDNET: a lexical database for English,” Communications of ACM, vol. 38, no. 11, 1995, pp. 39-41. doi: 10.1145/219717.219748.

[19] P. Qin, Z. Lu, Y. Yan, and F. Wu, “A New Measure of Word Semantic Similarity Based on WordNet Hierarchy and DAG Theory,” Proc. of International Conference on Web Information Systems and Mining, 2009, pp. 181-185, doi: 10.1109/WISM.2009.44.

[20] D. Yang, and D.M.W. Powers, “Measuring semantic similarity in the taxonomy of WordNet,” Proc. of the 28th Australasian Computer Science Conference, 2005, pp. 315-322, doi: 10.1.1.87.678.

[21] P. Resnik, “Using information content to evaluate semantic similarity in a taxonomy,” Proc. of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 448-453, doi: 10.1.1.55.5277.

[22] D. Lin, “An Information-theoretic definition of similarity,” Proc. of the 15th International Conference on Machine Learning, 1998, pp. 296-304. doi: 10.1.1.55.1832.

[23] T.M. Cover and J.A. Thomas, “Elements of information theory,” Wiley series in telecommunications. Wiley, New York, 1991.

[24] T. Milo and S. Zohar, “Using schema matching to simplify heterogeneous data translation,” Proc. of the 24th International Conference on Very Large Data Bases, 1998, pp. 122-133. doi: 10.1.1.30.2620.

[25] B.S. Lerner, “A model for compound type changes encountered in schema evolution,” ACM Transactions on Database Systems, vol. 25, no. 1, 2000, pp. 83-127, doi: 10.1.1.105.1542.

[26] S. Melnik, H. Garcia-Molina, and E. Rahm, “Similarity flooding - a versatile graph matching algorithm,” Proc. of 18th International Conference of Data Engineering, 2002, pp. 117-128. doi: 10.1.1.61.4266.

[27] F. Giunchiglia, P. Shvaiko, and M. Yatskevich, “S-Match: an algorithm and an implementation of semantic matching,” Proc. of the European Semantic Web Symposium (ESWS), 2004, pp. 61–75, doi: 10.1007/978-3-540-25956-5_5.

[28] J. Madhavan, P.A. Bernstein, and E. Rahm, “Generic schema matching with Cupid,” Proc. of the 27th International Conference on Very Large Data Bases, 2001, pp. 49-58. doi: 10.1.1.17.4650.

[29] A.L. Dulmage and N.S. Mendelsohn, “Coverings of bipartite graphs,” Canadian Journal of Mathematics, vol. 10, 1958, pp. 517–534.

[30] W.B. Douglas, “Introduction to Graph Theory,” 2nd ed., Prentice Hall, Chapter 3, 1999, ISBN 0-13-014400-2.

[31] D. Carmel, Y. Maarek, Y. Mass, N. Efraty, and G. Landau, “An Extension of the Vector Space Model for Querying XML documents via XML fragments,” in ACM SIGIR 2002 Workshop on XML and Information Retrieval, Tampere, Finland, August 2002.

[32] C. H. Papadimitriou, “On the complexity of integer programming,” J. ACM, vol. 28, 1981, pp. 765–768.

[33] C.A.R. Hoare, ”Quicksort,” Computer Journal, vol. 5. no. 1, 1962, pp. 10-15.

[34] L.L. Mong, Y.H. Liang, H. Wynne, Y. Xia, “XClust: Clustering XML Schemas for Effective Integration,” Proc. in 11th ACM International Conference on Information and Knowledge Management (CIKM), McLean, Virginia, November 2002, doi: 10.1145/584792.584841.

[35] A. Boukottaya and C. Vanoirbeek, “Schema Matching for Transforming Structured Documents,” In DocEng, 2005, pp. 2-4, doi: 10.1145/1096601.1096629.

[36] Java WordNet Library (JWNL), available at: http://sourceforge.net/apps/mediawiki/jwordnet

[37] BizTalk Server available at: http://www.microsoft.com/biztalk/

[38] D. Aum¨uller, H.H. Do, S. Massmann, and E. Rahm, “Schema and ontology matching with COMA++,” Proc. of the International Conference on Management of Data (SIGMOD), Software Demonstration, 2005

[39] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel, “Performance measures for information extraction,” Proc. of DARPA Broadcast News Workshop, Herndon, VA, February 1999

[40] C.J. van Rijsbergen, “Information Retrieval,” 2nd ed., Butterworth, 1979.

[41] H.O. Thang, V.S. Nam, “XML Schema Automatic Matching Solution,” International Journal of Computer Systems Science and Engineering, vol. 4, no. 1, 2008, pp. 68-74.