Schema Matching Using Interattribute Dependencies

Jaewoo Kang and Jeffrey F. Naughton

Abstract—Schema matching is one of the key challenges in information integration. It is a labor-intensive and time-consuming process. To alleviate the problem, many automated solutions have been proposed. Most of the existing solutions mainly rely upon textual similarity of the data to be matched. However, there exist instances of the schema-matching problem for which they do not even apply. Such problem instances typically arise when the column names in the schemas and the data in the columns are opaque or very difficult to interpret. In our previous work, we proposed a two-step technique to address this problem. In the first step, we measure the dependencies between attributes within tables using an information-theoretic measure and construct a dependency graph for each table capturing the dependencies among attributes. In the second step, we find matching node pairs across the dependency graphs by running a graph-matching algorithm. In our previous work, we experimentally validated the accuracy of the approach. One remaining challenge is the computational complexity of the graph-matching problem in the second step. The problem instance we are facing is the weighted graph-matching problem to which no efficient solution has yet been found. In this paper, we extend the previous work by improving the second phase of the algorithm incorporating efficient approximation algorithms into the framework.

Index Terms—Schema matching, attribute dependency, graph matching.

1 Introduction

The schema-matching problem at the most basic level refers to the problem of mapping schema elements (for example, columns in a relational database schema) in one information repository to corresponding elements in a second repository. While schema matching has always been a problematic and interesting aspect of information integration, the problem is exacerbated as the number of information sources to be integrated, and hence, the number of integration problems that must be solved, grows. Such schema-matching problems arise both in "classical" scenarios such as company mergers and in "new" scenarios such as the integration of diverse sets of queriable information sources over the web.

Purely manual solutions to the schema-matching problem are too labor intensive to be scalable; as a result, there has been a great deal of research into automated techniques that can speed this process by either automatically discovering good mappings, or by proposing likely matches that are then verified by some human expert. In this paper, we present such an automated technique that is designed to be of assistance in the particularly difficult cases in which the column names and data values are "opaque," and/or cases in which the column names are opaque and the data values in multiple columns are drawn from the same domain. Our approach works by computing the "mutual information" between pairs of columns within each schema, and then using this statistical characterization of pairs of columns in one schema to propose matching pairs of columns in the other schema.

To clarify our aims and provide some context, consider a classical schema mapping problem where two employee tables are integrated. How should we determine which attributes in one table should be mapped to which attributes in the other table? First, one logical approach is to compare attribute names across the tables. Some of the attribute names will be clear candidates for matching, due to common names or common parts of names. Using the classification given in [47], such an approach is an example of *schema-based matching* [15], [46]. However, for many columns, schema-based matching will not be effective because different institutions may use different names for semantically identical attributes, or use similar names for unrelated attributes.

When schema-based matching fails, the next logical approach is to look at the data values stored in the schemas. Again referring to the classification from [47], this approach is called *instance-based matching* [18], [32], [39]. Instance-based matching also will work in many cases. For example, if we are deciding whether to match $Dept$ in one schema to either $DeptName$ or $DeptID$ in the other, by looking at the column instances, one may easily find the mapping because $DeptName$ and $DeptID$ are likely to be drawn from different domains. Unfortunately, however, instance-based matching is also not always successful.

When instance-based mapping fails, it is often because of its inability to distinguish different columns over the same data domain and, similarly, its inability to find matching columns using values drawn from different domains. For example, $EmployeeID$ and $CustomerID$ columns in a table are unlikely to be distinguished if both the columns use similar IDs. By the same token, if the two tables use different types of IDs for the same column, the traditional

- *J. Kang is with the College of Information and Communication, Korea University, Anam-Dong, Seongbuk-Gu, Seoul 136-705, Korea. E-mail: kangj@korea.ac.kr.*
- *J.F. Naughton is with the Department of Computer Sciences, University of Wisconsin-Madison, 1210 West Dayton Street, Madison, WI 53706-1685. E-mail: naughton@cs.wisc.edu.*

Manuscript received 12 Oct. 2007; revised 30 Apr. 2008; accepted 12 May 2008; published online 14 May 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-10-0498. Digital Object Identifier no. 10.1109/TKDE.2008.100.

TABLE 1
Two Tables from Car Part Databases

Model	Color	Tire
XLE	White	P2R6
XG2.5	Silver	XR5
LE	Red	GM6

A	B	C
GL3.5	b1	c1
XGL	b2	c2
XE	b3	c3

instance-based approach will fail to identify the correspondence between the two columns.

The technique we propose in this paper is also an instance-based technique. However, it applies to the cases where previously proposed techniques do not apply because 1) it does not rely on any interpretation of data values and 2) it considers correlations among the columns in each table. We emphasize that our claim is not that our technique dominates previously proposed techniques (it does not); rather, since it applies where previous techniques do not apply, it is a useful addition to a suite of automated schema mapping tools.

To gain insight into our approach, consider the example tables in Table 1. Suppose these tables are from two automobile plants in different companies. Imagine that the column names of the second table and data instances in columns B and C are some incomprehensible values to the schema-matching tools. Conventional instance-based matchers may find correspondence between the columns $Model$ and A due to their syntactic similarity. However, no further matches are likely to be found because the two columns B and C cannot be interpreted, and they share exactly same statistical characteristics; that is, they have the same number of unique values, similar distributions, and so forth.

To make progress in such a difficult situation, our technique exploits dependency relationships between the attributes in each table. For instance, in the first table in Table 1, there will exist some degree of dependency between $Model$ and $Tire$ if model partially determines the kinds of tires a car can use. On the other hand, perhaps $Model$ and $Color$ are likely to have very little interdependency. If we can measure the dependency between columns A and B and columns A and C, and compare them with the dependency measured from the first table, it may be possible to find the remaining correspondences.

As we can see, an advantage of using dependency relations in schema matching is that this approach does not require data interpretation; that is, even if the data sets in the schemas to be matched use different encodings, we can still measure the dependency relations. As a result, our proposed matching technique can be applied to multiple unrelated domains without retraining or customization. We refer to matching techniques that are not dependent of data interpretation as *uninterpreted matching*, and make this precise in the next definition.

Definition 1 (Interpreted versus uninterpreted matching). *Let* $M_1 = match(R(r_1, r_2, \ldots, r_n), S(s_1, s_2, \ldots, s_m))$ *and* $M_2 = match(R(r_1, r_2, \ldots, r_n), S(f_1(s_1), f_2(s_2), \ldots, f_m(s_m))$, *where* M_i *is a match result,* match *is a schema-matching algorithm,* R *is a source schema of size* n, S *is a target schema of size* m, *and finally* f_i *is an arbitrary one-to-one function applied to the values of column* i *in the target schema. We call the given matching algorithm,* match, *an* uninterpreted matching *if and only if the two match results* M_1 *and* M_2 *are identical regardless of the function* f_i. *Conversely, it is called an* interpreted matching *if the two results are different.*

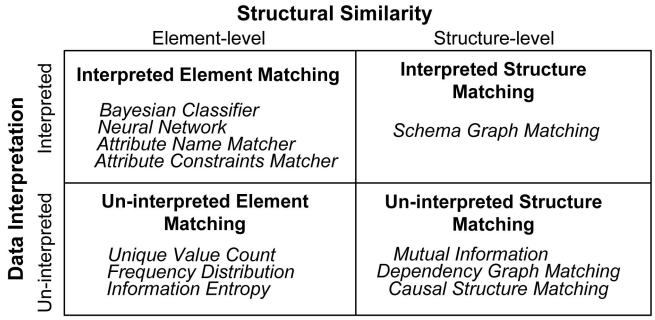


Fig. 1. Schema-matching technique classification.

In the following, it will also be useful to have the following definition, which captures the notion of whether the matching algorithm considers data elements in isolation or their relationship to other data elements.

Definition 2 (Element versus structure matching). *Structure-matching algorithms utilize the relationship between columns in a table, while element-matching algorithms only consider properties of individual columns.*

Fig. 1 illustrates classification of schema-matching techniques based on the use of data interpretation and structural similarity. While all four classes of techniques are valuable in different domains, we focus in this paper on uninterpreted structure matching. We propose a two-step technique that works in the presence of opaque attribute names and values. In this paper, we are making the following contributions:

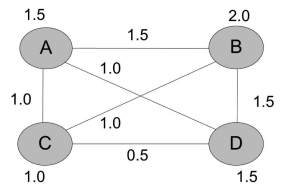
- We introduce a new criterion, *data interpretation*, in classifying schema-matching techniques. Along with structural similarity, we classify schema-matching techniques into four categories. Using this classification, we identify a new problem class that has not been addressed by existing techniques.
- We introduce a new two-step schema-matching technique that takes into account the dependency relations among the attributes.
- We reduce a schema-matching problem to a traditional graph-matching problem by capturing hidden dependencies between attributes and structuring them as a labeled graph.
- In order to deal with the computational complexity of the graph-matching step, we propose several efficient approximation algorithms that work for our schema-matching framework.
- We validate our approach with an experimental study, the results of which suggest that such an approach can be a useful addition to a set of (semi) automatic schema-matching techniques. Our experiments also show that, by exploiting relationships between columns, our techniques can do much better than a technique that only considers statistical properties of individual columns.

A	B	C	D
a1	b2	c1	d1
a3	b4	c2	d2
a1	b1	c1	d2
a4	b3	c2	d3

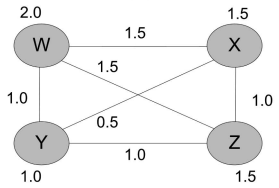
(a)

W	X	Y	Z
w2	x1	y1	z2
w4	x2	y3	z3
w3	x3	y3	z1
w1	x2	y1	z2

(b)



(c)



(d)

Fig. 2. Two input table examples and their dependency graphs. A weight on an edge represents mutual information between the two adjacent attributes and a weight on a node represents entropy of the attribute (or equivalently, self-information $MI(A;A)$). (a) Example table S_1. (b) Example table S_2. (c) Dependency graph G_1 of S_1. (d) Dependency graph G_2 of S_2.

The rest of this paper is organized as follows: Section 2 describes the two-step uninterpreted structure-matching technique. Section 3 presents efficient approximate algorithms for graph-matching problems arising in our problem context. Section 4 validates the framework with an experimental study. Section 5 presents related work. Lastly, Section 6 concludes this paper and identifies future work.

2 Uninterpreted Matching

In this section, we describe in detail our uninterpreted structure-matching technique. The algorithm takes two table instances as input and produces a set of matching node pairs. Our approach works in two main steps as shown below:

1. $G_1 \leftarrow \mathbf{Table2DepGraph}(S_1)$;
   $G_2 \leftarrow \mathbf{Table2DepGraph}(S_2)$ and
2. $(G_1(a), G_2(b)) \leftarrow \mathbf{GraphMatch}(G_1, G_2)$,

where $S_i = \text{input table}$, $G_i = \text{dependency graph}$, $(G_1(a), G_2(b)) = \text{matching node pair}$.

The function **Table2DepGraph**() in the first step transforms an input table like the one shown in Fig. 2a into a dependency graph shown in Fig. 2c. The function **GraphMatch**() in the second step takes as input the two dependency graphs generated in the first step and produces a mapping between corresponding nodes in the two graphs. The two steps are described in detail later in this section.

2.1 Preliminaries

To construct a dependency graph, we use mutual information and entropy, which are defined as follows:

Definition 3 (Mutual information [16]). *Let X and Y be two attributes with alphabets $\aleph$ and $\Im$, respectively. Consider some joint probability distribution $p(x,y)$ and marginal probability distributions $p(x)$ and $p(y)$ over two attributes. We define the mutual information of X and Y as*

$$MI(X;Y) = \sum_{x\in\aleph}\sum_{y\in\Im} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}.$$

Definition 4 (Entropy [16]). *Let X be an attribute with alphabet $\aleph$, and consider some probability distribution $p(x)$ of X. We define the entropy $H(X)$ by*

$$H(X) = -\sum_{x\in\aleph} p(x)\log p(x).$$

Note that both entropy and mutual information are functions of probability distributions and, thus, are independent of the actual values of attributes. This property allows them to be used in uninterpreted matching. Mutual information describes the correlation between the two attributes' probability distributions using a nonnegative real number. It measures the amount of information captured in one attribute about the other. This becomes more intuitive when we consider the relationship between mutual information and entropy. To explain this relationship, we need one more basic definition, that of conditional entropy.

Definition 5 (Conditional entropy [16]). *Let X and Y be two attributes with alphabets $\aleph$ and $\Im$, respectively. We define the conditional entropy of X and Y as*

$$H(X|Y) = \sum_{x\in\aleph}\sum_{y\in\Im} p(x,y)\log p(x|y).$$

Conditional entropy $H(X|Y)$ measures the uncertainty of attribute X given knowledge of attribute Y. It is a nonnegative real number and becomes zero when $X = Y$ or when there exists a functional dependency from Y to X. In these cases, no uncertainty exists for attribute X. On the other hand, if the two attributes X and Y are independent, the conditional entropy $H(X|Y)$ equals $H(X)$. We can now redefine the mutual information formula using entropy and conditional entropy:

$$\begin{aligned}
MI(X;Y) &= \sum_{x\in\aleph}\sum_{y\in\Im} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} = MI(Y;X)\\
&= \sum_{x\in\aleph}\sum_{y\in\Im} p(x,y)\log\frac{p(x|y)}{p(x)}\\
&= \sum_{x\in\aleph}\sum_{y\in\Im} p(x,y)\log p(x|y)\\
&\quad - \sum_{x\in\aleph}\sum_{y\in\Im} p(x,y)\log p(x)\\
&= \sum_{x\in\aleph}\sum_{y\in\Im} p(x,y)\log p(x|y)\\
&\quad - \sum_{x\in\aleph} p(x)\log p(x)\\
&= H(X) - H(X|Y) = H(Y) - H(Y|X).
\end{aligned}$$

As we can see in the equation, mutual information measures the reduction in uncertainty of one attribute due to the knowledge of the other attribute. In other words, it measures the amount of information that one attribute contains about the other. It is zero when two attributes are independent, and increases as the dependency between the two attributes grows. Note that mutual information of an attribute with itself (called self-information), $MI(X;X)$, is equivalent to the entropy of X, i.e., $H(X)$.

2.2 Modeling Dependency Relation

Consider the example illustrated in Fig. 2. Figs. 2a and 2b show two four-column input tables, and Figs. 2c and 2d show the corresponding dependency graphs. The **Table2DepGraph**() function produces such dependency graphs by calculating the pairwise mutual information over all pairs of attributes in a table and structuring them in an undirected labeled graph. For instance, each edge in the dependency graph G_1 (Fig. 2c) has a label indicating mutual information between the two adjacent nodes; for example, the mutual information between nodes A and B is 1.5, and so on. The label on a node represents the entropy of the attribute, which is equivalent to its mutual information with itself or self-information. Hence, we can model our dependency graph in a simple symmetric square matrix of mutual information, which is defined as follows:

Definition 6 (Dependency graph). *Let S be a schema instance with n attributes and $a_i (1 \leq i \leq n)$ be its ith attribute. We define dependency graph of schema S using square matrix M by*

$$M = (m_{ij}),\ where\ m_{ij} = MI(a_i; a_j), 1 \leq i, j \leq n.$$

The intuition behind using mutual information as a dependency measure is twofold: 1) it is value independent; hence, it can be used in uninterpreted matching and 2) it captures complex correlations between two probability distributions in single number, which simplifies the matching task in the second stage of our algorithm.

2.3 Matching Strategies

In this section, we focus on the second half of the schema-matching process: **GraphMatch**(). Before we delve into the main discussion, let us first examine the types of cardinality constraints that we need to consider in schema matching. Let A and B be two input schemas that we are trying to match. We consider three types of cardinality constraints:

- Bijective mapping ([1, 1]-[1, 1], in UML notation): Each attribute in A has a unique match in B, and vice versa. This corresponds to a case in which we know that the tables that we are trying to map have the same number of attributes, so the problem is just finding a correspondence between the attributes.
- Injective mapping ([0, 1]-[1, 1]): Each attribute in A has a unique match in B, while each attribute in B either has a unique match in A or remains unmatched. This corresponds to a case in which we know that table A's attributes are a subset of table B's, and so we have to discover this subset and then decide how to map attributes within this subset.
- Partial mapping ([0, 1]-[0, 1]): Each attribute in A either has a unique match in B or remains unmatched, and vice versa. This corresponds to the most general and difficult case in which we do not know which attributes of A map to B, nor do we even know how many attributes of A map to B. In this case, we need to find the best subset of attributes of A to map to B, and also need to find how this subset of A should be mapped.

In what follows, we will use distance metrics to evaluate the quality of matching. A distance is assigned to each instance of mapping between schema elements, and the goal is to find a mapping that optimizes the distance, i.e., minimize it or maximize it, depending on how the distance metric is defined. Bijective mappings and injective mappings both guarantee that all attributes in schema A will find matches in schema B, whereas partial mappings do not. Because of this, some distance metrics that work for bijective and injective mappings do not work for partial mappings. Let us formally define the class of such metrics:

Definition 7 (Monotonicity of distance metrics). *Let A and B be two dependency graphs with sizes (# of nodes) n and m, respectively, where $n \leq m$. Let $D_p(A, B)$ be the distance of best matching for two p node subgraphs of A and B. The distance metric $D_p(A, B)$ is monotonic if and only if $D_p(A, B) \geq$ (or $\leq$) $D_{p+1}(A, B)$ for all graphs A and B, and for all p in $1 \leq p \leq n - 1$.*

Monotonic metrics are not suitable for partial mapping because they reach their best score after either one attribute has been matched or all attributes have been fully matched depending on their direction of monotonicity, and hence, they will never produce a mapping in between (this problem does not arise with the bijective and injective mapping problems because the number of matches is known by definition). To see this, suppose we are matching two schemas $R(r_1, r_2, \ldots, r_n), S(s_1, s_2, \ldots, s_m)$. With a metric whose cost increases monotonically as the size of matching grows, some pair of columns will be chosen first as being the best match; suppose this is r_i matched to s_j, and that the cost of this match is c. With such metric, we can never improve upon c, and the matching algorithm will just return that the "best" match is r_i and s_j, in effect not even considering matchings for additional columns.

This is not appropriate for the partial mapping problem. Therefore, we need to be careful with metric selection in case of partial mapping. In this paper, we consider two basic distance metrics, one monotonic, the other not monotonic. Clearly, these are not the only possible metrics, and finding better metrics is an interesting area for future research. Consider the following basic distance metric.

Definition 8 (Euclidean distance metric). *Let A and B be two equal size dependency graphs, and a_{ij} and b_{ij} be the mutual information between the node i and j in graphs A and B, respectively. Let m be an index that maps a node in graph A to the matching node in graph B (i.e., $m(node\ in\ A) = matching\ node\ in\ B$). We define the euclidean distance metric for graphs A and B as*

$$D_M^U(A, B) = \sqrt{\sum_{i,j} (a_{ij} - b_{m(i)m(j)})^2}.$$

As we can see in the definition, the euclidean distance metric is monotonic; that is, the distance between two input graphs increases monotonically as the number of matches increases. Hence, we cannot use the metric on partial mapping problems. As we pointed out, we need a non-monotonic distance metric for partial mapping. Here is one such metric.

Definition 9 (Normal distance metric). *Let α be some positive constant. Similarly, we can define the normal distance metric for graphs A and B as*

$$D_M^N = \sum_{i,j}\left(1 - \alpha\frac{|a_{ij} - b_{m(i)m(j)}|}{a_{ij} + b_{m(i)m(j)}}\right).$$

In the second term of the subtraction, we normalized the difference of two pairing mutual information values by dividing by the sum of the two values. The intuition behind this normalization is that, for example, mutual information values 8 and 9 are likely to indicate a better match than the pairs 1 and 2 because the relative error in the latter is much greater than it is in the former. We refer to this normalized term, $|a_{ij} - b_{m(i)m(j)}|/(a_{ij} + b_{m(i)m(j)})$, as normal distance.

The normal distance falls in the range of [0, 1] because the mutual information is nonnegative real number. If we assume the mutual information values are uniformly distributed and we randomly choose two of them, the expected value of normal distance is $1/3$. Now, consider the control parameter α. In case of $\alpha = 3$, the expected value of whole distance metric becomes 0. In such cases, the mapping of randomly chosen two attributes will not contribute to the distance metric. Conversely, if the two attributes map correctly, the mapping will positively contribute to the distance metric.

By changing the parameter α, we can control the behavior of the distance metric. As we increase the α gradually from the original value, say 3, we will see the random mapping assignments start to contribute negatively to the distance metric. As a result, the matching returned from the normal distance metric with large α is likely to be more conservative than that with small α. In other words, metric with large α returns smaller but high-confidence candidate matches, while the metric with small α returns larger but less confident candidates.

Now, recall that one of our goals was to determine if mutual information matching is necessary, or whether entropy-only mapping was sufficient. To address this issue, we need an entropy-only version of the two distance metrics.

Definition 10 (Entropy-only euclidean distance metric). *Let A and B be two tables with equal number of attributes, and a_i and b_i be the entropies of attribute i in tables A and B, respectively. Let m be an index that maps an attribute in table A to the matching attribute in table B. We define the entropy-only euclidean distance metric for tables A and B as*

$$D_E^U(A, B) = \sqrt{\sum_i \left(a_i - b_{m(i)}\right)^2}.$$

Definition 11 (Entropy-only normal distance metric). *Similarly, we can define the entropy-only normal distance metric for graphs A and B as*

$$D_E^N(A, B) = \sum_i\left(1 - \alpha\frac{|a_i - b_{m(i)}|}{a_i + b_{m(i)}}\right).$$

The entropy-only matching works mainly in the same way as the mutual information-based matching. It matches the attributes across the two input tables by finding the mapping that optimizes the entropy-only metric.

Let us now examine the search (or graph matching) algorithms we will use. Because of the complexity of the problem, an exact search algorithm may not be a pragmatic solution. In practice, we can use an approximate search algorithm that trades off the accuracy of matching and the computational complexity. A large volume of literature has been devoted to finding such efficient, yet accurate graph-matching approximations. The two key questions that we want to answer are as follows:

- Would the proposed schema-matching technique work given the assumption that we have a perfect graph-matching algorithm?
- Is there an efficient algorithm for our matching problem?

The first question asks if the proposed framework is technically sound to produce accurate matching. In the experiments, we showed that using just the dependency relations among attributes, the proposed framework successfully produced high-accuracy matchings across tables. The second question is if the task of matching two graphs can be done efficiently. In order to address this problem, we investigated several approximate graph-matching algorithms that work for our problem context, as presented in the following section.

3 Weighted Graph-Matching Algorithms for Schema Matching

In this section, we investigate efficient approximation algorithms for the graph-matching problem in the second step of our approach. We focus our discussion particularly on the bijective mapping problem for two reasons. 1) The solution to this problem can be used as an integral part of the general solutions for the other two problems because the other problems can be formulated with multiple bijective mappings. For example, an injective mapping between graphs S (m nodes) and T (n nodes, where $m > n$) can be solved by finding an n node subgraph of S that minimizes the bijective mapping distance to T. The partial mapping problems can be formulated similarly. Of course, this may not be an ideal solution for them. Evaluating this approach versus approaches specifically tailored to the injective and partial mapping problems is an interesting area for future work. 2) The problem can be formulated in a clean mathematical optimization framework and because of that a large number of approximation algorithms have been developed. We will investigate a spectrum of the solutions covering a wide range of optimization techniques that can work for our problem context.

The bijective mapping problem between two dependency graphs S and T is essentially the problem of finding a permutation matrix P that minimizes the euclidean distance between the two dependency graphs'

adjacency matrices A_S and A_T, respectively, and can be formulated as

$$\min_{P\in\Pi}\left\|A_S - P^T A_T P\right\|_F^2,$$

where $\min_x f(x)$ is a function that finds an x that minimizes the $f(x)$, $\|\bullet\|_F^2$ is a square of a euclidean norm (or Frobenius norm, $\|A\|_F^2 = \sum_{i,j} a_{ij}^2$), and Π is the set of all permutation matrices.

The above problem is known as a weighted graph-matching problem (WGMP). WGMP is a purely combinatorial problem, and it is generally very difficult to find an exact 0-1 integral solution. The complexity of the problem is largely dependent on the choice of metric being optimized. However, we can show that, at the least, the general WGMP includes the graph isomorphism problem, for which no polynomial time algorithm has yet been found [25]. This leads us to focus on finding an efficient approximate algorithm that can produce a "nearly optimum" solution, rather than finding an exact search algorithm.

In what follows, we introduce five weighted graph-matching algorithms:

1. Umeyama's eigen-decomposition (ED) approach,
2. linear programming (LP),
3. convex quadratic programming (QP),
4. hill climbing (HC), and, finally,
5. branch and bound.

All but branch and bound are approximate algorithms. The first three are mathematical optimization approaches. The next, HC, is a heuristic iterative improvement algorithm and lastly, branch and bound is an exact search algorithm. Other competitive algorithms based on simulated annealing or deterministic annealing [27], [35], [50] were available, but we rejected them because they require a user to select tuning parameters manually, and we are seeking an automatic solution in our application.

3.1 Eigen-Decomposition Approach

Umeyama [54] introduced a polynomial time approximate algorithm for WGMP in the context of a vision problem. The proposed algorithm relaxes the original problem of finding the permutation matrix P to the problem of finding an orthogonal matrix X that minimizes the metric. The algorithm then finds an approximate solution for the original problem by manipulating the solution obtained from the relaxed problem. The relaxed problem can be written as

$$\min_{P\in\Pi}\left\|A_S - P^T A_T P\right\|_F^2 \geq \min_{X\in\Omega}\left\|A_S - X^T A_T X\right\|_F^2,$$

where Π is a set of all permutation matrices, and Ω is a set of all orthogonal matrices. Note that a permutation matrix is a special case of an orthogonal matrix, i.e., $\Pi \subset \Omega$.

Now, let $a_1 > a_2 > \ldots > a_n$ and $b_1 > b_2 > \ldots > b_n$ be the eigenvalues of matrices A_S and A_T, respectively, and their EDs be given by

$$U_S^T A_S U_S = \Lambda_S,$$
$$U_T^T A_T U_T = \Lambda_T,$$

where $\Lambda_S = diag(a_i)$ and $\Lambda_T = diag(b_i)$, and U_S and U_T are the orthogonal matrices that diagonalize A_S and A_T, respectively. The orthogonal minimizer X is obtained as follows:

$$\begin{aligned}
&\min_{X\in\Omega}\left\|A_S - X^T A_T X\right\|_F^2 \\
&\quad = \min_{X\in\Omega}\left\|U_S^T A_S U_S - U_S^T X^T U_T U_T^T A_T U_T U_T^T X U_S\right\|_F^2 \\
&\quad = \min_{X\in\Omega}\left\|\Lambda_S - U_S^T X^T U_T \Lambda_T U_T^T X U_S\right\|_F^2 \\
&\quad = \min_{X\in\Omega}\left\|\Lambda^T - \Theta^T \Lambda_T \Theta\right\|_F^2,
\end{aligned}$$

where $\Theta = U_T^T X U_S$. Since the eigenvalues in the two diagonal matrices Λ_S and Λ_T are ordered, to minimize the above metric, Θ must be an identity matrix. The orthogonal minimizer X is therefore, $U_T U_S^T$.

If the two graphs, S and T, were substantially close each other, the minimizer $X = U_T U_S^T$ would be approximately a permutation matrix, and rounding it up to a permutation matrix P of the original WGMP formulation would be trivial. Unfortunately, however, in most practical cases of our application, rounding up from X to P is not so straightforward. Umeyama [54] proposed, first, to calculate $X' = |U_T\|U_S^T|$ and then, to solve a linear assignment problem over the new matrix X' by treating X' as a weight matrix with entries representing weights associated with each assignment of nodes from the graph S to T. A standard assignment algorithm can be used to solve the assignment problem. We used the Hungarian method [1] in our experiments.

3.2 Linear Programming Approach

Almohamad and Duffuaa [3] introduced an LP approach for the WGMP. Whereas Umeyama relaxed the permutation matrix to an orthogonal matrix, their algorithm relaxes the original problem of finding a permutation matrix P to the problem of finding a doubly stochastic matrix X that minimizes the distance between the graphs. A doubly stochastic matrix $X = (x_{ij})$ has linear constraints as follows:

$$x_{ij} \geq 0, \sum_i x_{ij} = 1, \sum_j x_{ij} = 1, \text{ for all } i \text{ and } j.$$

Obviously, a permutation matrix is a special case of a doubly stochastic matrix. Another interesting aspect of this approach is that it optimizes the L_1 distance metric ($\|A\|_1 = \sum_{i,j} |a_{ij}|$) and not the L_2 (euclidean) distance metric. This essentially makes it possible to formulate WGMP as an LP optimization problem. The problem of finding a permutation matrix P that minimizes the L_1 distance between the two graphs S and T is

$$\min_{P\in\Pi}\left\|A_S - P^T A_T P\right\|_1.$$

Since P is a permutation matrix, the above can be rewritten to

$$\min_{P\in\Pi}\|PA_S - A_T P\|_1.$$

Now, we relax the problem by replacing P with a doubly stochastic matrix X as

$$\min_{P\in\Pi}\|PA_S - A_T P\|_1 \geq \min_{X\in\Sigma}\|XA_S - A_T X\|_1,$$

where Σ is the set of all doubly stochastic matrices. Reducing it to an LP problem, two goal variables U and V are introduced such that

$$XA_S - A_T X + U - V = 0,$$

where $U \geq 0$ and $V \geq 0$. Now, WGMP can be formulated as an LP problem as follows:

$$\begin{aligned} &\min_{X,U,V} \sum_{i,j} U + V \\ &\text{s.t. } XA_S - A_T X + U - V = 0, \\ &\sum_i x_{ij} = 1, \\ &\sum_j x_{ij} = 1, \\ &X \geq 0, U \geq 0, V \geq 0. \end{aligned}$$

Solving this linear program, we get a doubly stochastic matrix X that minimizes the L_1 distance between the two graphs S and T. The resulting doubly stochastic matrix X can be rounded up to a permutation matrix P by solving a linear assignment problem over the matrix X, as was done in the previous section.

3.3 Convex Quadratic Programming Approach

Anstreicher and Brixius [5] introduced a new bound for the quadratic assignment problem (QAP) [14]. The new bound is obtained by relaxing QAP to a convex QP optimization problem. Schellewald et al. [52] showed that WGMP can be reduced to QAP and solved QAP by minimizing the new bound introduced in [5]. In much the same way, we can relax WGMP directly to a convex QP problem. Unlike the LP relaxation, we do not need to choose an alternative metric. The relaxation process is given below.

As was done in the LP formulation, we can rewrite the original problem

$$\min_{P \in \Pi} \left\| A_S - P^T A_T P \right\|_F^2,$$

to an equivalent form

$$\min_{P \in \Pi} \| PA_S - A_T P \|_F^2.$$

Now, the problem of finding a permutation matrix P is relaxed to a problem of finding a doubly stochastic matrix X as

$$\min_{P \in \Pi} \| PA_S - A_T P \|_F^2 \geq \min_{X \in \Sigma} \| XA_S - A_T X \|_F^2.$$

Expanding the Frobenius norm square in the relaxed problem, we get a quadratic formula with variables of x_{ij}. The expanded quadratic formula can be written to

$$\| XA_S - A_T X \|_F^2 = vec(X)^T Q vec(X),$$

where $vec(X)$ represents a vector obtained by stacking the columns of X on top of one another in the natural order, and Q represents a coefficient matrix each entry of which is a coefficient of a corresponding term from the expanded quadratic formula. The matrix Q can be directly calculated from the two adjacency matrices A_S and A_T.

Finally, the QP formulation of WGMP is given below:

$$\begin{aligned} &\min_X vec(X)^T Q vec(X) \\ &s.t., \ \Sigma_i x_{ij} = 1, \Sigma_j x_{ij} = 1, X \geq 0. \end{aligned}$$

Note that the coefficient matrix Q is positive semidefinite, and therefore, the above formulation is a convex quadratic program. A convex QP problem can be solved efficiently using algorithms such as Interior-Point methods [4], [55].

Solving this quadratic program, we get a doubly stochastic matrix X that minimizes the euclidean distance between the two graphs S and T. As was done in the previous two approaches, we used the Hungarian method [1] in our experiments to round up the resulting doubly stochastic matrix X to a permutation matrix P.

Algorithm 1. Overview of branch and bound.

Input : Two graphs S and T
Output: Minimum distance permutation between the two graphs

Run a fast approximate algorithm to initialize the following variables; Current minimum distance ← distance obtained from the above; Current minimum distance permutation ← permutation obtained from the above;
Construct permutation tree for traversal;
while *Current permutation prefix ← path from the root to the current node in the permutation tree;*
if *the current node is a leaf node* **then**
  Calculate the distance of the graphs corresponding to the current permutation;
  if *the distance is smaller than the current minimum distance* **then**
    Current minimum distance ← current distance;
    Current minimum distance permutation ← current permutation;
  end
  if *next sibling exists* **then**
    Traverse to the next sibling;
  else
    Backtrack until the next available node is found;
  end
else
  Calculate the distance of the subgraphs that correspond to the current permutation prefix;
  Estimate the minimum distance of the remainder of the graphs;
  if *the sum of the two distances obtained above exceeds the current minimum distance* **then**
    if *next sibling exists* **then**
      Jump to the next sibling;
    else
      Backtrack until the next available node is found;
    end
  else
    Traverse down the tree;
  end
end
do
end
Return the current minimum distance and permutation;

3.4 Hill-Climbing Approach

So far, we have considered three deterministic approximation algorithms for WGMP. All of them are based on the relaxation of the original problem to an algebraic optimization framework. We now introduce a simple nondeterministic, iterative improvement algorithm, HC [51].

The HC algorithm is simply a loop that moves, in each state transition, to a state where the most improvement can be achieved. A state represents a permutation that corresponds to a mapping between the two graphs. We limit the set of all states reachable from one state in a state transition, to a set of all permutations obtained by one swapping of any two nodes in the permutation corresponding to the current state. The algorithm stops when there is

no next state available that is better than the current state. As we can see, it is nondeterministic; depending on where it starts, even for the same problem, the final states may differ. To avoid being stuck in a local minimum after an unfortunate run, the usual practice is to perform some number of random restarts.

3.5 Branch and Bound Approach

Due to the combinatorial nature of the problem, an exact search algorithm would hardly be practical, but we present here one based on the branch and bound method [1] for the purpose of comparison. As we will see in the experiments, this approach cannot handle problems as large as those handled by the approximate algorithms. Our implementation of the branch and bound approach for WGMP is shown in Algorithm 1.

The branch and bound in Algorithm 1 generates an initial permutation for the mapping using a fast approximate algorithm. It then constructs a permutation tree using the initial mapping as the seed. It traverses the tree in depth first order while improving the distance bound. If the current prefix produces a distance worse than the current bound, it branches to the next sibling without exploring the subtree. When it reaches to the leaf it computes the distance with the current permutation. If it is better than the current bound, it updates the bound and backtracks to the next available permutation.

So far, we have investigated algorithms for WGMP. These algorithms take as input two dependency graphs generated in the first step and find the mapping between them. Among the three mathematical optimization algorithms, the ED approach is the fastest. It runs asymptotically in the order of n^3, where n is the number of nodes in an input graph. The other two algorithms, LP and convex QP approaches, run asymptotically in the order of n^6 for the same n. The branch and bound is obviously the slowest as it performs the exact search. Lastly, the HC algorithm is a heuristic interactive improvement algorithm, and its running time largely depends on the number of restarts, seed selection, and the characteristics of gradient surface it performs search on. The experimental validation of the algorithms is given in the following section.

4 Experiments

In this section, we present the results of schema-matching experiments using our proposed approach. The validation is performed in two steps. In the first step, we attempt to address the first question (in Section 2) asking if the proposed schema-matching framework works given the assumption that we have a perfect graph-matching algorithm. The experimental results for this problem are presented in Section 4.1. In the second step, we address the remaining question asking if there is an efficient algorithm for our matching problem. The results for this problem are given in Section 4.2.

4.1 Validating the Framework Using Exact Search

In this section, we validate the framework using exact search in order to prove that the framework works provided that a perfect graph-matching algorithm exists. An exact search algorithm may not be an option in practice but we used it for the experiments in this section because we wanted to measure the accuracy of uninterpreted matching precisely and the use of approximation might affect the measurement to some degree, due to the algorithm's own approximation error. We ran experiments over two real-world data sets from different domains. For each type of cardinality constraint, we performed a set of experiments using different input and sample sizes.

4.1.1 Data Sets

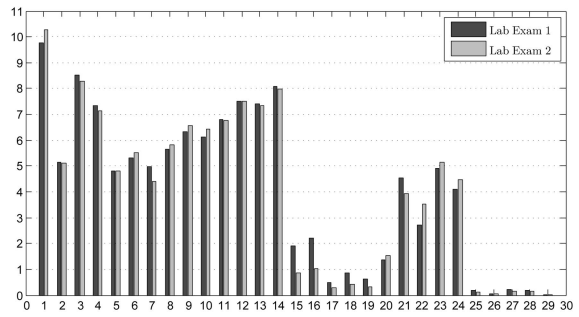
We used real-world data sets from two different data domains: medical data and census data. The medical data set we used in our experiments contains patients' lab exam results for diagnosing thrombosis.[1] Fig. 3a shows the measured entropies of 30 randomly chosen attributes of the thrombosis lab exam data, and Fig. 3c shows a fragment of the first 10 (out of the 30) attributes' data values. The original table contains 12 years worth of patient exam records, which is approximately 50,000 tuples, and each tuple consists of 44 attributes representing test types. The column data types are mostly numeric, and a significant portion of the table is left blank (see attributes 15-30 in Fig. 3a). Our basic experimental technique with the medical data set was to range partition the original table into two subtables based on exam dates (column 1) and to use these two subtables for experiments. We "pretended" that these subtables were two different tables that needed to have their schemas mapped. Obviously, we "knew" the correct answer for the mapping, but the mapping algorithm did not.

For our second data set, we used census data. Figs. 3b and 3d show attribute entropies and a table fragment from the census data set, respectively. We used two state census data files, CA and NY, in our experiments.[2] Each table consists of 240 attributes. We ran the experiments over a randomly chosen set of 30 attributes. Note that in Fig. 3d, attributes 8 and 9 are duplicated. The original census data files have some number of duplicate columns and two of them happened to be in the 30 attributes randomly chosen for our experiments. Evaluating the match results, we did not count mappings like $NY9$ to $CA8$ a correct match; therefore, the accuracy of matching was somewhat reduced degree by these duplicate columns.

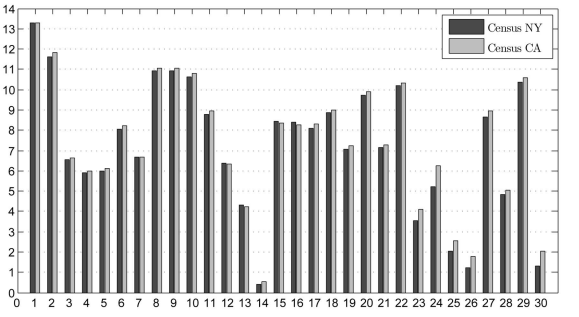
4.1.2 Bijective Mapping

Fig. 4 presents the results of bijective schema matching. We ran the experiment while increasing the number of attributes in two input tables to be matched. For each table width, 2 to 20, we iterated the measurement 50 times with randomly chosen subsets of attributes and averaged the results. Entropy-only matching results (labeled ET) are also presented to show the improvements obtained by taking into accounts of correlations between the attributes, which is given in the results of mutual information-based matching (labeled MI). Furthermore, we tested both

1. PKDD 2001 Discovery Challenge on Thrombosis Data. http://lisp.vse.cz/challenge/pkdd2001/.
2. US Census Bureau. ftp://ftp2.census.gov/census_2000/datasets/.



(a)



(b)

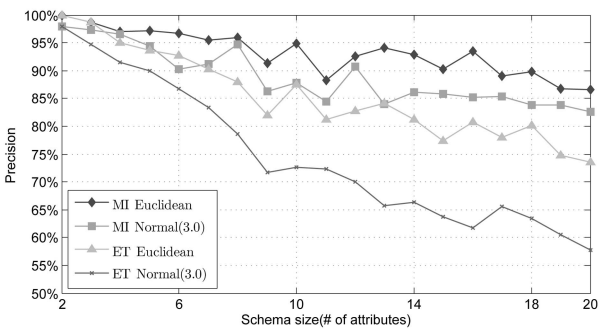
1	2	3	4	5	6	7	8	9	10
970709	23	530	104	6.4	4	14	0.5	232	100
971022	26	564	108	6.8	5.3	13	0.55	250	103
971224	25	483	90	6.5	5.1	15	0.62		
980120	26	578	101	7	4.6	16	0.49	224	93
980217	34	521	98		5.3	10	0.62	234	111
980316	35	540	95	6.5	5.7	13	0.55	242	112
980511	30	541	101	6.5	5.8	14	0.58	236	195
980630	26	559	102	6.9	5	13	0.55	240	89
980825	26	220	200	6.9	5.2	9.8	0.6	208	186
980922	32	239	192	7.3	5.6	13.2	0.63	270	152

(c)

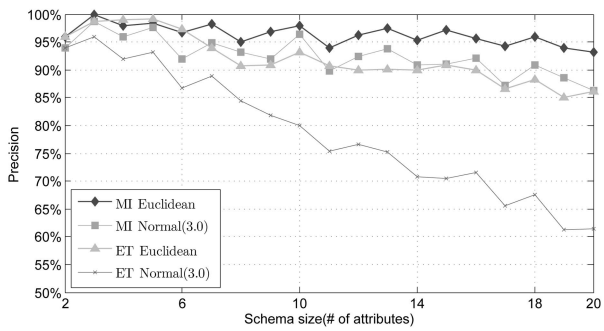
1	2	3	4	5	6	7	8	9	10
18091	1063	10	9	9	41	15	368	368	288
17511	3281	25	21	40	89	59	1211	1211	796
609	3424	29	13	15	148	26	1055	1055	861
3861	2884	18	7	4	114	11	670	670	568
18614	1478	12	10	15	40	16	630	630	459
3999	2414	29	16	27	87	21	967	967	753
5283	2385	42	17	39	46	40	968	968	622
21892	3053	28	17	16	99	33	1400	1400	1130
18554	14506	160	131	92	499	170	5084	5084	3965
12491	823	2	0	1	39	4	240	240	226

(d)

Fig. 3. Attribute entropies of two data sets. (a) Thrombosis lab exam 10,000 (# of tuples) samples. (b) Census data 10,000 samples. (c) First 10 columns of Lab Exam 1 fragment. (d) First 10 columns of Census CA fragment.



(a)



(b)

Fig. 4. Bijective mapping results. (a) Thrombosis lab exam 10,000 samples. (b) Census data 10,000 samples.

euclidean and normal distance metrics in both entropy-only and mutual information matching.

Fig. 4a shows the precision of match results using thrombosis lab exam 10,000 tuple samples. As we see in Fig. 4a, match results obtained from narrow tables are better than that from wider tables. As the tables get wider, the precision of matching deteriorates. Comparing the two matching techniques, the entropy-only matching combination shows much faster deterioration than mutual information matching. The best performer was the mutual information matching using the euclidean distance metric, and the worst was entropy-only matching using the normal distance metric.

Comparing two metrics, the euclidean distance metric works better than the normal distance metric in both the entropy-only and mutual information matching. We used 3.0 for the normal distance metric's control parameter α. However, the value of the control parameter has no effect in the match results in this case. As we mentioned in Section 2, the control parameter balances the precision and recall of the match results. Both precision and recall are, however, always the same in bijective mapping.

Fig. 4b shows the match results using the census data set 10,000 tuple samples. Although the overall precision is slightly better, the results look quite similar to those presented in Fig. 4a. Mutual information matching yielded superior results to entropy-only matching and the euclidean distance metric performed better than the normal distance metric. Mutual information matching with the euclidean metric produced a matching of approximately 93 percent accuracy when two 20 column tables were matched, in which on average, more than 18 attributes were correctly matched while only two mismatched. On the other hand, 85 percent accuracy was achieved by entropy-only matching using the same metric. In Fig. 4a with the lab exam data set, we had 86 percent and 74 percent accuracy for mutual information and entropy-only matching, respectively, which can be interpreted as 3 and 5 misses out of 20 true matches.

The results from census data were slightly better than those of the lab exam data. One explanation for this can be found in the entropy signature of the two data sets shown in Fig. 3. In Fig. 3a, we can see that the last six attributes, from 25 to 30, have very low entropy values. These are the

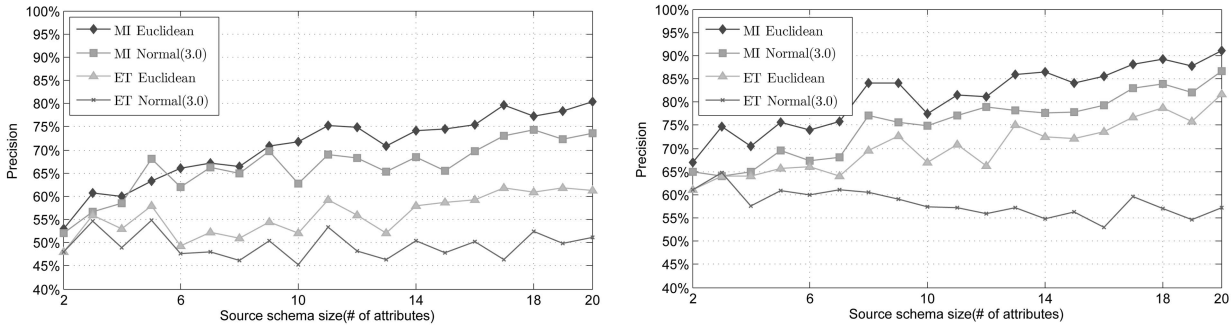


Fig. 5. Injective mapping results. Target schema size is kept constant at 22 attributes while source schema size varying. (a) Thrombosis lab exam 10,000 samples. (b) Census data 10,000 samples.

columns in the original data that have mostly null values. Because of the lack of information in them, these columns do not contribute much to the match results. By contrast, in the census data, only one such attribute exists, which is attribute 14 in Fig. 3b.

4.1.3 Injective Mapping

Fig. 5 illustrates the results of schema matching with the injective cardinality constraint. Fig. 5a shows the results from thrombosis lab exam data, and Fig. 5b shows census data. In this experiment, we kept the target schema size constant at 22 attributes while increasing the source schema size from 2 to 20 attributes. As was the case in the bijective mapping experiments, the census data match result is slightly better than that of the lab exam data. For example, when the schema size reached 20, census data yielded 91 percent precision while lab exam data turned out only 80 percent.

In both data sets, mutual information matching outperformed entropy-only matching. The precision of lab exam data matching was improved approximately 31 percent (from 61 percent in entropy only to 80 percent with mutual information), while precision in census data improved 12 percent (from 81 percent in entropy only to 91 percent for mutual information). We see that mutual information was more helpful for the lab exam data than it was for the census data. This is because in the lab exam data, more attributes had similar entropy, so that entropy-only mapping was more likely to get "confused." Turning now to compare our two metrics, euclidean and normal, the euclidean distance metric yielded better results overall in both data sets.

To summarize the situation up to this point, we have considered the performance of two matching methods and two distance metrics, and the results have been consistent with those in the bijective mapping case. However, there is a notable difference: the precision of matching in the injective case improves as the size of source schema increases, which is the opposite of what we saw in the bijective mapping. We turn now to explain this phenomenon.

Let us consider the matching as two-step process: selecting a subset of attributes from the target schema and searching for the correct permutation of this selected subset. The reason that the injective experiments had better performance with a larger source schema is that the first step is harder than the second. If the first step was easy, the results of the injective mapping experiments should have looked similar to that of the bijective experiments.

Suppose that the second step always returned the correct permutation. Then, the injective-matching problem reduces to choosing the correct attribute subset from the target schema. In fact, this assumption is not too far from the real situation because, as shown in the bijective mapping results, the second step indeed produces almost perfect results, especially when the number of attributes is small. For example, consider the case of finding 2 attributes out of 22 attributes. The total number of possible selections is 231, and one of them is the correct selection and 40 others have only one correct attribute (50 percent precision). The remaining 190 selections yield no match (therefore, 0 percent precision). Whereas, in case of finding 20 attributes out of 22, the maximum mismatch number is two; therefore, it will achieve 90 percent precision in the worst case. Considering this, it is easy to see why the precision improves in spite of the fast growing search space.

4.1.4 Partial Mapping

Fig. 6 illustrates the results of schema matching with the partial mapping cardinality constraint. Figs. 6a and 6c show the precision and recall of the thrombosis lab exam data results, and Figs. 6b and 6d shows the precision and recall of the census data results, respectively. In this experiment, we keep the size of both source and target schema constant at 12 attributes while varying the number of correct matches from 2 to 10 attributes. Unlike previous two cases, partial mapping requires a nonmonotonic distance metric because in this case, the size of source schema and the number of correct matches are not necessarily same. For the same reason, both precision and recall should be examined.

In this experiment, we used the normal distance metric with three different control parameter values: one, four, and seven. In Fig. 6a, $MINormal(1.0)$ represents mutual information matching using normal distance metric with $\alpha = 1$, and so on. Unlike the previous two cases, it is not easy to tell which approach dominates from the experiments. In fact, the choice of α is dependent on the application semantics. If an application prefers a small number of candidates with high confidence, then a larger α would be more suitable. In contrast, if the application is willing to accept relatively low confidence in match

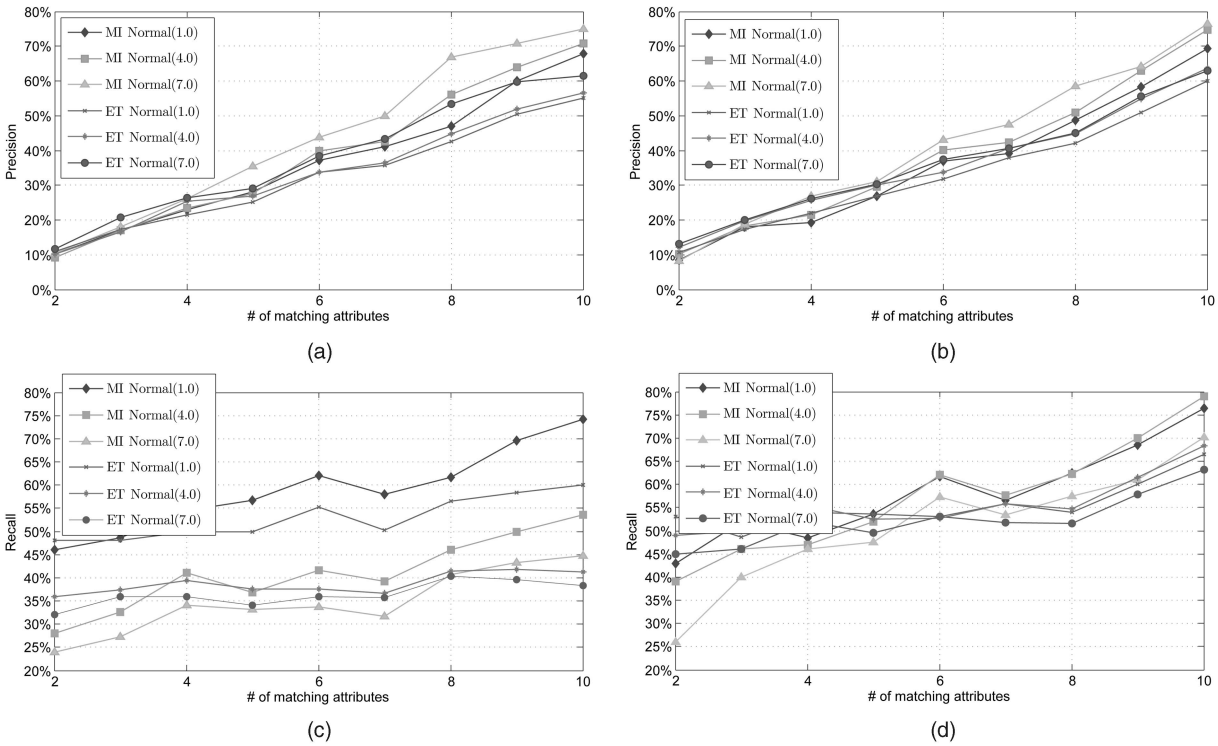


Fig. 6. Partial mapping results. The size of both source schema and target schema is set to 12 attributes, while the number of correct matches varies. (a) Precision of lab exam mapping results (10,000 samples). (b) Precision of census data mapping results (10,000 samples). (c) Recall of lab exam mapping results. (d) Recall of census data mapping results.

candidates but wants as many probable matches retrieved as possible, then smaller α should work better.

For instance, in Fig. 6a, $MINormal(7.0)$ achieved 75 percent precision where the two input schemas contain 10 true matches, while the same metric turned out only 45 percent recall at the same point in Fig. 6c. In other words, it produced candidate matches, 75 percent of which were correct, and the number of correct matches in the candidates was 45 percent of the number of total true matches. On the other hand, $MINormal(1.0)$ achieved approximately 67 percent precision, while it turned out 75 percent recall at the same point in the graph. It is intuitively clear that the normal distance metric with $\alpha = 1.0$ returned a larger number of candidates than the metric with $\alpha = 7.0$. The normal distance metric with $\alpha = 7.0$ (where $\#$ of matching attributes $= 10$) returned on average 6 candidates, while the metric with $\alpha = 1.0$ returned more than 11 candidates.

As was the case in the previous two scenarios, the performance on the census data set is slightly better than that of lab exam data. In Figs. 6b and 6d, $MINormal(4.0)$ achieved approximately 75 percent precision and 79 percent recall, where the number of matching attributes is 10. Comparing two matching methods, in the lab exam data set, mutual information matching improved entropy-only matching results by approximately 24 percent in both precision and recall, where the number of matching attributes was 10 and α was 1.0. In case of census data set, the improvement was 19 percent and 16 percent for precision and recall, respectively, at the same data point in Figs. 6b and 6d (i.e., $\#$ of matching attributes $= 10$) using $\alpha = 4.0$.

Unlike previous two scenarios, the search space for partial mapping remains same throughout the experiments with varying numbers of matching attributes. Although the search space did not change, the accuracy of results improved as the number of matching attributes increased. The explanation given for the injective mapping scenario applies here as well.

4.2 Evaluation of Approximate Matching Algorithms for Schema Matching

In this section, we try to address the second question asking if there is an efficient algorithm for matching that works for our problem context. We present the experimental results for evaluating the algorithms introduced in Section 3. We examined the five algorithms:

1. Umeyama's ED approach,
2. the LP approach,
3. the convex QP approach,
4. the HC approach, and finally,
5. the branch and bound algorithm.

For HC, we used five iterations, each from a randomly chosen starting point, and chose the best result from the five trials.

We implemented the algorithms using Matlab 6.5. A commercial optimization package, Mosek Matlab Toolbox V2.0 Build 20,[3] was used for the LP and QP approaches. We first formulated the WGMP as a linear or a quadratic programming problem, and then solved them using the optimization package. Experiments were performed on an

3. Mosek ApS. http://www.mosek.com/.

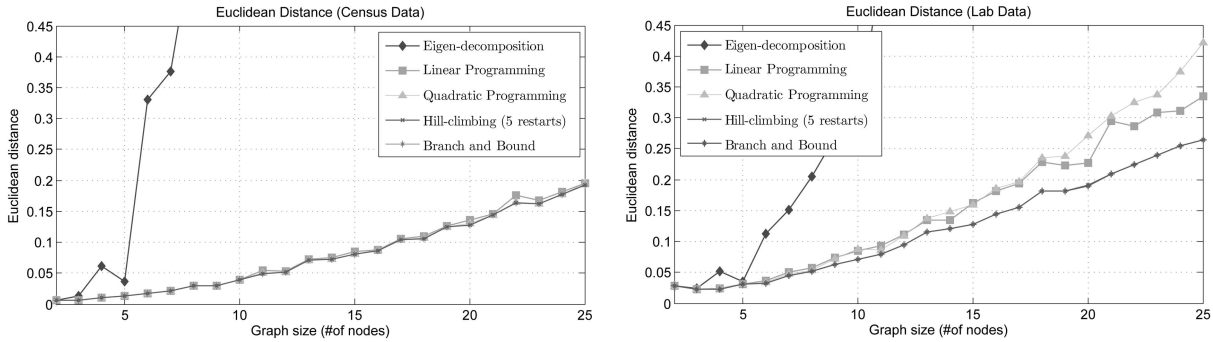


Fig. 7. Graph-matching algorithm performance comparison. Branch and bound represents the optimum distance matching. Others that are closer to branch and bound are the better performers. (a) Census data result. (b) Lab data result.

AMD Athlon XP 1.533 GHz machine with 1 Gbyte of memory, running Windows XP Professional. We iterated each experiment 50 times, and then averaged the results.

Fig. 7 presents the experimental results where we compared the euclidean distances obtained from each algorithm. Figs. 7a and 7b show the result obtained using the census data set and the lab data set, respectively. Each algorithm tries to find a mapping whose euclidean distance is closest to the optimum. The branch and bound algorithm is a special case. It is an exact search algorithm, and hence, it finds always the optimum mapping whose distance is the minimum among all possible mappings between the graphs.

The other four algorithms are approximate algorithms and find suboptimal solutions. We can evaluate the performance of an approximate algorithm by comparing the distance of a mapping it produces, to the distance of the optimum mapping produced by branch and bound. Algorithms yielding a distance closer to the optimum are the better performers. For experiments, we normalized the input graphs by dividing the adjacency matrices by the biggest entry in the two matrices.

In Fig. 7a, HC performed slightly better than the others and ED was the worst performer. Similarly, in Fig. 7b, HC outperformed the others and ED was the worst. The differences between the algorithms are illustrated more clearly in Fig. 8. Fig. 8 presents each algorithm's relative error measured against the optimum distance. The relative error is the difference between the distance obtained from an algorithm and the optimum distance, divided by the optimum distance.

It is clear in Fig. 8a that for the census data the best performer was HC and the next best performer was QP and then, LP and finally, ED. The algorithms also performed similarly for the lab data as shown in Fig. 8b. The only difference between the results on the two data sets was that the relative performance between QP and LP was not as clear as it was in Fig. 8a.

The computational complexity of an algorithm is another important factor to consider when we choose an algorithm. An exact search algorithm such as branch and bound would obviously be the best in terms of the accuracy but it could be too slow for some of the large problems. Fig. 9 illustrates the average running time of each algorithm. ED was the fastest among all. LP and QP finished a 20-node graph matching in slightly more than 1 second and a 25-node graph matching in about 3 seconds to 4 seconds. HC with five random restarts performed slightly better than both LP and QP.

Apparently, branch and bound was the slowest. In the census data test, it finished the 20-node graph matching in 3.5 seconds and the 25-node graph matching in about 72 seconds, while in the lab data test, the same problems took about 63 seconds and 601 seconds, respectively. The execution time of branch and bound is largely dependent on the input graphs. It runs faster with graphs that are closer each other (note that two census graphs are closer than lab graphs, see Fig. 7). It is because the distance

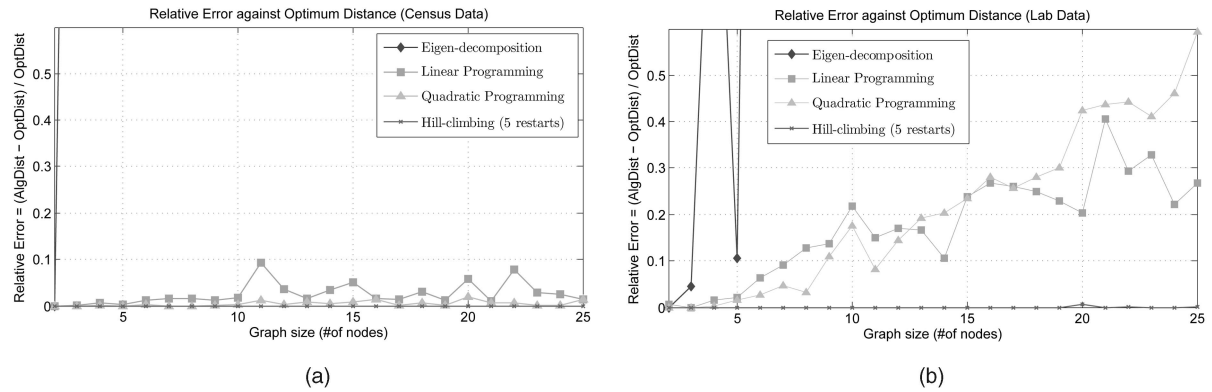


Fig. 8. Relative error against optimum distance. (a) Census data result. (b) Lab data result.

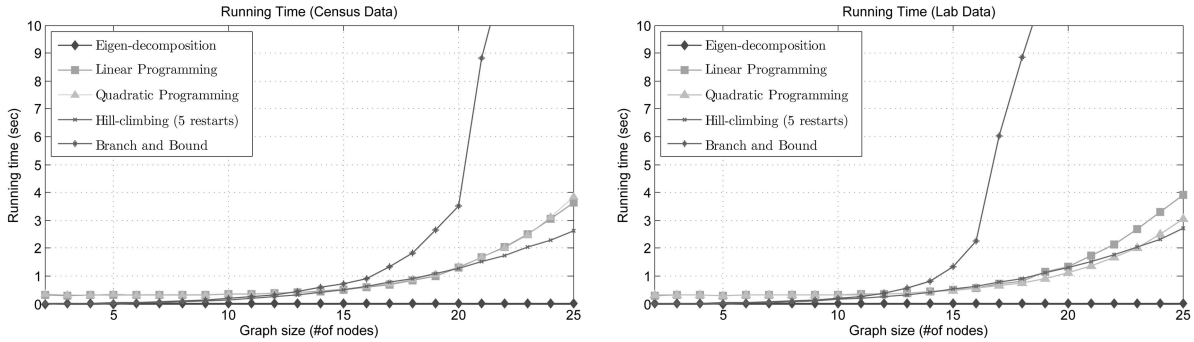


Fig. 9. Execution time of algorithms. (a) Census data result. (b) Lab data result.

between the graphs works as a bound for pruning the permutation tree branch and bound traverses, and the smaller bound is likely to prune more, earlier.

5 Related Work

Most previous work on schema matching have focused on developing interpreted matching techniques (see [48] and [53] for survey). Such techniques are largely dependent on identifying similarity in schema element names, common representation formats, or common data domains. Because our technique is based on uninterpreted matching, it can complement existing techniques and can be combined with traditional schema-matching systems.

Some proposed techniques employ machine learning. Li and Clifton proposed a neural network-based schema-matching prototype called SemInt [38], [39]. Berlin and Motro proposed Automatch, a technique based on machine learning with feature selection [9]. LSD and iMAP were proposed by Doan et al. [17], [18], [19]. They employ multistrategy learning to find mappings. He and Chang [29], [30] introduced a holistic schema-matching technique for discovering complex matches. Domshlak et al. [20] introduced a method to aggregate the matching results from ensemble of schema-matching algorithms. Although these systems are flexible, they rely on data interpretation, and therefore, they are not applicable to our problem domain.

Other work has considered rule-based schema matching. These include TranScm [46] and ARTEMIS [15]. Both TranScm and ARTEMIS are schema-based matching techniques and our uninterpreted instance-based technique can be combined with them to improve the accuracy of matching. Some other techniques represent a schema in a graph format and perform matching based on the structural similarity of the two graph representations. Cupid [40] and Similarity Flooding [41] fall into this category. Unlike our scheme, both Cupid and Similarity Flooding rely on schema-based structural similarity and, therefore, are not applicable to our problem domain.

Meanwhile, though they are not targeted to schema matching, many generic graph-matching algorithms have been developed in the theoretical computer science literature [3], [27], [52], [54]. These algorithms can be tuned to match our dependency graphs. We investigated some of these graph-matching algorithms in Section 3.

There has been a big thrust in research on schema mapping systems. Miller et al. introduced Clio [21], [23], [28], [32], [43], [56] that creates a mapping between two input schemas in an interactive fashion using user feedback. It produces as a mapping a view definition (mapping query) over the target schema. A metaquery engine then executes the mapping query and translates the data from the original schema onto the target schema. Our uninterpreted matching can complement Clio because Clio focuses on finding correspondences between data instances while uninterpreted matching focuses on finding mappings between schema elements.

A good deal of research has been conducted on building theoretical foundations of schema integration. Central to these efforts has been work on the notion of "equivalence" between two schemas [2], [6], [8], [34], [49]. Hull [34] introduced a formal definition of "equivalence" and "dominance" between two schemas, using relative information capacity. Using the result in [34], Miller et al. [44], [45] investigated a number of schema transformation and integration scenarios, and for each scenario, proposed a new definition of "correctness" that can be used in judging transformed schemas' correctness. Transformation from an original schema to an equivalent form is correct if the two schemas have identical information capacity (or expressive power).

In our work, we used mutual information and entropy to represent interaction between attributes. These concepts are popular in the information theory community and have been well accepted in other domains as well [16]. Although we found that mutual information is an effective tool for capturing dependencies in an uninterpreted manner, there exist other ways that this could be accomplished. One interesting approach would be to use Bayesian network structure learning [24], [26].

Bayesian networks capture dependency (or sometimes causal) relations between attributes in the form of conditional probability distributions. Another possible approach for capturing interaction between attributes would be to use integrity constraint inference techniques to infer functional and approximate dependencies among attributes [31], [33], [37]. It would be interesting to see how these approaches compare with our mutual information-based approach.

Finally, Bernstein et al. presented model management scenarios in their vision paper [11] and recently extended it to handle more expressive mappings and to include the mapping runtime in the model management framework [12]. They proposed a unified framework for applications to access underlying models using high-level operators such

as Match, Merge, Compose, Inverse, and ModelGen. A large body of research followed the model management initiative implementing the proposed operators [7], [10], [13], [22], [42]. The schema-matching technique reported in this paper works as a Match operator in model management. Developing remaining operators using our uninterpreted method would be an interesting area for future work.

6 Conclusion

We have proposed a two-step schema-matching technique that works even in the presence of opaque column names and data values. In the first step, we measure the pairwise attribute correlations in the tables to be matched and construct a dependency graph using mutual information as a measure of the dependency between attributes. In the second stage, we find matching node pairs across the dependency graphs by running a graph-matching algorithm.

To our knowledge, our work is the first to introduce an uninterpreted matching technique utilizing interattribute dependency relations. We have shown that while a single column uninterpreted matching such as entropy-only matching can be somewhat effective alone, further improvement was possible by exploiting interattribute correlations.

In this work, we also investigated approximation algorithms for the matching problem and showed that an efficient implementation can be possible for our approach. Among the algorithms we evaluated, the HC approach showed the most promising results. It found close to optimal solutions very quickly, suggesting that the graph-matching problems arising in our schema-matching domain are amenable to HC.

A good deal of room for future work exists. In our work, we have only tested two simple distance metrics—euclidean and normal. It is possible that more sophisticated distance metrics could produce better results. It would also be interesting to evaluate other dependency models using different uninterpreted methods.

Acknowledgments

Correspondence should be addressed to Jaewoo Kang. The authors would like to thank Jin-Yi Cai for helpful discussion. This paper is a substantially extended version of the authors' previous conference paper in [36]. This research was supported by US National Science Foundation Grants CSA-9623632, ITR 0086002, and in part by Korea University Grant, Microsoft Research Internet Services Grant, and the Second Brain Korea 21 Project Grant.

References

[1] *Operations Research: Deterministic Optimization Models.* Prentice Hall, 1995.

[2] J. Albert, Y.E. Ioannidis, and R. Ramakrishnan, "Conjunctive Query Equivalence of Keyed Relational Schemas," *Proc. 16th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS '97),* pp. 44-50, 1997.

[3] H.A. Almohamad and S.O. Duffuaa, "A Linear Programming Approach for the Weighted Graph Matching Problem," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 5, pp. 522-525, May 1993.

[4] E.D. Andersen and K.D. Andersen, "The Mosek Interior Point Optimizer for Linear Programming: An Implementation of the Homogeneous Algorithm," *Proc. High Performance Optimization Techniques (HPOPT),* 1997.

[5] K.M. Anstreicher and N.W. Brixius, "A New Bound for the Quadratic Assignment Problem Based on Convex Quadratic Programming," *Math. Programming,* vol. 89, pp. 341-357, 2001.

[6] P. Atzeni, G. Ausiello, C. Batini, and M. Moscarini, "Inclusion and Equivalence between Relational Database Schemata," *Theoretical Computer Science,* vol. 19, pp. 267-285, 1982.

[7] P. Atzeni, P. Cappellari, and P.A. Bernstein, "Modelgen: Model Independent Schema Translation," *Proc. 21st Int'l Conf. Data Eng. (ICDE '05),* pp. 1111-1112, 2005.

[8] C. Beeri, A.O. Mendelzon, Y. Sagiv, and J.D. Ullman, "Equivalence of Relational Database Schemes," *SIAM J. Computing,* vol. 10, no. 2, pp. 352-370, 1981.

[9] J. Berlin and A. Motro, "Database Schema Matching Using Machine Learning with Feature Selection," *Proc. 14th Int'l Conf. Advanced Information Systems Eng. (CAiSE '02),* pp. 452-466, 2002.

[10] P.A. Bernstein, T.J. Green, S. Melnik, and A. Nash, "Implementing Mapping Composition," *Proc. 32nd Int'l Conf. Very Large Data Base (VLDB '06),* pp. 55-66, 2006.

[11] P.A. Bernstein, A.Y. Halevy, and R. Pottinger, "A Vision of Management of Complex Models," *SIGMOD Record,* vol. 29, no. 4, 2000.

[12] P.A. Bernstein and S. Melnik, "Model Management 2.0: Manipulating Richer Mappings," *Proc. ACM SIGMOD '07,* pp. 1-12, 2007.

[13] P.A. Bernstein, S. Melnik, and J.E. Churchill, "Incremental Schema Matching," *Proc. 32nd Int'l Conf. Very Large Data Base (VLDB '06),* pp. 1167-1170, 2006.

[14] R.E. Burkard, E. Cela, P.M. Pardalos, and L.S. Pitsoulis, *The Quadratic Assignment Problem. In Handbook of Combinatorial Optimization,* vol. 2. Kluwer Academic Publishers, 1998.

[15] S. Castano, V. Antonellis, and S. Vimercati, "Global Viewing of Heterogeneous Data Sources," *IEEE Trans. Knowledge and Data Eng.,* vol. 13, no. 2, pp. 277-297, Mar./Apr. 2001.

[16] T.M. Cover and J.A. Thomas, *Elements of Information Theory.* John Wiley & Sons, 1991.

[17] R. Dhamankar, Y. Lee, A. Doan, A.Y. Halevy, and P. Domingos, "iMAP: Discovering Complex Mappings between Database Schemas," *Proc. ACM SIGMOD,* 2004.

[18] A. Doan, P. Domingos, and A.Y. Halevy, "Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach," *Proc. ACM SIGMOD,* 2001.

[19] A. Doan, P. Domingos, and A.Y. Levy, "Learning Source Description for Data Integration," *Proc. Third Int'l Workshop Web and Databases (WebDB '00),* pp. 81-86, 2000.

[20] C. Domshlak, A. Gal, and H. Roitman, "Rank Aggregation for Automatic Schema Matching," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 4, pp. 538-553, Apr. 2007.

[21] R. Fagin, P.G. Kolaitis, R.J. Miller, and L. Popa, "Data Exchange: Semantics and Query Answering," *Theoretical Computer Science,* vol. 336, no. 1, pp. 89-124, 2005.

[22] R. Fagin, P.G. Kolaitis, L. Popa, and W.C. Tan, "Composing Schema Mappings: Second-Order Dependencies to the Rescue," *ACM Trans. Database Systems,* vol. 30, no. 4, pp. 994-1055, 2005.

[23] R. Fagin, P.G. Kolaitis, L. Popa, and W.C. Tan, "Quasi-Inverses of Schema Mappings," *Proc. 26th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS '07),* pp. 123-132, 2007.

[24] N. Friedman, I. Nachman, and D. Peer, *Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate" Algorithm,* pp. 206-215, 1999.

[25] M.R. Garey and D.S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[26] L. Getoor, B. Taskar, and D. Koller, "Selectivity Estimation Using Probabilistic Models," *Proc. ACM SIGMOD,* 2001.

[27] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 4, pp. 377-388, Apr. 1996.

[28] L.M. Haas, M.A. Hernández, H. Ho, L. Popa, and M. Roth, "Clio Grows Up: From Research Prototype to Industrial Tool," *Proc. ACM SIGMOD '05,* pp. 805-810, 2005.

[29] B. He and K.C.-C. Chang, "Making Holistic Schema Matching Robust: An Ensemble Approach," *Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery in Data Mining (KDD '05),* pp. 429-438, 2005.

[30] B. He, K.C.-C. Chang, and J. Han, "Discovering Complex Matchings Across Web Query Interfaces: A Correlation Mining Approach," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '04),* pp. 148-157, 2004.
[31] K.-J.R.H. Mannila, "Dependency Inference," *Proc. 13th Int'l Conf. Very Large Data Base (VLDB '87),* pp. 155-158, 1987.
[32] M.A. Hernandez, R.J. Miller, and L.M. Haas, "Clio: A Semi-Automatic Tool for Schema Mapping," *Proc. ACM SIGMOD,* 2001.
[33] Y. Huhtala, J. Karkkainen, P. Porkka, and H. Toivonen, "Efficient Discovery of Functional and Approximate Dependencies Using Partitions," *Proc. 14th Int'l Conf. Data Eng. (ICDE),* 1998.
[34] R. Hull, "Relative Information Capacity of Simple Relational Database Schemata," *SIAM J. Computing,* vol. 15, no. 3, pp. 856-886, 1986.
[35] S. Ishii and M. aki Sato, "Doubly Constrained Network for Combinatorial Optimization," *Neurocomputing,* vol. 43, nos. 1-4, pp. 239-257, 2002.
[36] J. Kang and J.F. Naughton, "On Schema Matching with Opaque Column Names and Data Values," *Proc. ACM SIGMOD '03,* June 2003.
[37] J. Kivinen and H. Mannila, "Approximate Inference of Functional Dependencies from Relations," *Theoretical Computer Science,* vol. 149, no. 1, pp. 129-149, 1995.
[38] W.-S. Li and C. Clifton, "Semantic Integration in Heterogeneous Databases Using Neural Networks," *Proc. 20th Int'l Conf. Very Large Data Base (VLDB '94),* pp. 1-12, 1994.
[39] W.-S. Li and C. Clifton, "SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Networks," *J. Data and Knowledge Eng.,* vol. 33, no. 1, Dec. 2000.
[40] J. Madhavan, P.A. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid," *Proc. 27th Int'l Conf. Very Large Data Base (VLDB),* 2001.
[41] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching," *Proc. 18th Int'l Conf. Data Eng. (ICDE),* 2002.
[42] S. Melnik, A. Adya, and P.A. Bernstein, "Compiling Mappings to Bridge Applications and Databases," *Proc. ACM SIGMOD '07,* pp. 461-472, 2007.
[43] R.J. Miller, L.M. Haas, and M.A. Hernandez, "Schema Mapping as Query Discovery," *Proc. 26th Int'l Conf. Very Large Data Base (VLDB '00),* pp. 77-88, 2000.
[44] R.J. Miller, Y.E. Ioannidis, and R. Ramakrishnan, "The Use of Information Capacity in Schema Integration and Translation," *Proc. 19th Int'l Conf. Very Large Data Base (VLDB '93),* pp. 120-133, 1993.
[45] R.J. Miller, Y.E. Ioannidis, and R. Ramakrishnan, "Schema Equivalence in Heterogeneous Systems: Bridging Theory and Practice," *Proc. Fourth Int'l Conf. Extending Database Technology (EDBT),* 1994.
[46] T. Milo and S. Zohar, "Using Schema Matching to Simplify Heterogeneous Data Translation," *Proc. 24th Int'l Conf. Very Large Data Base (VLDB),* 1998.
[47] E. Rahm and P.A. Bernstein, "On Matching Schemas Automatically," *The VLDB J.,* vol. 10, no. 4, Dec. 2001.
[48] E. Rahm and P.A. Bernstein, "A Survey of Approaches to Automatic Schema Matching," *The VLDB J.,* vol. 10, no. 4, 2001.
[49] J. Rissanen, "On Equivalences of Database Schemes," *Proc. First ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS '82),* pp. 23-26, 1982.
[50] K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems," *Proc. IEEE,* vol. 86, pp. 2210-2239, 1998.
[51] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.
[52] C. Schellewald, S. Roth, and C. Schnorr, "Evaluation of Convex Optimization Techniques for the Weighted Graph-Matching Problem in Computer Vision," *Proc. 23rd DAGM Symp. Pattern Recognition (DAGM '01),* pp. 361-368, 2001.
[53] P. Shvaiko and J. Euzenat, *A Survey of Schema-Based Matching Approaches,* pp. 146-171, 2005.
[54] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 5, pp. 695-703, Sept. 1988.
[55] S.J. Wright, *Primal-Dual Interior-Point Methods,* SIAM, 1997.
[56] L.-L. Yan, R.J. Miller, L.M. Haas, and R. Fagin, "Data-Driven Understanding and Refinement of Schema Mappings," *Proc. ACM SIGMOD,* 2000.

Jaewoo Kang received the bachelor's degree in computer science from Korea University, Seoul in 1994, the MS degree in computer science from the University of Colorado at Boulder in 1996, and the PhD degree in computer science from the University of Wisconsin-Madison in 2003. He served as a faculty member in the Department of Computer Science, North Carolina State University before moving to Korea University, where he is currently an assistant professor of computer science in the College of Information and Communication. His research interests in a broad sense include understanding the fundamental aspects of building a large-scale information system that can answer complex queries over a large number of heterogeneous data sources. He focuses on tackling the challenge particularly in data integration, query optimization, semistructured data management, Web mining, and biomedical informatics.

Jeffrey F. Naughton received the bachelor's degree in mathematics from the University of Wisconsin-Madison and the PhD degree in computer science from Stanford University. He served as a faculty member in the Department of Computer Science, Princeton University before moving to the University of Wisconsin-Madison, where he is currently a professor of computer science. His research interests have focused on improving the performance and functionality of database management systems. He has published more than 100 technical papers and received the US National Science Foundation's Presidential Young Investigator award in 1991 and was named an ACM fellow in 2002.