μBE: User Guided Source Selection and Schema Mediation for Internet Scale Data Integration

Ashraf Aboulnaga Kareem El Gebaly

University of Waterloo
$\{$*ashraf, kelgebal*$\}$*@cs.uwaterloo.ca*

Abstract

The typical approach to data integration is to start by defining a common mediated schema, and then to map the data sources being integrated to this schema. In Internet-scale data integration tasks, where there may be hundreds or thousands of data sources providing data of relevance to a particular domain, a better approach is to allow the user to discover the mediated schema and the set of sources to use through an iterative exploration of the space of possible schemas and sources. In this paper, we present μBE, a data integration tool that helps in this iterative exploratory process by automatically choosing the data sources to include in a data integration system and defining a mediated schema on these sources. The data integration system desired by the user may depend on several subjective and objective criteria, and the user guides μBE towards finding this system by iteratively solving a series of constrained non-linear optimization problems, and modifying the parameters and constraints of the problem in the next iteration based on the solution found in the current iteration. Our formulation of the optimization problem is designed to make it easy for the user to provide such feedback. A simple, intuitive user interface helps the user in this process. We experimentally demonstrate that μBE is efficient and finds high-quality data integration solutions.

1. Introduction

The typical approach to data integration is to start by defining a common mediated schema for all the data sources, then to match and map the data sources to this mediated schema. This assumes that the user (1) knows all the sources that should be included in the data integration system, and (2) knows all the concepts that are expressed in these data sources (which will come together to form the mediated schema). These assumptions do not necessarily hold in cases where the data integration task involves hundreds or thousands of data sources, especially if these data sources are new to the user. Such "Internet-scale" data integration tasks are becoming increasingly more common [10, 12]. A prominent example of these tasks is integrating data from multiple hidden Web data sources [4]. These data integration tasks can also arise in other situations such as peer-to-peer networks supporting structured queries, personal information management, scientific data management [10, 12], and ad-hoc data integration in "mashups" of enterprise data sources.

In these Internet-scale data integration tasks, unlike in traditional data integration tasks, we cannot assume that there is a *data architect* with deep domain expertise and a global view of the different concepts expressed at the different data sources. The user in Internet-scale data integration tasks most likely has an understanding of the important concepts in the domain they are working in, but may not know all the different concepts expressed in all the available data sources, and how they are expressed in these sources. Thus, it is difficult for the user to decide a priori on the mediated schema, since this requires a thorough exploration of all available data sources to identify the concepts to include in this schema. Furthermore, the user may not want to include all available data sources in the *data integration system* being defined, especially if there is significant overlap in the data in the different sources. The main reason for this is that if a data source expresses the concepts it contains in a way that is different from other data sources, then including this source will reduce the semantic coherence of the global mediated schema and hence reduce the overall quality of the data integration system. Moreover, there are networking and processing costs associated with including a data source in the data integration system. These are the costs to retrieve data from the source while executing queries, map this data to the global mediated schema, and resolve any inconsistencies with data retrieved from other sources. The more sources we have, the higher these costs become.

Thus, we can see that a user who is building an Internet-scale data integration system needs to solve two problems: (1) which sources to include in the data integration system, and (2) what mediated schema to use. These two problems are highly interrelated. On the one hand, the selected data sources determine the concepts that can be part of the mediated schema. On the other hand, the sources that we select must include the important concepts that we want to make part of the mediated schema.

As a motivating example, consider a user who wants to integrate data from multiple hidden Web data sources

tonyawards.com: {keywords}
whatsonstage.com: {your town}
aceticket.com: {state, city, event, venue}
canadiantheatre.com: {phrase, search term}
londontheatre.co.uk: {type,keyword}
mime.info.com: {search for}
pbs.org: {program title, date, author, actor, director, keyword}
pa.msu.edu: {keyword}
wstonline.org: {keyword, after date, before date}
officiallondontheatre.co.uk: {keyword, after date, before date}
lastminute.com: {event name, event type, location, date, radius}

Figure 1. Schemas from CompletePlanet.com.

that deal with reserving theater tickets. One way to get a list of sources that deal with this domain is to issue the query "theater" in a hidden Web search engine such as CompletePlanet.com. At the time of this writing, this query produces 1021 results. Figure 1 shows a sample of the resulting sources and their schemas. Recent work on understanding hidden Web query interfaces can help the user extract these schemas [5, 19]. But the user must still decide on the sources to get data from and the mediated schema to use. It is plausible to assume that the user does not want to integrate data from all 1021 sources, since many of them provide information that may not be of interest to the user (e.g., ticket sales in a different country). Furthermore, including all these sources will unnecessarily increase the cost of executing queries, especially if the same information is repeated in multiple sources. As for choosing a mediated schema, Figure 1 illustrates that the data sources contain many different concepts expressed in many different ways. The user must decide which of these concepts will become attributes of the global mediated schema.

To decide on the data sources to include in a data integration system and the mediated schema to use, it is helpful if the user can *iteratively explore* the space of possible sources and mediated schemas to find a good solution. In exploring the space of possible solutions, the user is gaining a better understanding of the problem domain and the characteristics of the desired solution. The chosen solution may depend on some quantitative measures of quality, but it will likely depend as well on the subjective preferences of the user (e.g., people typically have a preferred site for purchasing books or tickets). In this paper, we address the problem of helping users in this difficult, but increasingly common [12], exploratory process.

We present our techniques in the context of a data integration tool that we call *μBE*, for *Matching By Example*[1]. Figure 2 presents the architecture of μBE. μBE takes as input the descriptions of a large number of data sources, their

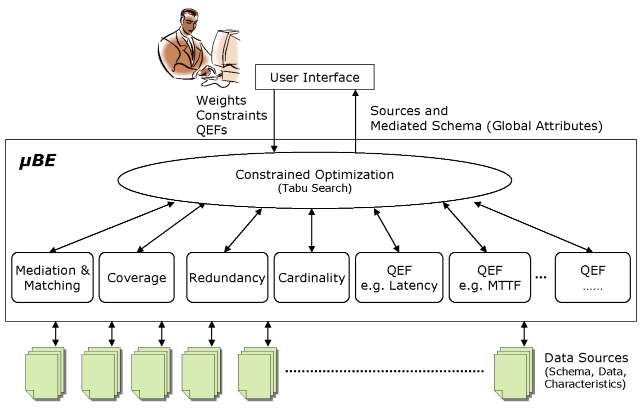


Figure 2. Architecture of μBE.

schemas, their data characteristics, and other source characteristics of interest to the user. Such descriptions can be obtained from a hidden Web search engine or some other source discovery mechanism, or they can be provided by the user. μBE chooses a *solution* to the data integration problem, consisting of a set of data sources to use and a mediated schema on these sources. The choice of solution depends on several quality metrics that capture (1) how well the schemas of the sources match with each other and with any constraints on the global mediated schema specified by the user, (2) the characteristics (quality and quantity) of the data at the sources, and (3) other source characteristics such as latency, reliability, or fees charged. The task of finding the best data integration solution is formulated as a non-linear constrained optimization problem, and μBE solves this problem using a combinatorial optimization technique. To enable an iterative exploratory process for finding a data integration solution that satisfies the user, μBE allows the user to provide feedback on the solution that it chooses. The user can specify new constraints on sources and mediated schema attributes to include, set new weights for the quality metrics, and define new quality metrics. μBE solves this new optimization problem, and the iterative feedback process continues with the user guiding μBE until he or she is satisfied with the solution that it finds. This iterative, best-effort approach is very well-suited to the kinds of Internet-scale data integration tasks that we are targeting.

The rest of this paper is organized as follows. In Section 2, we present a novel formulation of the source selection and schema mediation problem. Our formulation builds on prior work in the area of schema matching, but also considers a broader range of source characteristics and facilitates user feedback. In Section 3, we present a clustering algorithm that generates mediated schemas satisfying user constraints. Section 4 defines metrics for evaluating the quality of a data integration system based on the characteristics of the data at the sources included, and provides a technique for efficiently estimating these metrics. Sec-

[1]We chose this name because our approach bears some similarity to the query language *QBE*.

tion 5 describes techniques for handling other source characteristics. Section 6 describes the solution to the resulting constrained optimization problem and our model for interacting with users. Our experimental results are reported in Section 7. We present an overview of related work in Section 8 and conclude in Section 9.

2. Problem Definition

To define a data integration system, we must identify a set of data sources, a global mediated schema over these sources, and a mapping from the sources to the mediated schema. In μBE, we choose the data integration system by solving a non-linear constrained optimization problem, where the problem parameters and constraints are provided by the user. In this section, we define the necessary concepts and formulate the problem.

2.1. Data Sources and Schemas

μBE does not place any restrictions on the kinds of schemas and data sources that it can handle. Thus, μBE can be used with relational data sources, XML data sources, or any other kind of data sources. The only requirement is that there be a *schema matching* operator that enumerates pairs of schema elements at any given two sources, and computes a measure of *similarity* (or *quality of matching*) between each pair of schema elements. This enumeration of schema elements is used for our attribute clustering algorithm (Section 3).

In this paper, we focus on data sources that have a *relational* schema and on 1:1 matching. Thus, the schema of source i consists of a list of attributes $(a_{i1}, a_{i2}, \ldots, a_{in_i})$. For hidden Web data sources, such a schema can be automatically extracted [5, 19]. Nevertheless, we note that our formulation may be extended to accommodate compound schema elements by replacing the attributes in our definitions with compound elements (e.g., elements consisting of sets of attributes). This would enable us to handle matching with *n*:*m* cardinality by mapping *n*:*m* matches to 1:1 matches on compound elements.

The decision of which sources to include in the data integration system is based not only on the schema of the data sources, but also on the characteristics of the data at these sources, which we assume is a set of tuples. We do not require sources to export their entire data set, but certain metrics used by μBE to evaluate the quality of a solution do require the data sources to cooperate by providing the cardinality of their data (number of tuples) and a hash signature of this data. If we cannot obtain cooperation from the data sources, μBE can still find a data integration solution, but it will not be able to use the quality metrics that require source cooperation.

A novel feature of our work is that we also consider source characteristics that reflect non-functional properties of the sources that the user cares about, such as latency, availability, fees, or reputation. We simply call these the *source characteristics*. Some of these characteristics can be measured automatically by μBE, such as latency, but some must be provided by the source, such as fees charged.

Thus, a data source, s_i, from the point of view of μBE consists of a schema, a set of tuples, and a set of characteristics. Our universe, $\mathbb{U} = \{s_1, s_2, \ldots, s_N\}$, is the set of all data sources from which we will choose a solution. Our assumption is that the kinds of problems that μBE will be useful for will have hundreds to a few thousands of sources.

2.2. Mediated Schemas

The mediated schema in μBE consists of a set of attributes. We refer to an attribute in the mediated schema as a *Global Attribute* (GA). Data integration work traditionally assumes that the mediated schema consists of *named attributes* given in some data definition language [16, 18]. In our work, we attempt to automatically discover the mediated schema and the mapping from the data sources to this schema. This *automatic mediation* process allows us to find the GAs of the global mediated schema, but it does not result in named GAs, and we do not attempt to impose names on the generated GAs. Instead, we define a GA as a *set of attributes from different schemas of different data sources that match with each other*. All these attributes will map to the same mediated schema attribute, and there is no need to name this mediated schema attribute.

Since a GA expresses a matching between attributes from different sources, a *valid* GA cannot contain two attributes from the same source. Another way to look at this is that the same concept cannot be expressed by two different attributes from the same source.

Definition 1 *A GA, g, is a set of attributes, $\{a_{ij}\}$, from some sources in our universe, $\mathbb{U}$. g is valid iff $g \neq \phi$ and*

$$\forall a_{i_1 j_1}, a_{i_2 j_2} \in g \; (i_1 = i_2 \;\Rightarrow\; j_1 = j_2)$$

In μBE, we are looking for *valid* mediated schemas, which are mediated schemas in which the GAs do not intersect (i.e., an attribute cannot appear in two GAs). This is because the GAs represent different concepts, and a single attribute cannot represent two different concepts. A valid mediated schema must also span all the sources on which it is defined.

Definition 2 *A mediated schema, M, is a set of GAs, $\{g_l\}$. M is valid on a set of sources, S, iff*

$$\forall g_{l_1}, g_{l_2} \in M \; (l_1 \neq l_2 \;\Rightarrow\; g_{l_1} \cap g_{l_2} = \phi)$$

$$\forall s \in S \; (\exists g_l \in M \; (g_l \cap s \neq \phi))$$

It is useful to define the notion of *subsumption* of mediated schemas. We say that mediated schema M_1 subsumes M_2 if every GA in M_2 is contained in some GA in M_1.

Definition 3 *Mediated schema M_1 is said to* subsume *mediated schema M_2, denoted by $M_2 \preceq M_1$, iff*

$$\forall g_2 \in M_2 \ (\exists g_1 \in M_1 \ (g_2 \subseteq g_1))$$

By adopting this definition of GAs, we can allow the user to specify constraints requiring certain GAs to be present in the chosen mediated schema. This set of GAs provided by the user can be viewed as a *partial mediated schema*. The output of $\boldsymbol{\mu}$BE is also a set of GAs representing a mediated schema. Using GAs as the format for input constraints and for the output mediated schema makes it simple for the user to take the output of one iteration of $\boldsymbol{\mu}$BE and modify it (thus providing feedback) to get the input constraints for the next iteration.

2.3. Quality Evaluation Functions

We measure quality on multiple dimensions that depend on schema matching, data, and source characteristics. To evaluate quality on each of these dimensions, we define a *quality evaluation function* (QEF), $F_k(S)$, for each quality dimension, k. The QEF $F_k(S)$ takes a set of data sources, S, and returns a number in the range $[0-1]$ representing a measure of aggregate quality for S. The higher the value of $F_k(S)$, the better the quality of S on dimension k.

There are four main QEFs, $F_1, \ldots, F_4$. The first is the *matching quality*, F_1, which represents how well the schemas of the sources match with each other (Section 3). The other three are functions that depend on the characteristics of the data at the sources (Section 4). The user can also define other QEFs that evaluate other source characteristics (Section 5). We denote the set of all QEFs by $\mathbb{F}$.

We define the *overall quality*, $Q(S)$, of a set of sources, S, as the *weighted sum* of the QEFs. The weights, $W = \{w_1, w_2, \ldots, w_{|\mathbb{F}|}\}$, are all between 0 and 1, and they sum to 1. These weights reflect the relative importance to the user of the different quality dimensions. The weights are set by the user, and they can be changed between iterations of $\boldsymbol{\mu}$BE to guide the search for a solution towards different parts of the search space. Thus, $Q(S)$ is defined as: $Q(S) = \sum_{i=1}^{|\mathbb{F}|} w_i F_i(S)$, subject to: $0 \leq w_i \leq 1$ for all w_i, and $\sum_{i=1}^{|\mathbb{F}|} w_i = 1$.

2.4. Constraints

One of the ways that users guide $\boldsymbol{\mu}$BE is by specifying *constraints* on the optimization problem. These constraints can model user preferences, and can also guide the schema matching process, increasing its accuracy and efficiency. We allow users to specify two types of constraints: *source constraints* and *GA constraints*.

A source constraint specifies a particular source from $\mathbb{U}$ and requires it to be part of the solution chosen by $\boldsymbol{\mu}$BE. The set of source constraints, $C \subseteq \mathbb{U}$, does not specify how to match the sources in the constraints with each other or with other sources. A GA constraint is a valid GA, g, that the user requires to be part of the solution (i.e., the output must have a GA that contains g). We denote the set of GA constraints specified by the user by G. The GA constraints require the output mediated schema, M, to satisfy $G \preceq M$. Note that a GA constraint implicitly specifies a set of source constraints. If a GA constraint has an attribute a_{ij}, the source, s_i, from which this attribute is taken must be present in the solution.

The GA constraints specify a partial mediated schema. These constraints can be used to guide schema matching so that the chosen solution satisfies user preferences and a priori knowledge. For example, there may be insufficient evidence to support matching two attributes, but the user might *know* that these two attributes must match. In this case, the user can specify a GA constraint containing these two attributes, and the matching algorithm would grow this GA and take advantage of the "bridging effect" [19].

2.5. Optimization Problem

We can now formulate the constrained optimization problem solved by $\boldsymbol{\mu}$BE: Given $\mathbb{U}, \mathbb{F}, W, C, G$, find:

$$\arg\max_{S \subseteq \mathbb{U}} \ (Q(S)) = \sum_{i=1}^{|\mathbb{F}|} w_i F_i(S)$$

subject to: $|S| \leq m$, $\quad C \subseteq S$, $\quad G \preceq M$, $\quad \forall\ g \in (M-G)(F_1(\{g\}) \geq \theta)$, and $\quad \forall\ g \in (M-G)(|g| \geq \beta)$. $\mathbb{U}$ is the universe, $\mathbb{F}$ is the set of QEFs, W is the set of weights, C is a set of source constraints, G is a set of GA constraints, $Q(S)$ is the overall quality, m is the maximum number of sources that the user is willing to select, M is the mediated schema associated with S, θ is a lower bound on the matching quality, and β is a lower bound on the number of attributes in any output GA. The last two constraints (θ and β) only apply to GAs in $M-G$. For the GAs that the user specifies in G, there are no restrictions on match quality or size.

This is a highly non-linear constrained optimization problem that is repeatedly solved by $\boldsymbol{\mu}$BE with different parameters specified by the user. In Section 6, we describe our approach to solving this problem and to interacting with the user. But first, in the next three sections, we provide more details about computing the QEFs.

3. Matching by Constrained Clustering

An important QEF is the *matching quality* QEF, $F_1(S)$. $F_1(S)$ relies on a *schema matching operator* [18] that we define, $Match(S)$. $Match(S)$ determines the best matching between the schemas of the data sources in S, and returns this matching along with a measure of its quality to be used as the value of the QEF. $Match(S)$ produces a mediated schema, M, that consists of a set of GAs representing the best matching among the schemas of the data sources in S.

This mediated schema is the automatically generated mediated schema that **μ**BE presents to the user. It must satisfy the source constraints in C and the GA constraints in G (i.e., M must be valid on C, and $G \preceq M$ must hold). To simplify satisfying source constraints, we ensure for any call to $Match(S)$ that S contains C. As we can see, $Match(S)$ is at the core of **μ**BE, automatically generating the mediated schemas and measuring their quality.

To find the best mediated schema, M, $Match(S)$ requires a measure of similarity (i.e., quality of matching) between *pairs* of attributes in different data sources. $Match(S)$ can use any attribute similarity measure, whether it is schema based [18] or data based [14]. In our prototype of **μ**BE, our measure of similarity between a pair of attributes is the Jaccard similarity coefficient [6] between the 3-grams in the attribute names.

Regardless of the similarity measure that we use, the question we must answer is how to build a set of GAs that span multiple sources using a pairwise attribute similarity measure. To do this, we use a *greedy constrained similarity clustering* algorithm (Algorithm 1). The algorithm starts by placing each attribute from each data source in S in its own cluster. If there are GA constraints provided by the user, we place the GA from each constraint in its own cluster. We then repeat the iterative step of the clustering algorithm, which consists of growing the clusters by greedily merging *all pairs* of clusters whose similarity is above the matching threshold specified in our optimization problem, θ, as long as the merged cluster represents a valid GA. We define the similarity between two clusters as the *maximum similarity between an attribute from the first cluster and an attribute from the second cluster*. Based on this definition, the algorithm eliminates clusters from consideration if their highest similarity to any other cluster is lower than the matching threshold, since these clusters will never be merged with other clusters. In practice, this elimination significantly prunes the number of clusters considered by the algorithm. The algorithm terminates when it cannot find any more pairs of clusters to merge.

The clusters of attributes that are found by the algorithm together form the generated mediated schema, M, with every cluster of attributes representing a GA in M. The algorithm checks that M is valid on C (i.e., the source constraints are satisfied), and if so it returns M and its quality of matching. If not, there is no matching that satisfies both the matching threshold and source constraints for this set of sources, so we return a null schema and 0 matching quality.

We define the *quality of matching within a cluster* as the maximum similarity between any two attributes in this cluster. Since, by construction, the algorithm ensures that every attribute in a cluster has similarity $\geq \theta$ to at least one other attribute in the cluster, the quality of matching in the GAs returned by the algorithm is always at least θ. We define the quality of matching of the whole mediated schema, M, as the *average quality of matching for all the GAs of this schema*. This is the the value of the QEF, F_1, returned by the algorithm. It is clear that this value will, by construction, always be $\geq \theta$ for each GA individually (except for GAs that come from GA constraints, $g \in G$, which may have $F_1(\{g\}) < \theta$). Thus, the matching threshold is always satisfied.

GA constraints, along with our definition of cluster similarity as the maximum similarity between attributes from the two clusters, are very helpful in allowing users to guide the matching process. If there is not enough evidence to support placing attributes a and b in the same cluster, but the user *knows* that these attributes express the same concept, the user can introduce a GA constraint containing a and b. The clustering algorithm will keep this GA even though its matching quality is low. Moreover, the cluster can continue to grow even though it contains two dissimilar attributes: attributes that are similar to a will be added to the cluster because of their high similarity to a, and will not be penalized by the presence of b. The same will happen for attributes that are similar to b. This style of matching is why we call our tool *Matching By Example*. The user provides an example of a matching, and **μ**BE expands it.

Figure 3 illustrates the clustering algorithm. The example in the figure focuses on four attributes that are part of a larger clustering problem consisting of many attributes. Figures 3(a)–(c) show the case where there are no GA constraints. In this case, the algorithm starts by placing each attribute in its own cluster (Figure 3(a)). The two clusters with the maximum similarity are then merged, and the similarity between the new clusters and existing clusters is computed (Figure 3(b)). The process continues and we reach the clustering in Figure 3(c). Figures 3(d)–(f) show the case where the user specifies a GA constraint: `F_name` and `Prenom` must be in the same cluster. This constraint bridges the semantic gap between these two attributes. In this case, clustering starts by putting these two attributes in one cluster, and each other attribute in a cluster of its own. Clustering proceeds as before, but the GA constraint allows us to capture relationships that we could not capture before.

4. Coverage and Redundancy

We now turn our attention to the QEFs that depend on the data at the sources. These QEFs, F_2, F_3, and F_4, are cardinality, coverage, and redundancy, respectively. We also refer to them as $Card(S)$, $Coverage(S)$, and $Redundancy(S)$. These functions return a number between 0 and 1, and are defined by the following equations: $Card(S) = \frac{\sum_{s \in S}|s|}{\sum_{t \in \mathbb{U}}|t|}$, $Coverage(S) = \frac{|\bigcup_{s \in S} s|}{|\bigcup_{t \in \mathbb{U}} t|}$, and $Redundancy(S) = \frac{|S| - \frac{\sum_{s \in S}|s|}{|\bigcup_{t \in S} t|}}{|S|-1}$. $|s|$ is the total number of tuples (cardinality) at a source or set of sources.

Algorithm 1 Greedy constrained similarity clustering.

```
$Match$ ($S$ : set of sources, $C$ : set of source constraints, $G$ : set of GA constraints)
1  $M \leftarrow \{\}$          ▷ Initialize set of clusters (GAs) to return
2  Add all GAs in $G$ to $M$
3  Set a flag $c_i.keep$ = TRUE for all clusters in $M$
4  Add all remaining attributes in all sources in $S$ to $M$ as single-attribute clusters with $c_i.keep$ = FALSE
5  **repeat**
6      $done \leftarrow$ TRUE
7      Set two flags $c_i.merged$ = FALSE and $c_i.mergeCand$ = FALSE for all clusters in $M$
8      Find pairs of clusters $(c_i, c_j)$ that have similarity $\geq \theta$, and store them in a priority queue, $H_{sim}$, sorted by similarity
9      **while** $\neg(H_{sim}$ is empty$)$ **do**
10         ▷ Get the pair of clusters with the next highest similarity
11         $(c_1, c_2, sim) \leftarrow H_{sim}.pop()$
12         **if** $\neg c_1.merged \wedge \neg c_2.merged \wedge$ Merging the two clusters gives a valid GA **then**
13             Merge $c_1$ and $c_2$. Remove them from $M$ and add the merged cluster.
14             $c_1.merged \leftarrow$ TRUE, $c_2.merged \leftarrow$ TRUE
15         **elseif** $c_i.merged$ = TRUE for only one of $c_1$ and $c_2$ **then**
16             ▷ We found a cluster that could be merged, but the cluster it could be merged with has already been merged with another cluster.
17             ▷ We need another iteration, and we need to keep the cluster that we could not merge this iteration for the next iteration.
18             Set $c_i.mergeCand$ = TRUE for the cluster that has $c_i.merged$ = FALSE
19             $done \leftarrow$ FALSE
20     **for** all clusters $c_i \in M$ **do**
21         **if** $\neg$(newly merged cluster) $\wedge \neg c_i.mergeCand \wedge \neg c_i.keep$ **then**
22             Eliminate $c_i$ from $M$
23   **until** $done$
24 **return** $M$ and its quality of match if $M$ is valid on $C$, and return NULL otherwise
```

The $Card(S)$ QEF measures the amount of data in S. The $Coverage(S)$ QEF measures how much of the total data in the universe, $\mathbb{U}$, we are able to get from S. The $Redundancy(S)$ QEF is a measure of the degree of overlap between the data in the sources in S. A high degree of overlap is bad because we unnecessarily get the same data from many sources. The QEF is defined so that $Redundancy(S) = 0$ is the worst possible and $Redundancy(S) = 1$ is the best possible, as required by our optimization problem.

To compute these QEFs based on the data, we need to be able to compute the cardinalities of data sources, and the cardinalities of unions of data sources. The cardinalities of sources can be obtained directly from the sources. The difficulty is in computing the cardinalities of unions of sources, because (1) many sources do not allow unrestricted access to their data, and (2) even if the sources did allow access to the data, the sheer amount of data at the sources makes a solution that requires fetching all the data from the sources prohibitively expensive. Thus, we need a way to estimate the cardinality of set unions without accessing the data.

We develop a technique based on the well-known Flajolet and Martin *Probabilistic Counting with Stochastic Averaging* (PCSA) technique [9]. To count the number of distinct tuples in a large number of tuples, the basic PCSA technique uses a bit map as a synopsis for summarizing the data. A hash function is applied to all tuples, and the bit in the bitmap corresponding to the hash value is set. This hash signature is used to estimate the number of distinct tuples. To increase the accuracy of the estimation, PCSA uses multiple hash functions and multiple bitmaps.

We make the observation that if each data source computes the PCSA hash signature for the tuples it contains, and then we perform a bitwise OR of these hash signatures, that would be equivalent to computing the hash signature on the set union of all the tuples in all the data sources. Thus, we require all data sources to compute a hash signature for their tuples based on a set of pre-determined hash functions. These hash signatures are cached by μBE. To compute the cardinality of the union of some sources, we compute a bitwise OR of their hash signatures and apply the PCSA algorithm on the result. In our experiments, we have found this to provide accurate cardinality estimates.

This approach assumes that the data sources are willing to cooperate with μBE by computing hash signatures. We believe that this is a realistic assumption, since we are not asking the data sources for much. Computing the hash signature requires scanning the data only once and applying a few hash functions to each tuple. The hash signatures themselves are small (a few bytes or kilobytes), and they do not disclose any information about the actual values of the tuples. Furthermore, it is safe to assume that data sources *want* to be included in data integration systems and hence will cooperate with a tool like μBE. Nevertheless, if some data sources are not willing to provide hash signatures, we can still use μBE on *all* data sources, excluding the uncooperative sources from the coverage and redundancy computations, and assigning them 0 coverage and redundancy QEFs. The uncooperative sources may still be chosen by μBE if they score highly on other QEFs.

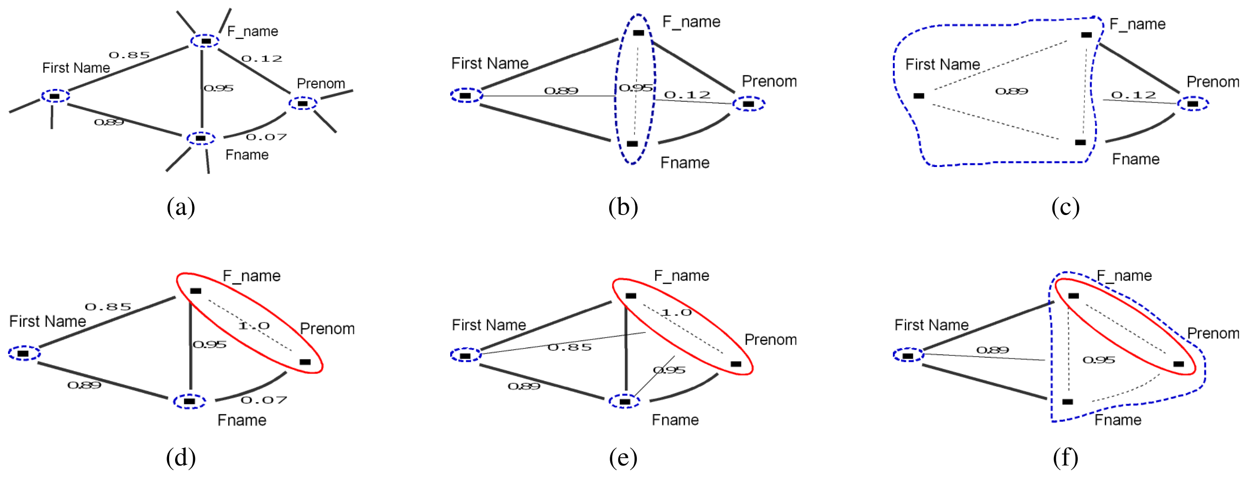
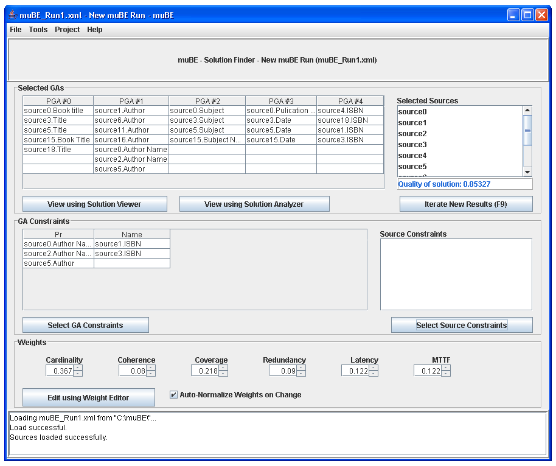


Figure 3. Constrained similarity clustering.


muBE_Run1.xml - New muBE Run - muBE
File Tools Project Help
muBE - Solution Finder - New muBE Run (muBE_Run1.xml)
Selected GAs
Selected Sources
View using Solution Viewer
View using Solution Analyzer
Iterate New Results (F9)
GA Constraints
Source Constraints
Select GA Constraints
Select Source Constraints
Weights
Edit using Weight Editor

PGA #0	PGA #1	PGA #2	PGA #3	PGA #4
source0.Book title	source1.Author	source0.Subject	source0.Pulication ...	source4.ISBN
source3.Title	source6.Author	source3.Subject	source3.Date	source18.ISBN
source5.Title	source11.Author	source5.Subject	source5.Date	source1.ISBN
source15.Book Title	source16.Author	source15.Subject N...	source15.Date	source3.ISBN
source18.Title	source0.Author Name			
	source2.Author Name			
	source5.Author			

source0
source1
source2
source3
source4
source5

Quality of solution: 0.85327

Pr	Name
source0.Author Na...	source1.ISBN
source2.Author Na...	source3.ISBN
source5.Author	

Cardinality	Coherence	Coverage	Redundancy	Latency	MTTF
0.367	0.08	0.218	0.09	0.122	0.122

[✓] Auto-Normalize Weights on Change

```
Loading muBE_Run1.xml from "C:\muBE\"...
Load successful.
Sources loaded successfully.
```

Figure 4. μBE user interface.

5. QEFs for Source Characteristics

μBE allows users to define QEFs that are based on other source characteristics such as latency, availability, fees charged, or reputation. These characteristics are defined on a per-source basis. Some of them can be automatically measured by μBE, while others must be provided by the user. To increase the flexibility in defining source characteristics, we allow the values of source characteristics to be positive real numbers of any magnitude. Thus, we need to define QEFs that aggregate a set of real numbers representing source characteristics of a set of sources into a measure in the range $[0 - 1]$. μBE provides several predefined aggregation functions, and users can define their own aggregation functions. As an example, we describe the *weighted sum* aggregation function, $wsum(S)$, which is the sum of the source characteristics for all the sources in S weighted (multiplied) by their cardinality, and normalized. The definition of $wsum(S)$ for source characteristic q_i is as follows:

$$\frac{\sum_{s \in S} \left(s.q_i - min_{\forall t \in \mathbb{U}}(t.q_i)\right) \times |s|}{\left(\sum_{s \in S} |s|\right) \left(max_{\forall t \in \mathbb{U}}(t.q_i) - min_{\forall t \in \mathbb{U}}(t.q_i)\right)}$$

This function can be used, for example, to aggregate availability. If a source has high availability and a large number of tuples, it is more valuable than a source with high availability but only a few tuples.

6. Combinatorial Search and User Interaction

Next, we turn our attention to how μBE solves the series of optimization problems, and how it helps the user to define these problems. The optimization problems being solved are non-linear constrained optimization problems. To solve these problems, we tried using *stochastic local search*, *particle swarm optimization*, *constrained simulated annealing*, and *tabu search*, and we found that tabu search [11] gives the best results. Tabu search is a combinatorial optimization algorithm whose key feature is that it partially remembers its path through the search space and uses this memory to declare parts of the search space as "tabu" for some time. The algorithm never visits the tabu parts of the space, thereby bounding the search time while still thoroughly exploring the search space. In μBE we use the constraints to define "permanently tabu" regions of the space. We omit further details for lack of space, and we refer the reader to literature on combinatorial optimization (e.g., [11]).

μBE has a simple, intuitive user interface. The interface allows the user to specify the optimization problem, and it presents the solution to this problem in a way that makes it easy to change the problem specification for the next iteration. By design, the constraints provided by the user (the input) have the same structure and format as the mediated schema generated by μBE (the output). We leverage this and allow the user to add constraints for the next iteration by modifying the output of the current iteration. Figure 4 presents a screenshot from the μBE user interface. More details about user interaction modes can be found in the demonstration accompanying this paper [2].

7. Experiments

7.1. Experimental Setup

We generated descriptions and data for 700 synthetic data sources. For each data source we need to define (1) the schema, (2) the data, and (3) the source characteristics. The schemas of the data sources are based on the BAMM repository of schemas [1]. The schemas in BAMM are extracted from Web query interfaces in different domains. For our experiments, we use the 50 schemas from the Books domain. Our 700 schemas consist of these 50 schemas and perturbed copies of them. To generate a perturbed copy of a schema, we add attributes to the schema, remove attributes from the schema, or replace attributes from the schema with other attributes whose names we get from a list of words unrelated to the Books domain. These perturbations follow a probability distribution that allows us to retain some of the characteristics of the original schemas, while at the same time having variability in our schemas. Our goal is to see whether μBE can discover the concepts of the domain it is working in, even if there is variability in the schemas.

Each data source has a number of tuples (cardinality) ranging from 10,000 to 1,000,000 that follows a Zipf distribution . The data tuples themselves are chosen randomly from a set of 4,000,000 distinct tuples consisting of random words. Half of our tuples are labeled as "General" and half are labeled as "Specialty." Half the data sources got all their tuples from the General pool. For the other half, we chose a small number of tuples from the Specialty pool and the rest from the General pool. The idea is to model the characteristics of data in Web sources. There are general items available in all Web sources dealing with a certain domain, and there are specialty items only available in a few sources. This affects our calculation of redundancy and coverage.

Each source has a *mean time to failure* (MTTF) source characteristic. The MTTF values follow a normal distribution, with mean 100 days and standard deviation 40. We use the $wsum$ aggregation function to aggregate MTTFs.

The default weights for matching quality, cardinality, coverage, redundancy, and MTTF are 0.25, 0.25, 0.2, 0.15, and 0.15, respectively. The matching threshold is $\theta = 0.75$.

We ran our experiments on a machine with dual 3.4GHz Intel Xeon CPUs and 4.0 GB of RAM running Windows Server 2003. Our implementation of μBE is in C++. The maximum memory footprint for all of our experiments never exceeded 70MB. Most of this memory was used for the hash signatures of the data sources that we store for calculating coverage and redundancy.

We conducted an extensive set of experiments, but due to lack of space we only present a subset of our results. Our experiments showed that tabu search is more robust and generates higher quality solutions than other optimization techniques, so we use it as the default algorithm for μBE, and we only report results from tabu search in this section.

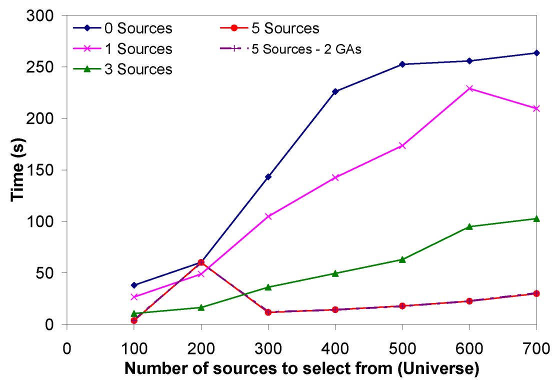


Figure 5. Varying number of sources.

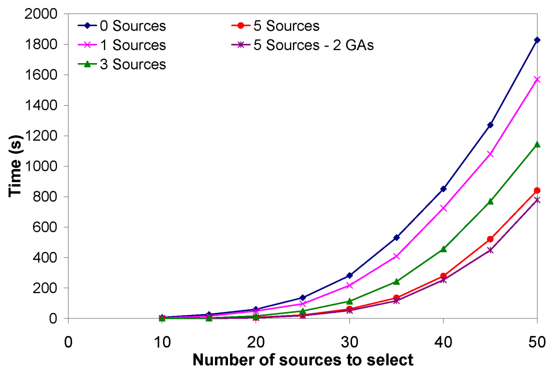


Figure 6. Varying sources to choose.

7.2. Execution Time

In this section, we study the execution time of μBE. Figure 5 shows the time for μBE to choose 20 sources from a set of data sources (a universe) whose size varies from 100 to 700 sources. Figure 6 shows the time to choose 10 to 50 sources from a universe of 200 sources. The figures show the execution times when there are no user specified constraints, when there are 1, 3, and 5 source constraints, and when there are 5 source constraints and 2 GA constraints.

The sources in the source constraints in our experiments are random sources with schemas that are fully conformant to one of the original BAMM schemas (i.e., unperturbed schemas). The GAs in our GA constraints have up to 5 attributes that represent accurate matchings of attributes that appear in different sources in the the BAMM schemas.

As expected, execution time increases with the size of the universe that has to be searched or the number of sources to choose. However, execution times are low enough to make μBE suitable for being used iteratively by the user as we envisage. Note that our vision of iterative use does not necessarily mean on-line interactive use. Data integration tasks are not done too frequently, and it is reasonable to expect sufficient time to be dedicated to them. Furthermore, the expected number of iterations will typically be small. Thus, as long as the response time of μBE is in the range

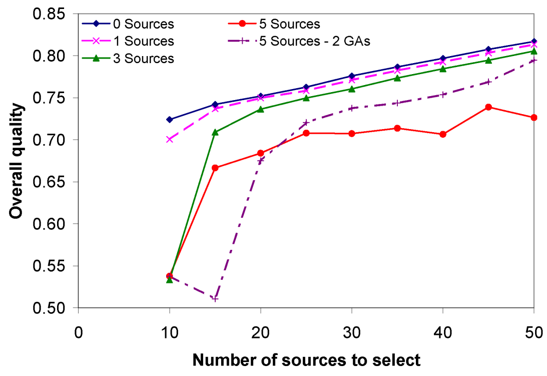


Figure 7. Overall quality.

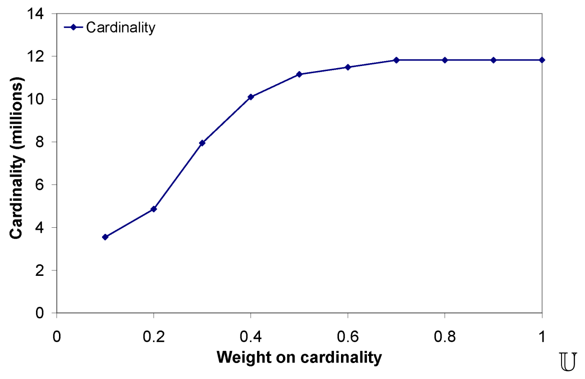


Figure 8. Sensitivity to weights.

Table 1. Quality of GAs.

Sources selected	True GAs selected	Attributes in true GAs	True GAs missed
10	4	20	4
20	8	58	2
30	9	94	2
40	11	137	1
50	14	169	0

of minutes, the interaction mode will still be convenient to the user and will allow for using μBE iteratively to solve a series of optimization problems to find the best solution.

Adding constraints reduces the execution time, since it restricts the space to be searched. This makes us even more confident in the suitability of μBE for iterative exploration, since the user will typically specify an increasing number of constraints for consecutive runs of μBE.

7.3. Solution Quality

We now turn our attention to evaluating the quality of the solutions generated by μBE. Figure 7 shows the overall quality of the solution chosen by μBE (the objective function we maximize) for the same settings shown in Figure 6. Quality increases with increasing the number of sources to choose, and decreases with increasing the number of constraints. This illustrates that tabu search is working effectively, since increasing the number of sources to choose gives it more options to exploit, and hence we expect it to get a better overall quality. Conversely, when we add constraints, we restrict the number of valid options to use, so we generally get a worse overall quality.

Overall quality is an important indicator of the effectiveness of our optimization algorithm, but what affects the user most are the sources and GAs that μBE chooses. To evaluate the quality of the sources and GAs chosen by μBE, we manually counted the number of distinct concepts in the BAMM schemas that we use. There are 14 distinct concepts in these schemas, so there can be up to 14 *true GAs* in the solution, corresponding to these concepts. Table 1 shows the number of true GAs in the solution chosen by μBE, when varying the number of sources to choose from a universe of 200 sources, with no constraints. The table also shows the number of attributes in these GAs, and the number of true GAs that were present in the sources chosen by μBE, but which μBE was not able to identify. We can see that as we allow μBE to choose more sources, it can find more of the true GAs and miss fewer of them, and it can cover more attributes with the GAs that it finds. In our experiments, μBE never produced false GAs. The number of true GAs found can be loosely interpreted as a measure of "precision" in identifying concepts, and the number of attributes covered by the GAs that are found can be interpreted as a measure of "recall" of these concepts. μBE does well on both dimensions and produces high quality, consistent solutions.

The quality of our coverage and redundancy estimates depends on the accuracy of the probabilistic counting algorithm. We have found this algorithm to be very accurate, with a worst case error of 7% compared to exact counting.

7.4. Sensitivity

Our final set of experiments studies the sensitivity of μBE to variations in the weights of the different QEFs. These weights are set by the user and might not be very precise, so we want μBE to be robust to slight variations in weights. We conducted several experiments where we randomly perturbed the values of all the weights by up to 15%, and we found that perturbing the weights caused at most 1 GA in the solution to change, and the selected sources rarely changed. Thus, we see that μBE is robust as required.

At the same time, we want changing the weights to bias the decision of μBE. To test that, we choose 20 sources from a universe of 200 sources and vary the weights on the *Card* QEF from 0.1 to 1, with the remaining weights all set to equal values. Figure 8 shows the cardinality of the chosen solution as the weight on the cardinality QEF changes. Increasing the weight allows us to bias μBE towards solutions with high cardinality, which means that weights are effective in influencing the decision of μBE. The curve flattens after a weight of 0.5 because by that time μBE is already choosing the solution that has the top cardinality sources that satisfying the matching threshold.

8. Related Work

μBE is related to work in the area of schema matching, although it does not rely on any particular schema matching technique. Some schema matching techniques can match only two schemas at a time [15, 16], while others can simultaneously match multiple schemas [3, 7, 8, 13, 14, 19]. DCM [13] and LSD [8] use multiple base learners to match multiple-schemas, but they assume the existence of a mediated schema, and they do not address the source selection problem. In [17], the problem of source selection is modeled as an optimization problem and solved using the Data Envelopment Analysis technique. The provided solution is computationally expensive so it does not scale beyond 10 to 20 sources, and the paper does not consider user interaction. Corpus-based matching [14] exploits the variety in the schemas of a corpus to learn the semantic correspondences among attributes. Like μBE, it uses a clustering approach to determine the matching, so it does not need a mediated schema. It uses a particular matching algorithm that cannot be changed, while μBE can use any matching algorithm as the basic building block of its clustering. Furthermore, μBE can also choose a set of sources to use based not only on matching quality, but also on other data and source characteristics. In [19], an interactive approach is used for determining mappings between Web query interfaces. User interaction is limited to choosing matching thresholds and accepting or rejecting mappings. μBE addresses a more general problem and has a richer user interaction model.

9. Conclusions

The traditional approach to data integration is to start by defining a mediated schema, and then to map all available data sources to this schema. In this paper, we argue that for Internet-scale data integration, a better approach is to simultaneously "discover" the best mediated schema for a particular domain and the best sources to use for this domain through an iterative exploration of the space of possible solutions. We propose μBE as a tool to help in this iterative, exploratory process. μBE chooses the best data integration solution based on the schemas, data, and characteristics of the available sources, and also on constraints and preferences specified by the user. It formulates a data integration problem as a constrained optimization problem that it solves using tabu search. Our experiments show that μBE is efficient enough to be used in the interactive mode we envisage, and that it generates high quality solutions.

References

[1] The UIUC Web integration repository. Computer Science Department, University of Illinois at Urbana-Champaign. http://metaquerier.cs.uiuc.edu/repository, 2003.

[2] A. Aboulnaga, K. El Gebaly, and D. Wong. μBE: User guided source selection and schema mediation for internet scale data integration (Demonstration). In *Proc. IEEE Int. Conf. on Data Engineering*, 2007.

[3] D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. Synthesizing an integrated ontology. *IEEE Internet Computing*, 7(5), 2003.

[4] K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the Web: Observations and implications. *SIGMOD Record*, 33(3), 2004.

[5] K. C.-C. Chang, B. He, and Z. Zhang. Toward large scale integration: Building a MetaQuerier over databases on the Web. In *Proc. Biennial Conf. on Innovative Data Systems Research (CIDR)*, 2005.

[6] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proc. IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, 2003.

[7] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2004.

[8] A. Doan, P. Domingos, and A. Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3), 2003.

[9] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2), 1985.

[10] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: A new abstraction for information management. *SIGMOD Record*, 34(4), 2005.

[11] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[12] A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 2006.

[13] B. He, K. C.-C. Chang, and J. Han. Discovering complex matchings across Web query interfaces: A correlation mining approach. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2004.

[14] J. Madhavan, P. A. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. In *Proc. IEEE Int. Conf. on Data Engineering*, 2005.

[15] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proc. Int. Conf. on Very Large Data Bases*, 2001.

[16] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proc. IEEE Int. Conf. on Data Engineering*, 2002.

[17] F. Naumann, J. C. Freytag, and M. Spiliopoulou. Quality-driven source selection using Data Envelopment Analysis. In *Proc. Int. Conf. on Information Quality (IQ)*, 1998.

[18] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4), 2001.

[19] W. Wu, C. T. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the Deep Web. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2004.