

# Using Probabilistic Information in Data Integration

Daniela Florescu  
AT&T Laboratories  
dana@research.att.com

Daphne Koller  
Stanford University  
koller@cs.stanford.edu

Alon Levy  
AT&T Laboratories  
levy@research.att.com

## Abstract

The goal of a mediator system is to provide users a uniform interface to the multitude of information sources. To translate user queries, given in a mediated schema, to queries on the data sources, mediators rely on explicit mappings between the contents of the data sources and the relations in the mediated schema.

Thus far, contents of data sources were described qualitatively. We describe the use of *quantitative* information in the form of probabilistic knowledge in mediator systems. We consider several kinds of probabilistic information: information about overlap between collections in the mediated schema, coverage and completeness of the information sources, and degrees of overlap between information sources. We address the problem of *ordering* accesses to multiple information sources, in order to maximize the likelihood of obtaining answers as early as possible. We describe a declarative formalism for specifying these kinds of probabilistic information, and we propose algorithms for ordering the information sources. Finally, we discuss a preliminary experimental evaluation of these algorithms on the domain of bibliographic sources available on the WWW.

## 1 Introduction

The rise in the number of data sources available on line has led to the development of several *mediator* systems, (also known as information integration systems), such as TSIMMIS [CGMH<sup>+</sup>94], HERMES [ACPS96], DISCO [FRV96, TRV97], SIMS [AKS96],

---

*Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.*

**Proceedings of the 23rd VLDB Conference  
Athens, Greece, 1997**

the Internet Softbot [EW94, KW96] and the Information Manifold [LRO96]. The goal of a mediator system is to provide users a uniform interface to the multitude of information sources, and therefore to free the user from having to find the information sources, interact with each one in isolation, and combine manually the information from the sources.

A user interacts with a mediator system via a *mediated schema*. A mediated schema is a set of *virtual* relations, which are not actually materialized, but are used for posing queries. The actual data is stored in the external data sources. The mediator includes a mapping between the contents of the data sources and the relations in the mediated schema as well as descriptions of the contents of the data sources. When the user poses a query, the mediator's query processor uses the content descriptions to determine which data sources are relevant to the query (i.e., may have tuples that are used in deriving solutions the query), translates the query into a set of subqueries on the data sources, and finally combines the answers appropriately.

Thus far, contents of data sources were described qualitatively (e.g., by a set of rules or view definitions). Qualitative descriptions enable the mediator to prune access only to completely irrelevant sources, which may still leave many sources to access. We would like our mediator to go one step further of relevance reasoning. The mediator should use additional information to *order* the access to the relevant data sources such that it maximizes the likelihood of obtaining answers early.

In this paper we demonstrate the utility of adding *quantitative* information in the form of probabilistic knowledge in mediator systems for the purpose of ordering access to information sources. We consider three kinds of probabilistic information. First, we consider information about overlap between collections in the mediated schema. Second, we consider information about the *coverage* of information sources. In general, most sources will provide only a *subset* of the contents in their descriptions. We would like to specify the degree to which they cover their description. Finally, a third kind of information we consider is the *over-*

lap between the sources, since some of the sources may be correlated (e.g., they may be derived from other sources by being partial copies or by integration).

We make the following contributions:

1. We provide a formalism for representing and using the three kinds of probabilistic information described above. The main challenge we address in designing the formalism is the need to specify a *complete* and *consistent* probability distribution over the set of collections, without having to consider the exponential number of combinations of collections.
2. We describe algorithms for using probabilistic information for the problem of ordering the access to data sources. In particular, our goal is to maximize the likelihood of obtaining answers as early as possible. An algorithm for producing the optimal ordering would be prohibitively expensive. Therefore, we describe two algorithms that produce approximations of the optimal ordering. The first is a greedy algorithm that only considers local interactions between sources. The second algorithm relies on precomputed orderings for a set of canonical queries, and uses those orderings for the given query.
3. We describe the implementation of the algorithms and a preliminary experimental evaluation. The experiments were performed using a collection of 100 bibliography repositories available on the WWW.

Section 2 describes the architecture of mediator systems we consider throughout the paper, and illustrates the need for probabilistic information. Section 3 describes the different kinds of probabilistic information we use and defines their meaning. Section 4 describes the algorithms for ordering information sources, and Section 5 describes our implementation and experiments. Section 6 describes several possible extensions to our framework and related work.

## 2 Mediator System Architecture

This section describes the architecture of mediator systems that we are considering and highlights the problem we address in this paper. We also introduce the example used throughout the paper and in our implementation.

### 2.1 Mediated Schema

A mediator provides users access to multiple data sources via a *mediated schema*. The mediated schema includes a set of possibly overlapping collections. Each

collection can be viewed as a unary relation whose extension is a set of objects. Properties of objects are modeled by a set of attributes which can be single or multi-valued. It should be emphasized that the collections and the attributes in the mediated schema are only *virtual* and not materialized.

Our example considers a mediator system providing access to multiple bibliography data sources available online.<sup>1</sup> Figure 1 shows a set of collections and a set of attributes we may associate with papers in our domain. The schema contains a collection CS-Paper denoting the set of publications in Computer Science. One partition of the CS-paper collection is by publication type, i.e., the collections Journal, Conference and Book, which are assumed to be pairwise disjoint. A second partition is by a topic hierarchy. Note that whereas in the first case, the collections in the partition were pairwise disjoint, these collections in the second partition are obviously overlapping.

### 2.2 Queries and Source Descriptions

We define *schema queries* to be conjunctions of atoms of the form  $c(X)$  or  $\neg c(X)$ , where  $c$  is a collection in the mediated schema. We intentionally limit the power of the query language in order to focus on the novel problems introduced by probabilistic reasoning. Possible extensions to the query language are discussed in Section 6.

A data source  $S$  supported by the mediator is described by a schema query  $Q_S$ . The meaning of a source description is that all objects in the source satisfy the conditions of  $Q_S$ . For example, the source described by the query  $\text{Database} \wedge \neg \text{Recovery} \wedge \text{Journal}$  contains only journal publications on databases, but does not contain papers about recovery. In addition,  $Q_S$  should be the most *restrictive* schema query that describes the contents of the source  $S$ . Note that the description of a source  $S$  does not necessarily mean that  $S$  contains *all* the objects satisfying  $Q_S$ .

An important point to note about the semantics of the mediated schema is that the extension of a collection is *not* necessarily the union of all the objects found in a given set of relevant data sources. For example, the collection Databases denotes the set of published papers in the field of Databases (by some agreed upon standard for what it means to be published), and does not depend on the specific sources available to the mediator at given point. This distinction is especially important when we start discussing the probability of finding a paper in a source, which should *not* change if we find a new source with papers we did not have before.

<sup>1</sup>See <http://glimpse.cs.arizona.edu/bib/> for a large collection of such sources.

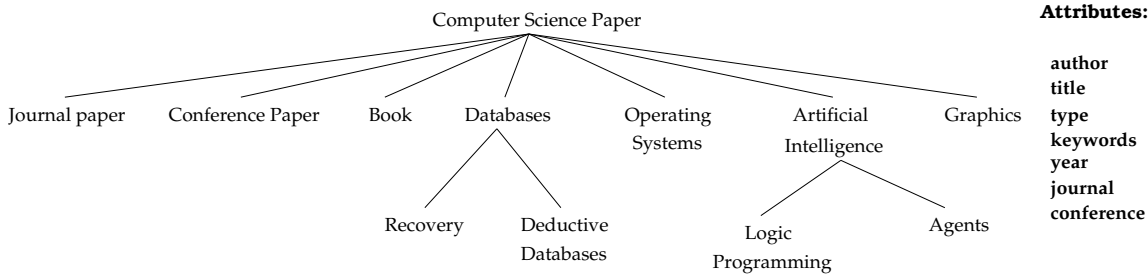


Figure 1: Mediated schema for publication domain.

### 2.3 Query Processing in the Mediator

It is important to enable users to specify arbitrary constants in their queries. Hence, a *user-query* to the mediator is a conjunction of atoms of the form  $c(X)$ ,  $\neg c(X)$  or  $X.R = a$ , where  $c$  is a collection in the mediated schema  $R$  is an attribute, and  $a$  is a constant. For example, the following query asks for Computer Science papers who have the keyword `query-languages`, and were published in TODS:

$$\text{ComputerScience}(X) \wedge X.\text{Journal} = \text{"TODS"} \wedge X.\text{keyword} = \text{"query - languages"} \wedge X.\text{year} \geq 1995.$$

As we explain shortly, our goal is to define a probabilistic model over our mediated schema, in order to derive the probability that a given source contains answers for a given query. However, user queries include *arbitrary* constant values (e.g., keywords). It is infeasible to construct a probabilistic model that considers all the sets of objects that can be defined by user queries.

Hence, we suppose that we have a method for mapping a user query to a schema query (on which we can reason), by mapping each attribute selection to a membership condition. For example, in our implementation we use the SMART information retrieval system [Buc85] to map from keywords to topics in our hierarchy. Thus, the previous user query will be translated into

$$\text{ComputerScience}(X) \wedge \text{Database}(X) \wedge \text{JournalPaper}(X).$$

Given the schema query obtained from the user query, the query processor of the mediator will determine which data sources are relevant to the schema query, and will order the relevant sources. It will then access each one. When accessing a specific data source, the mediator will send the *original* user query to the corresponding source.

Clearly, the mapping from a user query to a schema query may result in loss of information. The amount of information lost is heavily dependent of the level of detail of the mediated schema. However, the loss of information only affects the quality of the ordering of the information sources, and not the accuracy of the final answer to the query. In the rest of our discussion, the term *query* refers to schema queries.

## 3 Probabilistic Information

We begin by explaining the motivation for probabilistic information in mediator systems, and highlighting the challenges in designing our formalism. We then describe our formalism.

### 3.1 Motivation

Our goal is to improve the performance of a mediator system by obtaining answers to queries as fast as possible. In order to achieve this goal, we will try to first access the sources that have a higher probability of containing answers to the query. We now describe the kinds of probabilistic information that are needed for computing the probability that a source contains answers to a query.

The first kind of information concerns overlap between collections in the mediated schema. In current systems, we can only express three kinds of relationships between collections: (1) one collection is a superset of the other; (2) two collections are disjoint; (3) there is *some* overlap between a pair of collections. In our example, since no pair of topics in Computer Science are completely disjoint, if we asked for papers about Database Systems, the mediator would not prune *any* source containing Computer Science papers. Hence, we need a mechanism for specifying the *degree* of overlap between the collections. For example, we would like to specify that  $P(\text{AI} \mid \text{Databases}) = 0.2$ , denoting that the probability that a Database paper is also about AI is 0.2, and  $P(\text{Databases} \mid \text{Graphics}) = 0.05$ , showing that the overlap between the fields of Databases and Graphics is much smaller.

The second kind of probabilistic information we need to consider is the coverage of the information sources. Recall that we specified each source as containing a *subset* of a given schema query. However, this does not enable us to distinguish between sources that are relatively complete and those that are rather sparse. For example, we would like to specify that the probability of finding an arbitrary paper on Deductive Databases in source  $S_1$  is 0.8, which we can denote by  $P(S_1 \mid \text{DDB}) = 0.8$ .

To see how such information can be used, suppose that in addition to  $S_1$  we have a source  $S_2$ , described by  $P(S_2 | AI) = 0.1$ , i.e., that the probability of finding a paper on AI in  $S_2$  is 0.1. Suppose we are also given that the probability that a paper on Logic Programming is also a paper on Deductive Databases is 0.3 (i.e.,  $P(DDB | LP) = 0.3$ ). Consider the query asking for papers about Logic Programming. Although Logic Programming is a subtopic of AI, as we show in the next section, the probability of finding a paper on Logic Programming in  $S_2$  is only 0.1.<sup>2</sup> In contrast, the probability of finding such a paper in  $S_1$  is  $P(S_1 | DDB) \cdot P(DDB | LP) = 0.24$ . Therefore, the mediator should query  $S_1$  before  $S_2$ .

Finally, we may want to represent information on the overlap between information sources. While it is possible to assume that the overlap between sources can be automatically derived from the first two kinds of information, in some particular cases we may have more specific overlap information. For example, one source may be a view on other sources, and the query processor can take into consideration such information.

### 3.2 Challenge in Designing Probabilistic Representations

The key challenge in designing a formalism for specifying probabilistic information is that the size of the specification is exponential in the size of number of collections considered in the schema. For example, suppose we want to specify a probability distribution on the set of topics of Computer Science. That is, for every set of topics  $\mathcal{A}$  we want to know what is the probability that a paper belongs to all the topics in  $\mathcal{A}$ . To do so, we need to specify  $2^n$  numbers, where  $n$  is the number of topics. This presents two problems. First, from a modeling viewpoint, we don't want to have to specify such a large number of probabilities. Second, performing computations with such a large set of probabilities will be prohibitively expensive. Instead, we would like to specify only a small number of probabilities (e.g., intersections only between pairs or subset of the pairs of topics in Computer Science), and to efficiently compute the other probabilities that may be needed in the process of source selection.

Achieving this ideal requires that we pay attention to two issues. First, not every compact given set of probabilities entails a *unique* probability distribution. For example, if we state that  $P(S_1 | S_2) = 0.2$  and that  $P(S_1 | S_3) = 0.5$ , this does not completely specify  $P(S_2 | S_3)$ . Second, not every given set of probabilities may be *consistent*. For example, suppose we

<sup>2</sup>Since Logic Programming is a subtopic of AI, and  $S_2$  contains 10% of AI papers, we can assume (in the lack of other information) that  $S_2$  contains 10% of Logic Programming papers.

have stated that Source 1 provides half of the papers on Databases (denoted by  $P(S_1 | DB) = 0.5$ ), and that Source 2 provides 60% of Database papers (i.e.,  $P(S_2 | DB) = 0.6$ ). These two statements already entail that there must be *some* overlap between  $S_1$  and  $S_2$ .

In order to obtain unique and complete probability distributions we need to (1) limit the kinds of probabilistic assertions we can make and (2) make several *independence assumptions* about the probability distribution that enable us to compute the missing probabilities. We describe our formalism next.

### 3.3 Specifying Probabilistic Information

We recall some basic terminology. The probability of a query  $Q$ , denoted by  $P(Q)$  is the probability that an object randomly chosen from the domain is an answer to  $Q$ . Similarly, we can define the probability of a source  $S$ ,  $P(S)$ , to be the probability that an object appears in  $S$ . In our discussion we will specify and compute *conditional probabilities*.<sup>3</sup> For example, the conditional probability of  $Q_1$  given  $Q_2$ , denoted by  $P(Q_1 | Q_2)$ , is the probability that an object that is an answer to  $Q_2$  is also an answer to  $Q_1$ . Given a query  $Q$  and a source  $S$ , the final goal is compute  $P(S | Q)$ , i.e., the probability that an answer to  $Q$  appears in the source  $S$ .

#### 3.3.1 Probabilistic Information in the Mediated Schema

We begin with the task of specifying probabilistic information in the mediated schema, i.e., overlaps between the collections of objects in the schema. The overlap between a collection  $c_1$  and a collection  $c_2$ , denoted by  $P(c_2 | c_1)$  is the conditional probability that an object belonging to  $c_1$  also belongs to  $c_2$ . Our goal is to define a complete probability distribution on the set of collections that enables us to compute the conditional probabilities of  $P(Q_1 | Q_2)$ , for  $Q_1$  and  $Q_2$  being arbitrary queries. To encode such a probability distribution efficiently, we propose the following *tree-encoding* (shown in Figure 2).

To define the tree, we assume there exists some (arbitrary) ordering  $c_0, \dots, c_n$  on the set of collections, where  $c_0$  is a superset collection of  $c_1, \dots, c_n$ . Every node  $m$  in the tree, has a label  $L(m)$  associated with it, which is a schema query. The root of the tree is labeled with  $c_0$ . The children of the root are labeled from left to right with  $c_1, (\neg c_1 \wedge c_2), \dots, (\neg c_1 \wedge \dots \wedge \neg c_{n-1} \wedge c_n), (\neg c_1 \wedge \dots \wedge \neg c_n)$ .

In general, a node is labeled by the schema query  $l_1 \wedge \dots \wedge l_i$ , where  $l_i$  is either  $c_i$  or  $\neg c_i$ . The children

<sup>3</sup>Recall that the conditional probability  $P(A | B)$  is equal to  $P(A \wedge B)/P(B)$ .

of the node  $m$  labeled  $L(m) = l_1 \wedge \dots \wedge l_i$  are labeled as follows from left to right:  $(L(m) \wedge c_{i+1})$ ,  $(L(m) \wedge \neg c_{i+1} \wedge c_{i+2})$ ,  $\dots$ ,  $(L(m) \wedge \neg c_{i+1} \wedge \neg c_{i+2} \wedge \dots \wedge \neg c_{n-1} \wedge c_n)$ ,  $(L(m) \wedge \neg c_{i+1} \wedge \neg c_{i+2} \wedge \dots \wedge \neg c_n)$ .

Finally, with every node  $m$ , we associate the probability  $P(L(m))$ . To compute the probability of a query  $Q$ , we add up all the probabilities of all in the set  $\mathcal{N}$  constructed as follows:

- Initially,  $\mathcal{N}$  includes all nodes in the tree whose label is a superset of the conjuncts of  $Q$ .
- Then, we remove from  $\mathcal{N}$  any node whose parent is also in  $\mathcal{N}$ .

Consider the tree shown in Figure 2. To compute the probability of Artificial Intelligence papers, we add the probabilities of  $P(\neg DB \wedge AI)$  (from level 1) and  $P(DB \wedge AI)$  in level 2 obtaining 0.21.

The most important property of the tree encoding is that it specifies a *unique* and *complete* probability distribution over the set of collections. Hence, we can compute any conditional probability of the form  $P(Q_1 | Q_2)$ , where  $Q_1$  and  $Q_2$  are queries.

In the worst case, the size of the tree can be exponential in the number of collections (but still less than the total number of possible queries over the set of collections which is  $3^n$ ). In practice, however, in many domains it is the case that objects belong to a very few number of collections at a time. As a consequence, many nodes in the tree will have a probability of 0, and hence the tree can be stored compactly. In particular, if objects can belong at at most  $l$  collections at a time, the tree will have at most  $O(n^l)$  nodes.

The time to compute the probability of a query  $Q$  from the tree is at most linear in the number of nodes in the tree. It is interesting to note that computing the probability of a query  $Q$  depends on the position of the collections that are mentioned in  $Q$  in the ordering of collections. In particular, computing the probability of  $c_i$  requires looking at at most  $2^i$  nodes. In order to avoid probabilistic computations at run-time, it is possible to precompute from the tree the probability of every possible schema query. Even then, the tree is a useful tool for easing the specification of the probability distribution.

Another important feature to note about the tree encoding is the fact that it gracefully incorporates a strict subsumption hierarchy. Let's assume that we have a partial ordering on the set of collections in the mediated schema describing a strict inclusion relationship between collections (e.g., Agents is a subset of AI). When building the tree we choose an ordering on the collections satisfying the following constraint. If  $c_1$  includes  $c_2$ , then  $c_1$  appears in the ordering *before*  $c_2$ . Assuming such an ordering, the number of nodes

in the tree drastically decreases. It is obvious to note that the subtrees rooted at nodes whose label includes  $\neg c_1 \wedge c_2$  will be empty (i.e., all nodes have probability 0).

### 3.3.2 Coverage of Information Sources

The second kind of probabilistic information we specify is the coverage of each source. Recall that every source  $S$  is described by an expression  $S \subseteq Q_S$ , denoting that the set of objects in  $S$  is a subset of the objects described by the query  $Q_S$ . In the probabilistic setting, we add to the description the conditional probability  $P(S | Q_S)$ . This denotes the probability that an object belonging to the answer to  $Q_S$  is found in the source  $S$ . For example, the statement  $P(S_1 | AI \wedge DB) = 0.5$  means that the source  $S_1$  contains 50% of papers on AI and Databases. Note that  $S \subseteq Q_S$  entails that  $P(S | \neg Q_S) = 0$ .

Recall that the final goal is to compute  $P(S | Q)$ , i.e., the probability that an answer to a query  $Q$  will be found in the source  $S$ . For this goal, it is not enough to know the probability that an object is in  $S$ , given that it is in  $Q_S$ . In addition we make the following conditional independence assumption: not only are the objects in  $S$  a subset of  $Q_S$ , but they are *uniformly* distributed over  $Q_S$ , that is the query  $Q_S$  describing the source renders the source *independent* of any other properties the objects may have. Formally, this independence assumption is stated as follows:

$$P(S | Q_S, Q) = P(S | Q_S).$$

Given this independence assumption, we can derive the following equality that enables us to compute the probability that a source contains an answer to a query  $Q$  using the probabilistic information in the mediated schema:

$$\begin{aligned} P(S | Q) &= P(S | Q_S, Q) \cdot P(Q_S | Q) + \\ &\quad P(S | \neg Q_S, Q) \cdot P(\neg Q_S | Q) \\ &= P(S | Q_S, Q) \cdot P(Q_S | Q) \\ &= P(S | Q_S) \cdot P(Q_S | Q). \end{aligned}$$

The first term is simply the coverage of the source (which is given), and the second value can be computed from the tree. For example, suppose our query asks for papers on Databases and Operating Systems. The probability that answers to the query are in source  $S_1$  is  $P(S_1 | DB \wedge OS) = P(S_1 | AI \wedge DB) \cdot P(AI \wedge DB | DB \wedge OS)$ .

One reason that our independence assumption is a reasonable one in practice is that  $Q_S$  is supposed to be the most restrictive query characterizing the contents of the source. Hence, given that we do not have more specific information about the contents of  $S$ , it

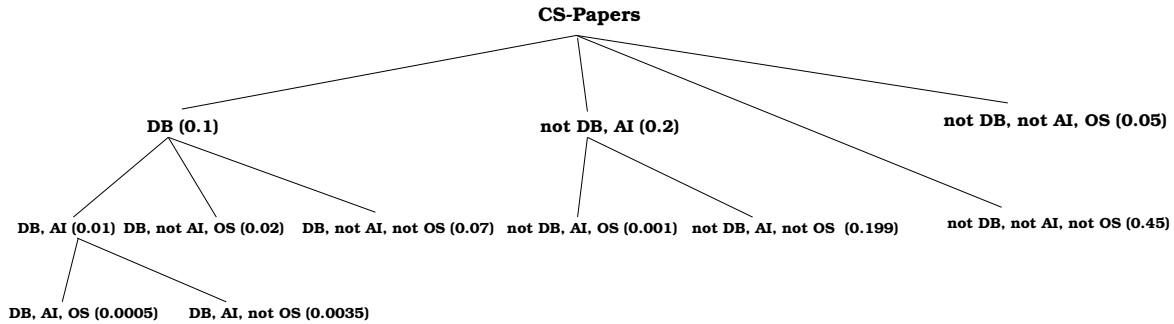


Figure 2: A probability distribution on Computer Science topics.

is reasonable to assume that within  $Q_S$ , the contents of  $S$  are uniformly distributed.

Furthermore, the independence assumption also provides the following important advantage in practice. A provider of a data source only has to describe the coverage of the source w.r.t. the query describing it. He does not have to describe the coverage w.r.t. *every* collection (or combination of collections) in the mediated schema (some of which he will know little about). The coverage of other topics in the source is automatically derived from the information in the mediated schema and the source description.

### 3.3.3 Overlap Between Information Sources

To motivate the need for computing the overlap between information sources, consider the following scenario. Suppose we have accessed a source  $S_1$  and we are trying to decide which source to query next. Given a source  $S$ , it is not enough to compute  $P(S | Q)$ . Instead, we really need to know  $P(\neg S_1 \wedge S | Q)$ . That is, we need to know what is the probability of finding an answer of  $Q$  in  $S$  which does *not* appear in  $S_1$ .

In the absence of any additional information, we can assume that the contents of the sources are independent of each other. That is, the presence of an object in one source does not change the probability that the object belongs to another source. In this case, we can compute the overlap of  $S_1$  and  $S_2$  w.r.t. a query  $Q$  as follows (using values that can be computed from our tree and coverage information):

$$\begin{aligned}
 P(S_1 \wedge S_2 | Q) &= P(S_1 \wedge S_2 | Q_1, Q_2, Q) \cdot \\
 &\quad P(Q_1 \wedge Q_2 | Q) \\
 &= P(S_1 | Q_1) \cdot P(S_2 | Q_2) \cdot \\
 &\quad P(Q_1 \wedge Q_2 | Q). \quad (1)
 \end{aligned}$$

The probability  $P(\neg S_1 \wedge S_2 | Q)$  is simply  $P(S_2 | Q) - P(S_1 \wedge S_2 | Q)$ . Note that this process can be generalized to compute a probability of the form  $P(\neg S_1 \wedge \dots \wedge \neg S_{i-1} \wedge S_i | Q)$ , though we need to add up  $2^i$  terms.

In practice, data sources are not always independent of each other, since many sources are correlated (e.g., they may be derived from other sources by being partial copies or by integration). When we have additional information about source overlap, we would like to be able to use it.

Suppose we are given two correlated sources,  $S_1$  and  $S_2$ , that are described by the queries  $Q_1$  and  $Q_2$ , respectively (i.e., we are given  $P(S_1 | Q_1)$  and  $P(S_2 | Q_2)$ ). If we are also given overlap information, expressed by the probabilities  $P(S_1 | S_2, Q_1, Q_2)$  and  $P(S_2 | S_1, Q_1, Q_2)$ , we can also compute  $P(\neg S_1 \wedge S_2 | Q)$ . Unfortunately, unlike the previous case, this process does not easily generalize to compute  $P(\neg S_1 \wedge \dots \wedge \neg S_{i-1} \wedge S_i | Q)$ . It does generalize when the pairs of correlated sources are pairwise disjoint. However, if a source  $S_1$  is correlated to more than one source (e.g., to  $S_2$  and  $S_3$ ), then we need to specify all the combinations of the form  $P(S_A | Q_A)$ , where  $S_A$  is a subset of  $\{S_1, S_2, S_3\}$ , and  $Q_A$  is a subset of  $\{Q_1, Q_2, Q_3\}$ . A possible solution to this problem is discussed in Section 6.

## 4 Algorithms

We now consider the problem of ordering the information sources. Assume that the mediator can access  $k$  data sources in parallel at any given point. For example, if the data sources are on the WWW, this is the value of  $k$  that maximizes the network bandwidth. Our goal is to develop an algorithm that will give us the *best*  $k$  sources to access at a given point. In this paper we assume that the cost of accessing every source is the same.

Formally, we want the  $k$  sources to maximize the probability of finding answers to the query. That is, given a query  $Q$ , we want to choose sources  $S_1, \dots, S_k$ , that maximize the value of  $P(S_1 \vee \dots \vee S_k | Q)$ . This value can be computed by the following formula, whose terms can be computed as described in Section 3.

$$\begin{aligned}
 P(S_1 \vee \dots \vee S_k | Q) &= P(S_1 | Q) + P(S_2 \wedge \neg S_1 | Q) + \\
 &\quad \dots + P(S_k \wedge \neg S_1 \wedge \dots \wedge \neg S_{k-1} | Q). \quad (2)
 \end{aligned}$$

In general, finding the  $k$  sources that maximize the probability of finding answers requires considering every subset of  $k$  sources. Considering all such subsets would be prohibitively expensive at run-time (in particular, it would entail considering  $O(n^k)$  possibilities, where  $n$  is the number of sources). Therefore, we describe efficient algorithms based on two effective heuristics. In the next section we show that in our experiments the algorithms are both efficient and produce nearly optimal orderings.

#### 4.1 Two Greedy Algorithms

The first algorithm, **simple-greedy**, only uses the information in the mediated schema and the coverage information. Given a query  $Q$ , the algorithm orders the sources according to the value of  $P(S_i | Q)$ , which is the probability of finding an answer to  $Q$  in  $S$ , and chooses the best  $k$  sources. The potential overlap between sources are completely ignored, except that we do not include a source  $S_1$  in the set of  $k$  selected sources if there is another source  $S_2$ , such that  $P(S_2 | S_1 \wedge Q) > 1 - \epsilon$ , where  $\epsilon$  is very small (this would imply that the source  $S_1$  is nearly subsumed by  $S_2$ ).

The number of times that the algorithm invokes the probabilistic computation is linear in the number of sources. Since the algorithm ignores the overlaps between sources, it produces nearly optimal solutions when the overlaps between sources are relatively small. To analyze the degree of overlap we distinguish two cases: (1) source overlap is estimated based on the conditional independence assumption, and (2) we have explicit overlap information.

In the first case, the overlap is estimated using Equation 1. The overlap is proportional to the product of the coverages of the sources within the query describing them, and to the conditional probability that an answer of the query is in both queries describing the two sources. Therefore, the overlap is expected to be small unless all these factors are simultaneously large. Thus, the algorithm will in general produce nearly optimal orderings except for the cases when we have several sources that are nearly complete, and the conjunction of their descriptions nearly covers the required query.

In the second case, the overlap between the sources may be arbitrarily skewed. Consequently, cases can be designed for which the algorithm behaves arbitrarily badly. However, since the algorithm checks for nearly subsumed sources, it avoids bad behavior in at least one common case of source correlation.

The second algorithm, **greedy-select**, is shown in Figure 3. In the first iteration, it computes the probability of finding answers to the query in each source,

```

procedure greedy-select( $Q, S_1, \dots, S_n, k$ )
begin
   $Selected = \{\}$ .
   $S = \{S_1 \dots S_n\}$ .
  until  $Selected$  contains  $k$  sources or  $S$  is empty do
    for each  $S \in S$ ,
       $p_S = P(S \wedge \bigwedge_{S_i \in Selected} \neg S_i | Q)$ .
      let  $S_{max}$  be the source in  $S$  for which  $p_S$  is
        maximal.
      add  $S_{max}$  to  $Selected$  and remove it from  $S$ .
  return  $Selected$ .
end greedy-select.

```

Figure 3: Greedy algorithm for ordering sources.

i.e., for each source it computes  $P(S_i | Q)$ , and it selects the source that maximizes this value. Assume the sources  $S_1, \dots, S_i$  were selected in the first  $i$  steps. In step  $i + 1$ , we compute for each remaining source  $S$ ,  $P(\neg S_1 \wedge \dots \wedge \neg S_i \wedge S | Q)$ , i.e., the probability of finding an answer to  $Q$  in  $S$  that has not been found in the previous chosen sources. We select the source that maximizes this value, and continue in a similar fashion until  $k$  sources are selected. Note that the algorithm is independent of whether we have explicit overlap information about the sources or whether we are using the conditional independence assumption for calculating  $P(\neg S_1 \wedge \dots \wedge \neg S_i \wedge S | Q)$ .

The running time of the algorithm is  $O(2^k \cdot n)$ , where  $n$  is the number of sources (though, assuming  $n$  is large and  $k$  is small, still very efficient compared to the  $O(n^k)$  of the optimal algorithm). In contrast to algorithm **simple-greedy**, this algorithm does take into consideration possible source overlap, and can produce better orderings when the sources are strongly correlated.

#### 4.2 A Merge Algorithm

The third algorithm, **merge-select**, shown in Figure 4, relies on a set of precomputed orderings and merges them appropriately at run-time. For every atomic query,  $q$ , of the form  $c(X)$  or  $\neg c(X)$ , we compute and store the ordering of the sources according to the value  $P(S_i | q)$ . Each ordering also associates a numerical utility with every source, which is consistent with the ordering. At run-time, given a query of the form  $q_1(X) \wedge \dots \wedge q_m(X)$ , the algorithm considers every source  $S$ , and *merges* the relative ordering of  $S$  in the orderings for  $q_1, \dots, q_m$ , using a specific merging function. The sources returned are the  $k$  sources for which the merged ordering is the highest. Note that the precomputed orderings that the algorithm uses do not take into consideration the source overlaps. The reason for this is that an order which would take into

```

procedure merge-select( $Q, S_1, \dots, S_n, k, \mathcal{O}_1, \dots, \mathcal{O}_m, f$ )
//  $\mathcal{O}_i$  is the optimal orderings for the  $i$ 'th conjunct of  $Q$ .
//  $f : \mathcal{R}^m \rightarrow \mathcal{R}$  is a merging function for utilities.
begin
let  $\mathcal{S}$  be the set of sources that appear in the first  $k$ 
positions of at least one of  $\mathcal{O}_1(S_i), \dots, \mathcal{O}_m(S_i)$ .
for  $S_i \in \mathcal{S}$ 
 $u_i = f(\mathcal{O}_1(S_i), \dots, \mathcal{O}_m(S_i))$ 
where  $\mathcal{O}_j(S_i)$  denotes the utility of  $S_i$  in the
ordering  $\mathcal{O}_j$ .
let  $S'_1, \dots, S'_n$  be the ordering of  $\mathcal{S}$  by descending value
of  $u_i$ .
return  $S'_1, \dots, S'_k$ .
end merge-select.

```

Figure 4: Merge algorithm for ordering sources.

consideration source overlaps will not be preserved after removing a chosen source from the order, which happens during the merging.

Two factors make the running time of this algorithm especially efficient. First, the running time depends only on the value of  $k$ , and not on the number of sources. Second, the algorithm does not perform probabilistic computations at run-time. The intuition behind this algorithm is that in order for a source to be good for a conjunction  $q_1 \wedge \dots \wedge q_m$ , it should be good for at least one of the conjuncts. One case in which this will fail to be true is when we have a source  $S$ , described by a query  $Q_S$ , which is a conjunction of  $q_i$ 's,  $S$  has relatively good coverage  $Q_S$ , and  $Q_S$  has relatively low coverage of each of the conjuncts composing it. For example, a source giving good coverage of  $AI \wedge OS$  may be missed, since the conjunction covers very little of either AI or OS. Note that in the common case in which sources are described by a single collection, this cannot happen.

## 5 Experiments

The algorithms presented in the previous section trade off optimality of the source ordering with run-time performance. In this section we briefly describe the implementation of the algorithms within the Information Manifold system [LRO96] and the preliminary experiments showing their utility. The goal of the experiments is to test the relative running times of the different algorithms, and to compare the quality of the orderings they produce w.r.t. the optimal ordering.

Our experiment considered the domain of Computer Science publications. Aside for collections describing classes of authors, our mediated schema included the hierarchy of 30 subtopics of Computer Science. The probabilistic information about overlaps between topics in the mediated schema was determined

statistically by analyzing a corpus of labeled Computer Science abstracts<sup>4</sup> (i.e., each abstract was labeled with the set of topics associated with the paper). We considered a set of 80 data sources, all of which are bibliography servers found on the WWW (some gleaned from the Glimpse bibliography server [BDMS94]). The descriptions of the sources were given manually.

We considered two different scenarios w.r.t. source coverages. In the first, we assumed none of the sources give high coverage of their descriptions. In this case, all sources gave coverages of 20%–40% of their description. In the second scenario, 5% of the sources had high coverage (more than 70%), and the rest had low coverage. In each scenario, the coverages were generated randomly.

In each scenario we varied the size of the queries between a single conjunct and 3 conjuncts, and tested the algorithms with a value of  $k = 3$ , i.e., to choose the three best sources. We measured the running time of each algorithm and the value of the ordering it produced, i.e., the value of  $P(S_1 \vee \dots \vee S_3 \mid Q)$ . We denote this value by the function  $f$ .

The overlap between data sources was computed based on the assumption that sources are independent (i.e., Equation 1). The merge function used in **merge-select** was a simple sum of the relative positions in each of the orderings. We also implemented an algorithm that computes the optimal ordering by considering all subsets of  $k$  sources. For each algorithm we measured the ratio of the utility of the resulting ordering to that of the optimal algorithm and to that of **simple-greedy**. We also measured the ratio of the running time of the algorithm to the running time of **simple-greedy**.

The results for **merge-select** are shown in Figure 1. The results show that the running time of **merge-select** is significantly lower than that of **simple-greedy** and **greedy-select**. The reason for this is that **merge-select** does not perform probabilistic computations at run-time. The quality of the merge algorithm depends on the number of conjuncts in the query. With only one or two conjuncts, there is no surprise that the merge algorithm is the most appropriate. However, the performance degrades as the number of conjuncts increases.

The results for **greedy-select** are shown in Figure 2. They show that the running time of **simple-greedy** is lower than that of **greedy-select**, by a factor of up to 10. Since in our experiments the source overlap is computed by assuming the sources are independent of each other, there are no significant differences between the utility of the orderings computed by **greedy-select** and **simple-greedy**.

<sup>4</sup>Found in <ftp://ftp.cs.cornell.edu/pub/smart/cacm>.

	Scenario 1			Scenario 2		
Query size	$f/f(\text{Optimal})$	$f/f(\text{Greedy1})$	$\text{time}/\text{time}(\text{Greedy1})$	$f/f(\text{Optimal})$	$f/f(\text{Greedy1})$	$\text{time}/\text{time}(\text{Greedy1})$
1	0.92	1	0.04	0.93	1	0.04
2	0.91	0.95	0.006	0.92	0.96	0.005
3	0.80	0.7	0.02	0.79	0.72	0.02

Table 1: Experimental results for algorithm **merge-select**

	Scenario 1			Scenario 2		
Query size	$f/f(\text{Optimal})$	$f/f(\text{Greedy1})$	$\text{time}/\text{time}(\text{Greedy1})$	$f/f(\text{Optimal})$	$f/f(\text{Greedy1})$	$\text{time}/\text{time}(\text{Greedy1})$
1	0.95	1	4	0.96	1	4.1
2	0.96	1	6	0.95	1	6.2
3	0.98	1	10.5	0.96	1	10

Table 2: Experimental results for algorithm **greedy-select**

## 6 Conclusions and Related Work

In this paper we have introduced the use of probabilistic information to the problem of data integration. The key import of probabilistic information is that it enables us to order the accesses to large collections of information sources in a way that provides answers as soon as possible to queries. We presented a formalism that expresses three important kinds of probabilistic information: overlaps between collections in the mediated schema, coverage of information sources and overlap between information sources. The key advantage of our formalism is that it enables us to specify a complete and consistent probability distribution by providing a relatively small number of probabilities. We proposed several effective algorithms for utilizing the probabilistic information. The algorithms represent a tradeoff between computational cost and the optimality of the resulting orderings. Finally, we discussed and evaluated experimentally the advantages and weaknesses of our algorithms. The experiments showed that the algorithms perform efficiently in practice and produce orderings that are close to optimal.

In order to better illustrate the novel aspects of probabilistic reasoning in mediator systems, we have purposely simplified some aspects of our framework. To complete the discussion, we now discuss several important extensions to the basic framework, which our experience has shown to be of practical importance. The extensions concern the query language, the source descriptions, specification of source overlap, and incorporating costs of accessing sources.

In our discussion we limited the user queries and schema queries to conjunctions of collections literals and literals specifying the value of an attribute. The techniques described in this paper extend (using the semantics of the query language P-CLASSIC [KLP97]) to queries containing simple kinds of existential quantifiers. In particular, queries can contain pairs of conjuncts of the form  $\exists Y[X.R = Y \wedge c(Y)]$ , selecting objects for which at least one value for the attribute

$R$  is a member of the collection  $c$ , and can contain conjuncts of the form  $\forall Y[X.R = Y \Rightarrow c(Y)]$ , selecting objects for which all values of the attribute  $R$  are members of the collection  $c$ . Note that such conjuncts do not allow to specify arbitrary joins. Supporting a more general query language would require developing appropriate semantics, which is known to be a non-trivial problem.

We limited each source to be described by a single schema query (and the associated coverage). A important extension is to enable the query describing the source to be a union of conjunctive queries. In practice, some sources may cover more than one collection, and we do not want to describe it by the least common ancestor of these collections, which may be an over-generalization. Fortunately, this extension can be supported in our framework because all the definitions we provided apply to unions of conjunctive queries as well.

In order not to require the specification of an exponential number of probabilities, we limited the kinds of source overlap information that we can provide. We are currently considering an extension based on Bayesian networks [Pea88]. Informally, the advantage of Bayesian networks is that they cut down drastically the number of joint probabilities that need to be specified, while still providing a unique and complete probability distribution. Informally, for a given source  $S$ , a Bayesian network only requires to specify the combinations of sources for which explicit overlap with  $S$  is given.

The work in this paper is based on the assumption that the cost of accessing the data sources is the same for all sources. While our probabilistic formalism would not need to be changed in order to accommodate costs of sources, we would need to reconsider the utility function we are trying to maximize and modify our algorithms appropriately. It should be noted that the question obtaining cost estimates for external data sources is still an active subject of research [ACPS96].

Our work can be compared with works in several domains. In database systems, the main use of statistical information about data which have been used in the past concerns selective estimation for the purpose of computing join orders for query plans (e.g., [IC93]). In contrast, in our work the goal of the probabilistic information is to order the accesses to the information sources in such that we obtain as many answers as possible early on, as opposed to minimizing the cost of answering the query.

There is a large body of work in the Artificial Intelligence community on probabilistic reasoning, using Bayesian networks as the main representational tool. However, as it turned out, Bayesian networks turn out not to be well suited for representing a possibly overlapping collection hierarchy. The reason is that encoding such a hierarchy in a Bayesian network results in a net with a high degree of connectivity, and in this case, computing the needed probabilities from the net is computationally expensive. Hence, we developed a novel tree structure that is better suited for our purposes. However, it is interesting to note that our tree structure by itself, will not scale up well if we want to represent multiple related hierarchies. In such a case, we obtain significant advantages by combining the tree structures via a Bayesian network like structure.

Our work can also be contrasted with information retrieval systems. Ideally, if the entire contents of the external data sources were available to the mediator, we could use an information retrieval system to evaluate which source best covers a set of constants given in a query. However, in our setting we are only given descriptions of the information sources, and not the contents themselves. Furthermore, an information retrieval system would only enable us to provide a set of keywords as a query, and not a relational query over a schema as we would like.

An important issue in using probabilistic information is how we obtain the values for the probabilities. In many cases the probabilities can be derived from statistical information. This information can come directly from the data sources, or from other corpora. For example, in our system the probability distribution on the topics of Computer Science was obtained from a labeled corpus of Computer Science abstracts. Several strategies can be devised for computing values for the coverages of sources. For example, one can compute the coverage of a source by comparing it with a source that is known to be almost complete, or with the union of the contents of the available sources (which can be considered as an approximation of a complete sources). Furthermore, several techniques have been developed for automatically learning probabilistic models (see [Hec96] for a survey), and these techniques can be adapted for our context as well.

## References

- [ACPS96] S. Adali, K. Candan, Y. Papakonstantinou, and V.S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proceedings of SIGMOD-96*, 1996.
- [AKS96] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. *International Journal on Intelligent and Cooperative Information Systems*, (6) 2/3:99–130, June 1996.
- [BDMS94] C. Mic Bowman, Peter B. Danzig, Udi Manber, and Michael F. Schwartz. Scalable internet resource discovery: Research problems and approaches. *CACM*, 37(8):98–107, August 1994.
- [Buc85] Chris Buckley. Implementation of the SMART information retrieval system. Technical Report TR85-686, Department of Computer Science, Cornell University, Ithaca, NY 14853, May 1985.
- [CGMH<sup>+</sup>94] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In proceedings of IPSJ, Tokyo, Japan, October 1994.
- [EW94] Oren Etzioni and Dan Weld. A softbot-based interface to the internet. *CACM*, 37(7):72–76, 1994.
- [FRV96] D. Florescu, L. Raschid, and P. Valduriez. A methodology for query reformulation in cis using semantic knowledge. *Int. Journal of Intelligent & Cooperative Information Systems, special issue on Formal Methods in Cooperative Information Systems*, 5(4), 1996.
- [Hec96] David Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, 1996. Available from <http://www.research.microsoft.com/research/>.
- [IC93] Y. Ioannidis and S. Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM Transactions on Database Systems (TODS)*, 18(4):709–748, December 1993.
- [KLP97] Daphne Koller, Alon Levy, and Avi Pfeffer. P-CLASSIC: a tractable probabilistic description logic. Proceedings of the AAAI Fourteenth National Conference on Artificial Intelligence, 1997.
- [KW96] Chung T. Kwok and Daniel S. Weld. Planning to gather information. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, 1996.
- [LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd VLDB Conference, Bombay, India*, 1996.
- [Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.
- [TRV97] A. Tomasic, L. Raschid, and P. Valduriez. A data model and query processing techniques for scaling access to distributed heterogeneous databases in disco. *IEEE Transactions on Computers, special issue on Distributed Computing Systems*, 1997.