Cooperative Query Answering with Density Scores

Felix Naumann
Humboldt-Universität zu Berlin
naumann@dbis.informatik.hu-berlin.de

Ulf Leser
Technische Universität Berlin
leser@cs.tu-berlin.de

Abstract

Mediator-based information systems answer global queries by rewriting them into a combination of queries against physical data sources. One assumption in most systems is that only such combinations are considered as valid that obtain values for each selected attribute of the query. Another assumption is that systems must compute and execute all valid combinations, i.e., they strive to retrieve all possible answers. These assumptions frequently lead to user frustration: First, in many scenarios an incomplete answer is much more appreciated than no answer at all. Second, if many valid combinations exist, it is very time-consuming to execute them all.

We present a cooperative query planning method that avoids both problems. First, it treats incomplete and complete source combinations in a logically equivalent manner, i.e., incomplete answers are also considered. Second, it ranks and selects combinations based on their information density, i.e., based on the expected usefulness of the results. Our selection process naturally prefers complete answers, but can also cope with incomplete answers. Furthermore, users can guide the ranking by specifying cost constraints and preferences. We present algorithms to counter the exponential nature of the problem.

ADVANCES IN DATA MANAGEMENT 2000 Krithi Ramamritham, T.M. Vijayaraman (Editors), Tata McGraw-Hill Publishing Company Ltd.,

1 Cooperative Information Integration

Information integration has two facets. In mission-critical OLTP systems, information integration stands for integrated access to a given set of databases. For instance, globally operating companies strive for decision support systems that allow queries ranging over databases at different locations [14]. The expected behavior of such a *database federation* is that of a central database. Therefore, answers must be complete. The answer to a query against the federated schema must contain values for all selected attributes, and include tuples from all relevant data sources. Furthermore, it is expected that the data sources are well-maintained and reliable. Decision makers will not accept query results of a low quality because incomplete or missing answers may result in wrong and possibly costly decisions. However, recent years, and especially the success of the World Wide Web, have fostered different types of applications that also require information integration. In those applications, users do not expect perfect and complete query results. Actually, they do not even want "complete" answers since a complete answer drowns them in an undigestible flood of information. Instead, users expect *cooperative behavior* of the query engine.

Example 1 Imagine a user searching for *some* books written by Stephen King, including book-title, reviews, and, *if possible*, the publisher:

u_1(title, publisher, review) ← books(ISBN, author, title, publisher, review, price), author='King';

Imagine two data sources S_1 and S_2 providing information about books. S_1 exports authors and titles, and occasionally includes a review. S_2 offers reviews for each book it has in its database. Answering user query u_1 using a traditional, non-cooperative query engine fails because neither S_1 nor S_2 provide publisher information. Most users will perceive this answer as frustrating and misleading. By stating *desirable but not necessary* variables in the query, the user inadvertently reduces the result size.

Imagine that a cooperative user recognizes the problem and poses a new query that does not select publisher. A conventional integration engine will find that both S_1 and S_2 can contribute to the answer. Therefore, both sources are accessed. However, it is likely that the process will also be perceived as frustrating: First, it takes time to query both sources. Second, there was no need to get a complete answer because the user actually wanted only *some* books. For both queries, querying only S_2 would have been the best choice. □

Other scenarios in which users expect cooperative behavior – and not completeness at any cost – include knowledge management systems, web information portals, and bargain finders [6]. In this paper we propose a method to include cooperative behavior into a mediator-based information systems (MIS). MIS have recently become a popular infrastructure for the integration of information from autonomous and heterogeneous data sources [15]. In an MIS, users query multiple data sources simultaneously and transparently by accessing a mediator with a single, homogeneous interface. It is the task of the mediator to dynamically select data sources to find answers to a user query. Following the tradition of federated databases, research on MIS has focused on providing correct and complete answers to any query. Great effort is put into algorithms that find all ways of answering a query. This process is costly and unnecessary if perfect and complete results are not required. Furthermore, it is inflexible and cumbersome to use. In the types of systems we envisage, it is more important to obtain a *sufficient subset of all answers* in reasonable time and at a reasonable price.

To this end, we introduce *density scores* and thereby improve the flexibility of query answering in MIS. Density scores qualitatively describe data sources. A mediator uses density scores to guide the generation and selection of query plans. Intuitively, density scores measure the *fullness* of the attributes exported by a sources. We include two other features into our planning model to further increase its ability to provide satisfying, but not necessarily complete, answers. First, we allow for user- and query-specific *attribute weightings*, since not all attributes are of equal importance in all queries. Second, our planning algorithms carefully obey a *cost constraint*, which expresses an upper border on the cost of computing answers.

Example 2 Recall the previous example. The user query with a user weighting is now:

u_2(title[10], publisher[5], review[8]) ← books(ISBN, author, title, publisher, review, price), author='King';

The numbers give the importance of each attribute in the query, where 10 means "mandatory" and 1 means "negligible". Attributes of selection predicates are always considered mandatory. Hence, u_2 requests answers which must contain title and which should contain the publisher and review. Furthermore, suppose that the user states that at most one source should be queried, since a few answers are already satisfying. The two available sources are described as:

S_1(ISBN[1],author[1],title[1],review[0.5]);
S_2(ISBN[1],author[1],title[1],review[1]);

Here, the numbers give the ratio of objects which have a non-`null` value for the corresponding attribute. The fact that S_1 has only *some* reviews is reflected in a low score for this attribute. Given the user weighting, density scores, and the cost constraint, the mediator chooses which sources are more appropriate for the query at hand. Both sources will be punished for not providing publisher, but this will not exclude them from further consideration. On the other hand, S_2 will benefit from providing more reviews. Therefore, the mediator will select S_2. □

Our approach incorporates various elements of cooperative behavior:

- Users always obtain *some* answers, even if complete answers are not possible.
- Users may *constrain the amount of work* performed during query execution. This shields users from unsatisfyingly long response times.
- Users may express preferences for the *importance* of attributes. This weighting is used to tailor the quality estimation of plans to the specific query needs.

Related work. Information integration using a MIS architecture is addressed by many research projects, such as TSIMMIS [2], Information Manifold [8], and Aurora [16]. Our data model is similar to that of Yerneni et al. [17]. However, none of these projects can cope with queries for which no complete answer exist, and none is capable of selecting plans based on their usefulness. In [11] we proposed a framework for selecting query plans based on information quality. However, there we described algorithms which find the N best plans, while, in this paper, we aim at algorithms that find the single best plan, i.e., the best set of sources. Our understanding of cooperative behavior is close to that described by Minker [9]. However, the seven different aspects reviewed there cannot be adopted easily to the types of MIS that we consider. Regarding their classification, we concentrate on *user goals and preferences* and on *relaxed answers*. Minkers work aims at *explaining* why certain queries have no answers, e.g., by giving intentional answers, while we focus on somehow producing *reasonable answers*. Gaasterland and Lobo describe a central database system that offers support for cooperative behavior in two aspects [3]: First, users can annotate queries with preferences. Second, facts (and rules) are annotated with a *degree of confidence*. Incomplete answers are not considered.

Structure of this paper. The following section gives an overview of our system. Section 3 defines density scores and introduces our approach to use the scores for information integration. In Section 4 we present two algorithms that produce cooperative query execution plans based on density scores and different cost models. Section 5 concludes the paper.

2 Overview

We use a mediator-wrapper architecture as proposed by Wiederhold [15] (see Figure 1). The goal is to access a given set of heterogeneous data sources through a single query interface. Therefore, each source is encapsulated by a *wrapper*. The *mediator* provides a global schema, accepts user queries against it, and plans query execution across the sources. If a query against an data source is to be executed, it is passed to the appropriate wrapper. This wrapper translates the request into some form understandable by the source, e.g., by filling out a WWW form or by compiling an SQL query. The wrapper then submits the query to the source and receives the results. Finally, those results are reformatted and returned to the mediator. Since this paper concentrates on the task of the mediator, we use the terms wrapper and source synonymously.

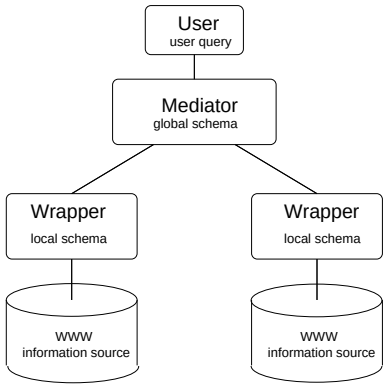

Figure 1: The general mediator architecture

The mediator represents the data comprised by the MIS through a relational, global schema. For simplicity in this paper, we assume that the global schema consists of only one relation. The extension to global schemas with more than one relation can be found in [10]. However, notice that a single relation is a sufficient and convenient model for many applications. Especially on the web, it is often considered as too complex if users must pose queries including joins.

We denote each tuple in the global relation an *object*, which has an identifier (OID) and a set of attributes. For example, in a book information system a book is an object. Its identifier is the ISBN; possible attributes are title, author, etc. We use the OID to merge results obtained from different sources, as explained in Section 3.2. We do not require that OIDs are keys. However, we do assume that OIDs are consistent across sources, i.e., if two sources present an object with the same OID, then the MIS deems these objects to represent the same real-world entity. Such global identifiers exist in many domains: Stocks have ticker symbols, books have an ISBN, persons have a passport number etc. Often, OIDs can be constructed from other values, such as building a person ID from the person's name and address.

2.1 Describing Data Sources

An MIS comprises a set $S = \{S_1, \ldots, S_n\}$ of data sources. Only the sources store and manage information physically and permanently. A *source* S_i represents a set of objects, its size is $|S_i|$. We assume that the content of a data source can be retrieved through some interface, but we make no assumptions about whether or not a data source is capable of performing selections on attribute values. If a user query contains conditions that a source is not able to execute, the mediator may first download all objects and then apply the conditions himself. The problem of deciding where operations should be executed is orthogonal to the problem we are considering (see for instance [5] or [1]). Schematically, a source S_i is a relation with a subset of the attributes of the global relation. Each source must provide the OID. Other attributes may be missing. We call the attributes that are provided by S_i the *exported attributes* of S_i. Each query submitted to a source S_i is associated with a *cost* $c(S_i)$. We assume that this cost is independent from the amount of data retrieved. Hence, any information service charging per-query conforms to this model. We discuss two variations of our cost model in Section 4.

Example 3 In the example our MIS has the global relation

books(ISBN,author,title,publisher,review,price);

and comprises the following four data sources:

S_1(ISBN,author,title,review);
S_2(ISBN,author,title);
S_3(ISBN,author,title,publisher);
S_4(ISBN,author,title,review,price);

□

In Section 3 we extend the description of data sources with *density scores*; each exported attribute of a source is annotated with a number indicating the ratio of objects in

this source which have a non-`null` value for this attribute. We use density scores to qualitatively describe the content of data sources, and to describe the expected quality of answers that are merged from multiple sources.

2.2 User Queries

A user query u is a selection of attributes of the global relation, possibly together with conditions on the attribute values. Each attribute of u is annotated with an *importance weight*, i.e., a number indicating its importance to the user. The weighting can be given within any common range; we use the range from 1 (not important) to 10 (very important). Attributes that are not in u are implicitly assigned the weight zero. The mediator will use the weighting to decide which sources are most useful for the query. Together with every query the user may specify a *total cost limit* L.

Example 4 The user query from the introduction is:

u_2(title[10], publ.[5], review[8]) ← books(ISBN, author, title, publ., review, price), author='King';

Given this weighting, a user will probably be more satisfied with answers obtained from S_4 than with answers obtained from S_3, because S_3 lacks the important review attribute, while S_4 only lacks the less important publisher attribute. □

2.3 Answering User Queries

The purpose of the mediator is to answer a user query u by accessing data sources. The purpose of cooperative query answering is to find a set (or a *plan*) $P \subseteq S$ of sources such that (a) the information requirement of the user expressed through u is fulfilled *optimally*, and that (b) the cost of accessing all sources in P does not exceed the cost limit L. Essentially, we compute P in the following way: First, we assume that each data source *virtually* exports all attributes of the global relation. The density score of those attributes that a source does not export *physically* are set to zero. Next, we consider all possible combinations of data sources. Let P' be such a combination. If the cost of P' if higher than L, then P' is immediately discarded. Otherwise, we compute the *expected density* $d_u(P')$ of the result obtained by accessing all sources in P' and merging the results. Computing $d_u(P')$ takes into account the density of each source for each attribute that is in u, favoring sources with high density scores over sources with low scores. Eventually, the set of sources with the highest expected density is chosen.

Fortunately, it is not always necessary to consider all $2^{|S|}$ combinations of sources. In Section 4 we present two algorithms in $O(|S|^2)$. The first addresses a uniform cost model and achieves optimal results, the second addresses a variable cost model and we show bounded optimality.

3 Density Scores

In this section we describe how the usefulness of a plan for a given user query is computed. We define usefulness as the amount of data a plan returns or its *completeness*. Completeness includes both the number of tuples returned (coverage) and the percentage of non-`null` values in the result (density). We present the entire completeness model in [10]. In this paper we focus on the density measure and how it can be used to improve query results in a cooperative manner.

We proceed as follows: First, we define the source-specific *attribute density* (see Definition 1). Essentially, density measures the amount of non-`null` values in that attribute. For instance, book information sites do not (and often cannot) provide reviews for all books, an address information service will not have the email address of all listed people, etc. Next, we define the *density of a source* wrt. user query u as the average density of the attributes requested in u (see Definition 2).

Definition 3 adds the user weighting, which determines the usefulness of a single source for answering u. In Section 3.2 we show how to compute the density of a plan, i.e., a set of sources, explaining how data from different sources is logically merged.

3.1 Density of a Source

Since each attribute in each source typically is filled to a different degree, the basis of our measure is the attribute level.

Definition 1 *Let A be the set of attributes of the global relation. The density of attribute $a_k \in A$ in source S_i is*

$$d_{S_i}(a_k) := \frac{|\{o \in S_i | o[a_k] \neq \texttt{null}\}|}{|S_i|}$$

Density of an attribute can be interpreted as the probability that an object has a non-`null`-value for that attribute. We assume that all sources export the OID attribute, and that this attribute always has a density of 1. An attribute of the global schema that is not exported by a source, is assigned a density score of zero. All other attributes have values between 0 and 1.

Example 5 For our example we assume the following attribute densities for the four sources:

S_1(ISBN[1],author[1],title[1],review[0.5]);
S_2(ISBN[1],author[1],title[1]);
S_3(ISBN[1],author[1],title[1],publisher[0.5]);
S_4(ISBN[1],author[1],title[1],review[0.25], price[0.9]);

Consider sources S_1 and S_3. They correspond to the global schema in different ways (see Figure 2). Both sources export ISBN, author, and title, each with density 1. Source S_1 additionally exports review, while Source S_3 additionally exports publisher, each with different density scores.

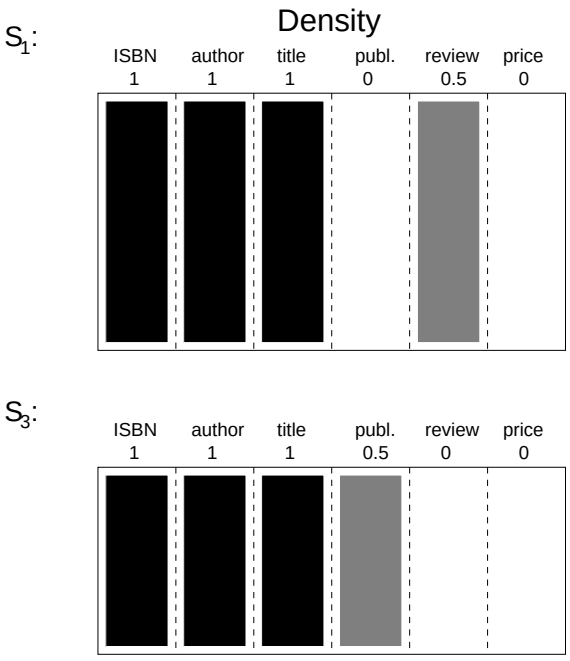

Figure 2: S_1 and S_3 with density scores

Now we define the density of an entire source S_i wrt. a given user query u. Therefore, we consider the attributes of u in the following manner: An attribute exported by S_i that is not in u does not contribute to the source density. An attribute that is in u but not exported by S_i is assigned a density score of 0 and hence lowers the overall density score of the source. The density of S_i wrt. u is the average density of all attributes of u.

Definition 2 *The* density *of a source S_i with respect to a user query u is*

$$d_u(S_i) := \frac{\sum_{a_k \in u} d_{S_i}(a_k)}{|u|}$$

where $|u|$ is the number of attributes in u.

Definition 3 *Let $w_u(a_k)$ denote the weight of a_k in u. The* weighted density *of a source S_i wrt. a user query u is:*

$$d_u(S_i) := \frac{\sum_{a_k \in u} d_{S_i}(a_k) \cdot w_u(a_k)}{\sum_{a_k \in u} w_u(a_k)}$$

Example 6 For u_2(title[10], publisher[5], review[8]), we obtain the following source densities:

$$d_{u_2}(S_1) = \frac{14}{23}; \quad d_{u_2}(S_2) = \frac{10}{23};$$
$$d_{u_2}(S_3) = \frac{12.5}{23}; \quad d_{u_1}(S_4) = \frac{12}{23};$$

Hence, S_1 would promise the most useful result for u_2 if only a single source were used. Merging results from several sources promises better results. □

Definition 3 can already be used for query dependent source selection. Sources with a high density score promise better results. Note again, that we do not consider the size of the source here. Density is only one aspect of the completeness of a source.

3.2 Density of Merged Results

Definitions 1 to 3 define density for single sources. However, a user query is in general not answered by accessing only a single source, but by merging results from a set of sources. We call such a set a *plan* as introduced in Section 2.3. In this section we define the density of plans.

Combining attributes. In general, a result merged from two sources will contain (i) attributes that are exported by both sources (ISBN, author, and title in Figure 3), (ii) attributes exported by only one source (publisher and review), and (iii) attributes that are not exported by either source (price). For the second type of attributes, all available values are used in the merged result, and the density score of the merged result will be the density score of the only attribute providing the data. In the third case, the result will have an empty column, and the density score of this column is 0. The first case is more complicated. Both sources compete in filling the result table with attribute values for those objects that are present in both. Three cases can occur for each object :

- Both sources have a `null`-value ⇒ The result has a `null`-value at that position.
- Exactly one source provides data ⇒ The result uses that data at that position.
- Both sources provide data ⇒ Some resolution function must determine what value will appear in the result table. Resolution functions can be of various types, depending on the type of attribute, the usage of the value, and many other aspects [18, 13].

Calculating merged density. Let $d_{S_i}(a_k)$ and $d_{S_j}(a_k)$ be the density scores of the attribute a_k in the sources S_i and S_j, respectively. Let $d_{S_{i,j}}(a_k)$ denote the density score of a_k of the merged result. Note that we assume mutual independence of attribute values, i.e., the probability that an attribute has a non-`null`-value for an object represented in S_i is independent from its appearance in S_j.

Theorem 1

$$d_{S_{i,j}}(a_k) = d_{S_i}(a_k) + d_{S_j}(a_k) - d_{S_i}(a_k) \cdot d_{S_j}(a_k)$$

The formula of Theorem 1 is associative and commutative but not monotone, i.e., $d_{S_i}(a_k) \leq d_{S_j}(a_k) \not\Rightarrow d_{S_{i,k}}(a_k) \leq d_{S_{j,k}}(a_k)$. This fact makes optimization difficult as we will see in Section 4.1.

Example 7 The density of the review attribute of the merged result of S_1 and S_4 is:

$$d_{S_{1,3}}(\text{publ.}) = 0.5 + 0.25 - 0.5 \cdot 0.25 = 0.625 \square$$

The calculation of the density of the overall merge result essentially remains as in Definition 3, but we now use the merged attribute density scores:

Definition 4 *The merged density of two sources S_i, S_j for a user query u is*

$$d_u(S_i, S_j) := \frac{\sum_{a_k \in u} d_{S_{i,j}}(a_k) \cdot w_u(a_k)}{\sum_{a_k \in u} w_u(a_k)}$$

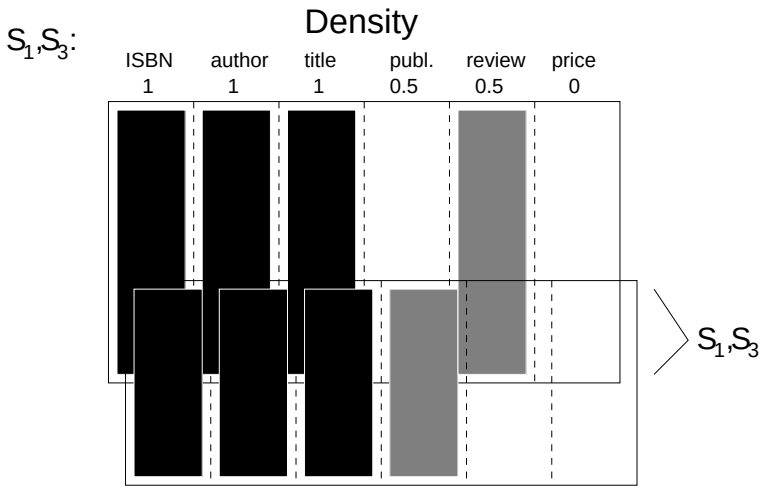

Figure 3: Merged result of S_1 and S_3 with new density scores

This definition only calculates the merged density of two sources but due to its associativity it can be applied to larger sets of sources. As already mentioned, we do not consider the number of objects in a source in this paper. This is equivalent to assuming that all sources cover the same objects. Density is only one aspect of an overall completeness measure. Again, we refer to [10] for the completeness model that includes source sizes and different overlap situations between sources.

Example 8 To illustrate, we give the density scores of the merged results of source S_1 with the other three sources:

$$d_{u_2}(S_1, S_2) = \frac{14}{23}; \quad d_{u_2}(S_1, S_3) = \frac{16.5}{23};$$
$$d_{u_2}(S_1, S_4) = \frac{15}{23}.$$

Hence, the best combination of *two sources* is S_1 and S_3 in the example. □

4 Density-driven Planning

This section describes how we create query execution plans. A plan is a set of sources, representing queries performed by wrappers. These queries need not be executed in a specific order. After execution, each wrapper returns a set of objects, each object with an identifier and some attributes. It is the task of the mediator to merge these objects to a common response to the user. Our goal is to do this in a way that satisfies users most, even when only parts of the query can be answered, and even when only a certain number of sources can be queried.

We assume a cost model in which the mediator must pay some price for each query issued towards a source. This cost model can be found often in existing applications. A mediator that accesses distributed data sources will either pay a price for the data or will have some other license agreement as to how many queries can be submitted during a given period. Based on this cost model, our algorithm strives to find the best possible set of sources within a cost limit L, specified together with the user query. Parallel execution of queries is possible, but not rewarded since response time is not part of our optimization goal. We present two variations of our cost model: The simple case assumes the same cost for each source (uniform cost). For this case we develop an optimal algorithm. The general case assumes variable costs of the sources (variable cost). We develop an algorithm for which we can prove bounded optimality.

4.1 Uniform cost

All queries against sources have the same uniform cost C. Thus, the number of sources that can be queried within the limit is fixed at $\lfloor \frac{L}{C} \rfloor$. Maximizing density would be simple if the density formula were monotone. A greedy algorithm which in each step simply chooses the next most dense source would produce optimal results. However, choosing one source changes the additional density that the remaining sources can supply in a non-monotone way.

Example 9 Consider the simple example of three sources and two attributes:

	a_1	a_2
S_1	(0.6	0.6)
S_2	(1	0)
S_3	(0	1)

A greedy algorithm with a cost limit of two sources would choose S_1 first and next either S_2 or S_3. However, the optimal solution would be to choose only S_2 and S_3. A greedy algorithm misses this optimum because $d_u(S_1) > d_u(S_2)$ but $d_u(S_1, S_3) < d_u(S_2, S_3)$. □

To cope with the exponential nature of the problem we present a slightly refined greedy algorithm (Algorithm 1). It is greedy because in each step it locally chooses the source that will contribute the most (function getMaxSource()). After selecting a source, we reassess the density scores of all remaining sources such that the new scores reflect the *additional density* a source contributes, given the set of sources already selected. If plan P is the set of sources already in the plan, then the reassessed density score for $S_i \notin P$ is $d_u(S_i) = d_u(P, S_i) - d_u(P)$. This refinement "looks ahead" one step as compared to a simple greedy version.

Algorithm 1 initializes a set R of remaining sources with all sources and a set P of already chosen sources which represents the plan. Because all sources have cost C and the limit is L, we can query $\lfloor \frac{L}{C} \rfloor$ sources. In the loop we choose the remaining best source, move it from R to P and reassess all sources in R according to the new current plan P.

The first step of the algorithm simply chooses the source with the highest density. The next source chosen will be the one that complements the first source best. Typically this will be a source that exports different attributes than the first source, since the scores for these attributes were not reduced during reassessment.

Algorithm 1 Uniform Cost High Density Planning

Input: Sources $S_1, \ldots S_n$, user query u, user weighting w, cost limit L, uniform cost C
Output: query plan for u

```
1: R = {S_1, ..., S_n};
2: P = {};
3: for i = 1 to ⌊L/C⌋ do
4:    S = getMaxSource(R);
5:    remove S from R;
6:    add S to P;
7:    reassess scores in R;
8: end for
9: execute P;
```

Example 10 Recall the density scores wrt. u_2 of the four sources:

$$d_{u_2}(S_1) = \tfrac{14}{23}; \quad d_{u_2}(S_2) = \tfrac{10}{23};$$
$$d_{u_2}(S_3) = \tfrac{12.5}{23}; \quad d_{u_2}(S_4) = \tfrac{12}{23}.$$

Let $L = 3$ and $C = 1$, i.e., 3 sources can be queried. The algorithm greedily chooses S_1 to be queried first, i.e., $P = \{S_1\}$ and $d_{u_2}(P) = 14/23$. The reassessed density scores of the remaining sources are

$$d_{u_2}(S_2) = \tfrac{14}{23} - \tfrac{14}{23} = 0;$$
$$d_{u_2}(S_3) = \tfrac{16.5}{23} - \tfrac{14}{23} = \tfrac{2.5}{23};$$
$$d_{u_2}(S_4) = \tfrac{15}{23} - \tfrac{14}{23} = \tfrac{1}{23}.$$

Hence, S_3 is chosen next, i.e., $P = \{S_1, S_3\}$ and $d_{u_2}(P) = 16.5/23$. Again the density scores are reassessed

$$d_{u_2}(S_2) = \tfrac{16.5}{23} - \tfrac{16.5}{23} = 0;$$
$$d_{u_2}(S_4) = \tfrac{17.5}{23} - \tfrac{16.5}{23} = \tfrac{1}{23},$$

and the final plan is $P = \{S_1, S_3, S_4\}$ with an overall density score of $d_{u_2}(P) = \frac{17.5}{23}$. □

The computational complexity of Algorithm 1 is $O(n^2)$ because the loop is performed $\lfloor \frac{L}{C} \rfloor \leq n$ times and within the loop we search for the best source and reassess all remaining sources. Algorithm 1 yields optimal results as we show with the following property:

Theorem 2 *Let P be the set of already chosen sources and let S be a source among the set R of remaining sources. If $d_u(P, S) \geq d_u(P, S_i)$ for all $S_i \in R$, then $d_u(P, S, P') \geq d_u(P, S_i, P')$ for all plans P' containing only sources from R.*

PROOF:

$$\begin{aligned} & d_u(P, S, P') \\ & \quad = d_u(P, S) + d_u(P') - d_u(P, S) \cdot d_u(P') \\ & \quad \geq d_u(P, S_i) + d_u(P') - d_u(P, S_i) \cdot d_u(P') \\ & \quad = d(P, S_i, P') \end{aligned}$$ □

That is, if an algorithm chooses the source S that fits best to the current plan P, the resulting plan will be better than any other plan, regardless of the plan suffix P'. Algorithm 1 follows this choice model.

4.2 Variable cost

If we drop the uniform cost assumption, we turn the problem into a variation of the knapsack problem, which is NP-complete [7]. In the knapsack problem, we have a number of items (sources) with different costs and different benefits (density scores), a limit, and the goal of optimizing the overall benefit. Our problem differs from the traditional knapsack problem, again because the benefit of a source is not fixed, but varies non-monotonically, depending on the sources already queried. Considering the exponential nature of the knapsack problem, it is almost impossible to guarantee an optimal solution for any reasonable number of sources. Polynomial approximation algorithms with optimality guarantees of $\frac{1}{1+\frac{1}{k}}$ and beyond have been presented for the knapsack problem [12]. We use one of the most simple approximation algorithms, which guarantees results within 50% of the optimum [4] and adapt it to our problem.

Algorithm 2 is also greedy, but extends the previous algorithm in two ways: First, to account for variable costs, it selects sources by their density/cost ratio and not simply by their density (function getMaxRatioSource() in Line 5). Also it cannot simply choose a fixed number of sources, but must keep track of the remaining budget (Line 12). Second, before executing sources in that greedy order, the algorithm compares in line 15 the final expected density score with that of the single source with the highest density (not the highest ratio) obtained in Line 14. The better of the two choices is executed. This well-known technique avoids notorious worst cases.

Algorithm 2 Variable Cost High Density Planning

Input: Sources $S_1, \ldots S_n$ with costs $c(S_i)$, user query u, user weighting w, cost limit L
Output: query plan for u

```
1:  R = {S_1, ..., S_n};
2:  P = {};
3:  usedcost = 0;
4:  for i = 1 to n do
5:     S = getMaxRatioSource(R);
6:     remove S from R;
7:     if usedcost+c(S) > L then
8:        break;
9:     end if
10:    add S to P;
11:    reassess scores in R
12:    usedcost = usedcost + c(S);
13: end for
14: S_max = getMaxSource( {}, {S_1, ..., S_n};
15: if d_u(P) < d_u(S_max) then
16:    P = S_max;
17: end if
18: execute P;
```

The complexity of Algorithm 2 also is

$O(n^2)$. The results of the algorithm can be further improved for instance by "filling in" inferior sources if the cost limit is not yet reached. The following theorem shows that the algorithm finds plans that reach a density that is at least 50% of the optimal solution.

Theorem 3 *Algorithm 2 yields 50% optimality.*

PROOF: Let Ra be the density result of the algorithm without lines 14–16, i.e., without the technique of choosing the single best source. Let Al be the density result of the complete algorithm, let $S_{\max}$ be the single source with maximum density, and let Opt be an optimal solution. Let Fr be the fractional solution obtained by adding to Ra a fraction of the next source S_{Next} to be chosen. This fraction of a source has the size to exactly use up the cost limit. Trivially $Fr \geq Opt$, i.e., the density of plan Fr is greater than or equal to the density of an optimal plan. We distinguish the two cases of the algorithm choosing either the greedy sequence or the single largest source.

$$
\begin{aligned}
&Ra \geq S_{\max}: \\
&\quad 2Al = 2Ra \geq Ra + S_{\max} \\
&\qquad \geq Ra + S_{\text{Next}} \geq Fr \geq Opt \\
&Ra < S_{\max}: \\
&\quad 2Al = 2S_{\max} \geq Ra + S_{\max} \\
&\qquad \geq Ra + S_{\text{Next}} \geq Fr \geq Opt
\end{aligned}
$$

□

5 Conclusions

The seamless integration of several information systems poses great challenges to information management, many of which are well described in research literature. However, little work has been reported towards cooperative query answering. We contribute to this issue by extending a mediator-based information system with the ability to (a) answer queries even if no complete answer is available, to (b) constrain the amount of money or time invested in query answering, and to (c) obtain qualitatively good or even optimal answers in a fraction of the time it takes to compute the complete answer. To this end, we introduced density scores, which quantify and qualify the amount of information returned by an data source. Using this measure, we relaxed the traditional conditions on query planning in information integration, i.e., returning complete answers. Instead, our cooperative query answering method aims at producing satisfying plans and answers. For a monetary cost model, we described two algorithms that find such answers.

Density of plans is only one aspect of a completeness model for plans. The combination of density scores with coverage score that reflect the size of a source calculates the true completeness of a source. Future work will include algorithms that optimize completeness for plans.

Acknowledgments. This research was supported by the German Research Society, Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK 316). We also thank Marc Uetz and Ramana Yerneni for many helpful comments.

References

[1] Chen-Chuan K. Chang and Hector Garcia-Molina. Mind your vocabulary: Query mapping across heterogeneous information sources. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 335–346, Philadelphia, 1999.

[2] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of IPSJ Conference*, pages 7–18, Tokyo, Japan, 1994.

[3] Terry Gaasterland and Jorge Lobo. Qualified answers that reflect users needs and preferences. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 309–320, Santiago de Chile, Chile, 1994.

[4] Michael R. Garey and David S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, New York, 1979.

[5] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang. Optimizing queries across diverse data sources. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 276–285, Athens, Greece, 1997.

[6] Reynolds Hadley and Tom Koulopoulos. Enterprise knowledge has a face. *Intelligent Enterprise*, 2(5), 1999.

[7] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations.* Plenum Press, New York, 1972.

[8] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems - Special Issue on Networked Information Discovery and Retrieval*, 5(2):121–143, 1995.

[9] Jack Minker. An overview of cooperative answering in databases. In *Conference on Flexible Query Answering Systems*, pages 282–285, Roskilde, Denmark, 1998.

[10] Felix Naumann and Johann Christoph Freytag. Completeness of information sources. Technical Report 135, Humboldt-Universität zu Berlin, Institut für Informatik, 2000.

[11] Felix Naumann, Ulf Leser, and Johann Christoph Freytag. Quality-driven integration of heterogenous information systems. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, Edinburgh, 1999.

[12] S. Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM*, 22:115–124, 1975.

[13] Peter Scheuermann, Wen-Syan Li, and Chris Clifton. Multidatabase query processing with uncertainty in global keys and attribute values. *Journal of the American Society for Information Science (JASIS)*, 49(3):283–301, March 1998.

[14] Amit Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[15] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), 1992.

[16] Ling Ling Yan, M. Tamer Oezsu, and Ling Liu. Accessing heterogeneous data through homogenization and integration mediators. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 130–139, Kiawah Island, North Carolina, 1997.

[17] Ramana Yerneni, Yannis Papakonstantinou, Serge Abiteboul, and Hector Garcia-Molina. Fusion queries over internet databases. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, Valencia, Spain, March 1998.

[18] C. Yu and W. Meng. *Principles of database query processing for advanced applications.* Morgan Kaufmann, San Francisco, CA, USA, 1998.