Density Scores for Cooperative Query Answering

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 inherent assumption in most systems is that only a combination that satisfies the global query completely is considered valid, i.e., it must obtain values for each required attribute. Furthermore, most systems strive for complete answers, i.e., they try to access all relevant sources. These requirements frequently lead to a system behavior that entails a high potential for user frustration: In many scenarios a partially incomplete answer is much more appreciated than no answer at all. Also, obtaining the data from all sources, which can be very costly, is often not necessary.

Based on this observation, we developed a cooperative query planning method. For a given query, the set of data sources is selected based on density scores obeying a user-defined price-constraint. The selection process naturally prefers complete answers, but can also cope with incomplete sources. An important result of this study is that, even under the assumption of a very simple cost model, finding an optimal solution is already NP complete. To encounter this complexity, we also include several heuristics.

1 Information Integration – Cooperating with the User

With the development of the Internet and efficient data transfer methods, federated database systems (FDBS) have become a popular means to integrate information from remote information sources [SL90]. This FDBS approach aims at querying multiple databases simultaneously through a single, homogeneous interface. For instance, many globally operating companies have set up a federation of their databases to allow queries covering information of all locations. To correctly and satisfyingly answer such queries, the participating databases must be well maintained. Decision makers will not accept query results of a low quality. Incomplete answers will result in incorrect aggregation, missing values may provoke wrong and possibly costly decisions.

Recently, the advantages of information integrating systems have attracted a new audience: private information consumers. This new type of users typically neither demands nor expects perfect and complete query results. They rely on simple read-only query interfaces and cooperative behavior of the query engine. Imagine a user searching for a particular book, its authors, publisher, and, if possible, some reviews. Such a user will most likely not find a missing publisher annoying, and would certainly prefer an answer without a review than no answer at all.

This new user profile greatly relaxes the demands on information integration if compared to the typical deployment scenario of federated database systems. Current research on FDBS systems has focused on providing correct and complete answers to any query. To obtain these answers, great effort must be put into algorithms that find complete query execution plans and great effort into their execution. Since data is randomly scattered over data sources, usually not only one, but all plans are executed, ensuring that the maximal complete query result is retrieved. This process is costly and unnecessary if perfect and complete results are not required. In the relaxed case, it is more important to find a *subset of all plans* such that an answer of sufficient quality is obtained in reasonable time and at a reasonable price. Furthermore, incomplete plans are preferred to empty answers.

To make use of these relaxed demands, we propose a *density measure* to guide planning in a mediator-based architecture for information integration. Density measures the *fullness* of the attributes exported by the local sources. An attribute with an actual value only in every second exported object will have a density score of 0.5. If an attribute is completely full, i.e., has a value for every object, it will receive the maximum density score of 1. An attribute that is simply not exported by that specific source will have 0 as a density score. With the help of this measure we find plans that produce the most dense results, given a user query and a predefined constraint on the number of sources that may be used.

Our approach incorporates various elements of cooperative behavior. First, it will always find *some answers*, even if complete answers are not possible. Second, it gives the user the possibility to *constrain the amount of work* performed during query execution, shielding him from unsatisfyingly long response times. We propose two algorithms that find good solutions under such constraints, where the quality of a plan is measured by the expected density of the result. Since density is attribute-specific, and not all attributes required in a user query are of equal importance, we thirdly allow the specification of an *attribute importance* that is used to tailor the quality estimation to the specific query needs.

Related work. The problem of integrating distributed and heterogeneous information from autonomous sources is addressed by many research projects, such as TSIMMIS [CGMH$^+$94], Information Manifold [LSK95], Garlic [CHS$^+$95], and others. Our approach and most of these projects share the adoption of the wrapper-mediator architecture as proposed by Wiederhold [Wie92]. Our logical model is similar to that of Yerneni et al. [YPAGM98].

However, none of the mentioned projects can cope with queries for which no complete answer exist, and none tries to select plans based on a density model or other quality criteria. Therefore, they all can be frustrating to use at times, when the inclusion of an further attribute suddenly renders a query unanswerable, without any hint given towards why that happened.

Our understanding of cooperative behavior is close to that described by Minker [Min98]. However, the seven different aspects reviewed there can not be easily adopted to the types of federated information systems that we consider. Regarding their classification, we concentrate on *user goals and preferences* and on *relaxed answers*. In contrast to our approach, Minkers work aims at *explaining* why certain queries have no answers, e.g., by giving intentional answers. On the other hand, we focus on actually producing reasonable answers. Gaasterland and Lobo describe a central database system that offers support for cooperative behavior in two aspects: First, users can annotate queries with preferences. Second, facts (and rules) are annotated with a *degree of confidence* [GL94]. Incomplete answers are not considered. Finally, De Michelis et al. coined the term "cooperative information system" for systems that are capable of cooperating with *other systems*, e.g., by facilitating *management of change* [DMDJ$^+$98]. This is orthogonal to our goal: Enabling cooperation with the user.

We regard density as an information quality criterion. Information quality considerations are not wide spread in the research community. Wang and Strong have empirically identified fifteen IQ criteria regarded by data consumers as the most important [WS96]. Some research projects have focused on integrating all these criteria into their model, for instance the Data Warehouse Quality project [JQJ98] or [NLF99]. Other projects have chosen only one or a few of these criteria for examination. Motro and Rakov, for instance, have studied the completeness and soundness criteria [MR98]. The completeness criterion subsumes our density criterion as defined below. However, the authors do not go beyond a general definition of the terms and only state possible application areas, whereas we actually assess and use the criterion for planning.

Structure of this paper. The following section explains our architecture and integration model. Section 3 defines attribute and source density and introduces our approach to determine and make use of the density scores for information integration. In Section 4 we present two planning algorithms that produce cooperative query execution plans by using the density scores. Section 5 concludes the paper.

2 The Planning Model

This section introduces our underlying, simple object model, the mediator architecture, and the source descriptions which we annotate with density scores.

2.1 Basic terms and assumptions

An *object* is our basic piece of information. An object has an identifier and a set of attributes. Attributes of an object can be stored in different sources as long as each source retains the identifier of the object. For example, in a book information system a book is an object. Its identifier is the ISBN, possible attributes are title, author, etc.

An *information source* represents a set of objects. These objects can be retrieved through some interface. An information integration system (IIS) covers one or more sources $(S_1, \dots, S_n)$. We make the following four assumptions to our model:

1. Global ID: We assume that each object in our IIS has a globally unique identifier. The identifier is consistent across sources, i.e., if two sources present an object with the same ID, then the IIS considers these objects to represent the same real-world entity. IDs are merely used to merge information; we do not require that they define functional dependencies, nor that each source stores at most one object per ID. IDs are called "merge attributes" in [YPAGM98].

   While this may seem like a strong assumption, it is true for many domains: Stocks have their name as a global ID, books have an ISBN, persons have a passport number etc. If no such ID is available we assume that one can be constructed. A person ID could be the combined name and address fields.

2. Global schema: We assume the global schema of the IIS to consist of only one relation. This relation contains the global ID and the union of all attributes delivered by sources. A *user query* is a selection of different attributes. A source is described as a view on the global schema which projects out any attribute not delivered by the source. However,

each source must deliver the ID attribute. We assume heterogeneity to be resolved inside the wrappers (see below).

Again, having only one global relation seems overly restrictive, but is in many cases a convenient and sufficient model. It is however not suitable if complex object relationships need to be modeled.

3. Cost model: We assume a certain cost being assigned to each query submitted to a source. This cost is independent from the order of execution and of the amount of data retrieved. Any information service charging per-query conforms to this model, and it is for instance also applicable if the number of queries the mediator can pass to a certain wrapper is constrained. We will discuss two variations of the model in Section 4.1.

4. Closed world: We assume that the union of all objects of the modeled sources is all the relevant data there is. We define $W := |\bigcup_{j=1}^{n}\{S_j\}|$ to denote the number of all objects covered by the IIS, i.e., the "size" of the entire system. Note that in general $W \neq \sum_i^n |S_i|$, since the sources may be extensionally overlapping[1].

These assumptions apply to such common applications as meta-search engines, stock information services, or book bargain finders.

2.2 Mediator Architecture

We follow the generic mediator wrapper architecture as proposed by Wiederhold [Wie92] (see Figure 1). Each data source is encapsulated by a software adapter, the *wrapper*. In this paper, we use the term wrapper or source to actually denote the view with which a wrapper is described. The *mediator* provides a global schema, accepts user queries against it, and plans query execution across the sources. If a query against a data source needs 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 a SQL query. The wrapper then submits the query to the source and receives the results. Results are finally reformatted to conform to the conventions of the mediator and returned to the mediator.

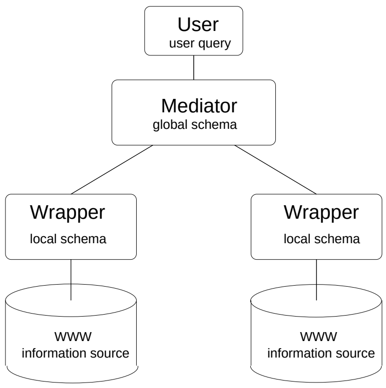

Figure 1: The general mediator architecture

Hence, each wrapper hides syntactical and technical heterogeneity in data sources from the mediator. This can e.g. require that an address, represented as a single attribute in

[1]The union-operator is duplicate removing.

the global schema, is constructed out of a set of different attributes which store the same information in the source. Other examples include the conversion of currencies, overcoming of structural mismatches, or the synthesis of global IDs.

2.3 Source descriptions and query planning

Many research projects have addressed the problem of answering user queries within a mediator architecture. TO this end, sources are typically described by some sort of views. Either concepts of the global schema are described as views on export schemata (global-as-view), such as in Pegasus [SAD+95] and IRO-DB [FGL+98], or each concept of an export schema is described as a view on the global schema (local-as-view), e.g., in Information Manifold [LRO96] and Infomaster [GKD97]. A combination of those two extremes is proposed in [Les98], based on query correspondence assertions. All these approaches have in common, that to answer a user query, source descriptions must be combined in a semantically correct manner which requires a laborious query planning process. Since many combinations may be correct, the complete answer to a user query is defined as the set of all correct plans.

Due to the fact that only complete plans are considered as correct, all these approaches will often fail to produce any answer, or will heavily restrain the number of objects returned. By stating *desirable but not necessary* variables in the query, the user inadvertently reduces the result size. Imagine an information source exporting book objects with their title, author, and price, and another source which contains reviews for a small number of books. Suppose a user asking for books of a certain author together with title, publisher and reviews. Using conventional planning methods, no answer will be obtained, since the publisher information is nowhere not available. Now suppose a different query asking the same query, but without requiring the publisher. Only such books will be returned for which the second source contains a review - in the extreme case, if source 2 happens to not review the searched author, an empty answer will be computed. In both cases user frustration will be high, although a little cooperation from the mediator could resolve such problems.

We avoid these difficulties in the following manner. First, for each wrapper, we annotate each exported attribute with a density score (see next section). Next, given a user query, we assign each attribute that is requested by the user query and that is not produced by a wrapper a density score of 0 for this wrapper. During query planning, each wrapper is considered to be able to answer any user query – though wrappers with missing attributes will be punished by being ranked low. Together with the user query, a cost-constraint can be specified which indirectly restricts the number of wrappers that can be accessed. The mediator will then choose wrappers such that the overall expected density of the result is as high as possible.

Query planning in our setting hence consists of selecting an optimal combination of wrappers. Note that the algorithms we present in Section 4 will always favor complete plans, i.e., plans that obtain values for all required attributes. But, in contrast to other query planning methods, if no complete plan exists they still produce valuable answers.

3 Density

Density is the "extent to which data are of sufficient breadth, depth, and scope for the task at hand" [WS96]. We interpret density as the "fullness" of the wrappers: Typical information sources have many missing values (`null`-values) in the attributes they provide – or put the other way, sources often provide attributes they do not completely cover. 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. The missing values result in incomplete results, i.e., tables with `null`-values. In traditional FDBS these `null`-values lead to dropping the object from the result. Density measures the percentage of non-`null`-values.

3.1 Density of a source

As each attribute in each wrapper is typically filled to different degrees, first we define density at attribute level.

Definition 1. *The* density *of an attribute a_{ik} of source S_k is*

$$den(a_{ik}) := \frac{|\{S_k | a_{ik} \neq \mathtt{null}\}|}{|S_k|}$$

The density of an attribute is thus the degree to which it is filled with non-`null`-values. It can also be interpreted as the probability that an arbitrary object has a non-`null`-value for that attribute. A density score of 1 is given to a completely filled attribute. For instance, we assume the identifier to have a density of 1 in each source. An attribute that is not exported by a source will have a density score of 0. All other attributes have values between 0 and 1.

To define the density of a source *with respect to a given query*, we extend the previous definition. We include the attributes of the user query uq in the following manner: An attribute exported by the source that is not queried for does not contribute to the source density. An attribute that is queried for, but not represented by the source, is getting assigned a density score of 0 and hence lowers the quality of the result. We define density of a source as the average density of all attributes in the query.

Definition 2. *The* density *of a source S_k wrt user query uq is*

$$den_{uq}(S_k) := \frac{\sum_{a_i \in uq} den(a_{ik})}{|uq|}$$

where $|uq|$ is the number of attributes in the user query.

Adding user weighting. Usually not all attributes of a query are equally important to a user. Consider the previous example of a user asking for books of a certain author along with publisher and review information. While the user will at least be partly satisfied if the publisher attribute is missing, he will certainly be dissatisfied if no title attribute is returned.

To express these differences in attribute importance, the user can give a weighting w_i for each attribute a_i of the query. The weighting can be given within any common range; we use the intuitive range of 1 (not important) to 10 (very important). We redefine density of a source accordingly:

Definition 3. *The* density *of a source S_k with user weights w_i is*

$$den_{uq}(S_k) := \frac{\sum_{a_i \in uq} den(a_{ik}) \cdot w_i}{\sum_{a_i \in uq} w_i}$$

Definitions 1 to 3 define density for single sources (and for attributes thereof). To use the measure for query planning we define the density of a merged result, i.e., for information combined from different sources.

3.2 Density of merged results

Before defining density of merged results, we explain what happens at object level and at attribute level when merging the results of two wrappers for a given query.

Combining objects. When merging the results of two queries to different sources into a single result, three different parts are generated: The first part consists of objects only contained in the first source. The next part consists of objects covered by both sources. This part is the result of a join over the two results using the identifier as the join predicate. Objects with the same identifiers are merged into one. What happens to the values of common attributes is explained in the next paragraph. Finally, the last part contains objects that are only contained in the second source. Figure 2 explains the three parts of the merge result.

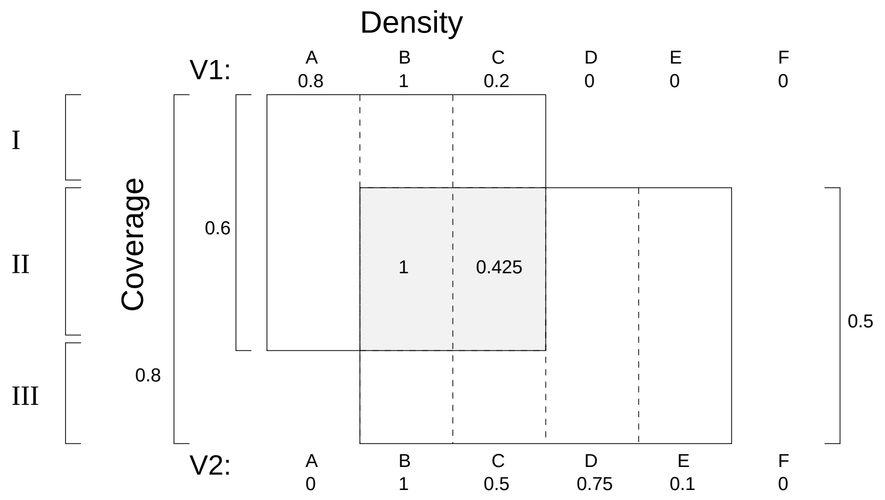

Figure 2: Merging two wrappers with differing attributes (overlap is shaded). Attributes that are exported by a wrapper can be recognized by a density score $\neq 0$

Combining attributes. In general, a result merged from two wrappers will contain (i) attributes covered by only one wrapper (attributes A, D and E in Figure 2), (ii) attributes that are covered by both wrappers (attributes B and C), and finally (iii) attributes that are not covered at all (attribute F). The first case is clear – when constructing the result, all available values are used in the merge result. `null`-values in the wrapper remain `null`-values in the result table. The third case is just as clear – the result will have an empty column.

The second case is more complicated. Both wrappers compete in filling the result table with attribute values for all objects which are present in both. Three cases apply for any object of the result table:

- Both wrappers have a `null`-value ⇒ The result has a `null`-value at that position.
- Exactly one wrapper provides data ⇒ The result uses that data at that position.

- Both wrappers provide data $\Rightarrow$ 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 [YM98]. Information quality metadata can greatly enhance resolution functions, for instance favoring the more recent value.

Calculating density. We now define how we compute the density of an attribute whose values are merged from to different wrappers. For attributes in the user query which are only¡ present in one wrapper, these computations are obsolete - the mediator simply picks the value from the appropriate wrapper. We first define the *coverage* of a source as

$$c(S_i) := \frac{|S_i|}{W} \tag{1}$$

where W is the size of the closed world (see Section 2.1).

Let $den(a_{ik})$ and $den(a_{ij})$ be the density scores of the attribute a_i in the sources S_k and S_j respectively, and let $a_{i(k,j)}$ be the density score of attribute a_i of the merged result. We distinguish the following five cases:

1. S_k and S_j are independent, i.e., the appearance of an object in S_k is independent of its appearance in S_j:

$$den(a_{i(k,j)}) = \frac{den(a_{ik}) \cdot c(S_k) + den(a_{ij}) \cdot c(S_j) - den(a_{ik}) \cdot den(a_{ij}) \cdot c(S_k) \cdot c(S_i)}{c(S_k) + c(S_j) - c(S_k) \cdot c(S_j)}$$

2. $S_k \supset S_j$, i.e., all objects in S_k also appear in S_j, although not necessarily with the same attributes or density scores:

$$den(a_{i(k,j)}) = \max[den(a_{ik}), \frac{den(a_{ik}) \cdot (c(S_k) - c(S_j)) + den(a_{ij}) \cdot c(S_j)}{c(S_k)}]$$

3. $S_k \subset S_j$: Analogous to case 2.

4. $S_k = S_j$, i.e., all objects in S_k also appear in S_j and vice versa, again not necessarily with the same attributes or density scores:

$$den(a_{i(k,j)}) = \max[den(a_{ik}), den(a_{ij})]$$

5. S_k and S_j are disjoint, i.e., S_k and S_j have no object in common. This case is not further discussed in this paper:

$$den(a_{i(k,j)}) = \frac{den(a_{ik}) \cdot c(S_k) + den(a_{ij}) \cdot c(S_j)}{c(S_k) + c(S_j)}$$

The calculation of the density of the overall merge result remains as in Definition 3 (average density). However, we now use the new attribute density scores depending on the dependency case:

$$den_{uq}(S_k, S_j) := \frac{\sum_{a_{i(k,j)} \in uq} den(a_{i(k,j)}) \cdot w_i}{\sum_{a_{i(k,j)} \in uq} w_i} \tag{2}$$

While these terms correctly calculate the density of two merged sources, they do not consider the transitivity of combining sources. Consider the simple case of three sources. Two of them are independent, while the third source is a subset of one of them. We have calculated the density score of the combination of the two independent sources. To now add the third source and calculate the new density score, we must somehow consider that it is a subset of a part of the current intermediate combination. Therefore, in the following section we will assume pairwise independence of all sources[2]. Other cases are addressed in Section 4.2.

4 Density-driven Planning

This section describes how we create query execution plans. A plan is a set of source queries performed by the 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.

4.1 Finding the best plans

We assume a cost model in which the mediator must pay some price for each query issued towards a source, which can often be found in existing applications. A mediator that often accesses distributed information sources will either pay a price for the information (if the information is offered by a third party) or have some other license agreement as to how many queries can be submitted during a given period. Within 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[3]. Parallel execution is not rewarded in this model since response time is not part of the model, but can still be performed whenever applicable. Also keep in mind that we assume independence of all sources.

We distinguish two cases within our cost model: The simple case assumes the same cost for each source and the general case assumes variable costs of the sources. For both cases we provide a heuristic.

Uniform cost. This first case assumes that 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 $\lfloor \frac{L}{C} \rfloor$.

Algorithm 1 describes our greedy heuristic to find the set of sources to query. After each querying of a source, we reassess the density scores of the remaining sources: The new

[2]Please note the algorithms work as well if we assume mutual disjointness of all sources.

[3]The dual problem is to find the least expensive plan to reach a certain density score. All results described below apply to this case analogously.

density score becomes the amount of *additional* density a source can contribute, given the set of sources already queried. Imagine two sources that each export the same single attribute, each with a density of 0.5. After querying one source, the density score of the second source is reduced to 0.25, since it only contributes that additional amount to the combined density score of 0.75. We define a function *getMaxSource()* which takes the current plan P and the remaining sources R returns a source from R, i.e., one that has not been queried yet, and that has the highest score of additional density to contribute to the current plan.

Algorithm 1 Uniform Cost High Density Planning

Input: Sources $S_1, \ldots S_n$, user query uq, user weighting uw, cost limit L
Output: query plan for uq

$R = S$;
for $i = 1$ **to** $\lfloor \frac{L}{C} \rfloor$ **do**
    $S =$ getmaxSource(P, R);
    $R = R \setminus S$
    $P = P + S$;
end for
execute P;

Algorithm 1 is of greedy nature, in that in each step it locally chooses the source that will contribute the most. The first step will simply choose the source with the highest density. Typically this will greatly reduce all other density scores. 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. The algorithm has a complexity of $O(n^2 \log n)$. It remains to be shown that the algorithm produces an optimal set of sources.

Variable cost. We now drop the condition of uniform cost. Unfortunately this turns the problem into a variation of the knapsack problem, which is NP-complete [Kar72]. As 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 slightly, since the benefit of a source is not fixed, but varies depending on the sources already queried.

Regarding the exponential nature of the knapsack problem, there is little use in trying to find an optimal solution for any reasonable number of sources. Approximation algorithms with optimality guarantees of $\frac{1}{1+\frac{1}{k}}$ and beyond have been presented for the knapsack problem [Sah75]. We use one of the most simple approximation algorithms which guarantees results within 50% of the optimum [GJ79] and adapt it to our problem.

Algorithm 2 extends the previous algorithm in two ways: First, to consider variable costs, it greedily selects sources by their density-cost ratio and not simply by their density. Thus, we define a function getMaxRatioSource(). Second, before executing sources in that greedy order, the algorithm compares the final expected density score with that of the single source with the highest density (not the highest ratio). The better of the two choices is executed. This technique avoids notorious worst cases and yields 50% optimality for the knapsack problem.

The complexity of Algorithm 2 also is $O(n^2 \log n)$, mainly due to reassessment for each chosen source. The results of the algorithm can be further improved for instance by "filling

Algorithm 2 Variable Cost High Density Planning

Input: Sources $S_1, \dots S_n$ with costs $c(S_i)$, user query uq, user weighting uw, cost limit L
Output: query plan for uq

$R = S$;
usedcost $= 0$;
for $i = 1$ **to** n **do**
  $S = \text{getMaxRatioSource}(P, R)$;
  $R = R \setminus S$;
  if usedcost$+c(S) > L$ **then**
    break;
  end if
  $P = P + S)$;
  usedcost $=$ usedcost $+ c(S)$;
end for
$S_{\max} =$ maximum density source;
if $den_{uq}(P) < den_{uq}(S_{\max})$ **then**
  $P = S_{\max}$;
end if
execute P;

in" inferior sources if the cost limit is not yet reached. While we still cannot guarantee optimal results, a very good performance of the algorithm can be expected.

4.2 Subset and superset relationships

By dropping the independence assumption and allowing subset and superset relationships between sources at the same time, finding good or optimal plans becomes more difficult: Consider the case of source S_1 being a subset of source S_2, i.e., all objects in S_1 also appear in S_2 with the same densities. When S_2 is chosen in a plan, S_1 should obviously be dropped from further consideration. However, if S_1 has already been chosen, and S_2 is queried now, having queried S_1 was a mistake.

Again we will consider the two cases of uniform and of variable cost, and discuss how the two algorithms perform.

Uniform cost. Any optimality result of the previous section for independent sources still holds in the presence of subset and superset relationships: Supersets and subsets have the same density scores, but the coverage score of a superset will be higher than that of the subset. Thus, the overall density of the superset is higher, and it will be chosen before any of its subsets. After reassessment, all subsets will have a density of 0 and will not be considered further[4].

Variable cost. For the case of variable costs we cannot use the simple approach of not considering subsets as proposed above. While in the previous case a subset was always considered worse than its superset and could thus be dropped, this is not the case anymore:

[4] A more simple solution would be to drop any subsets of any sources. For the sake of reusing Algorithm 1, we do not choose this alternative.

A subset still has less coverage than its superset, but it might also be less expensive. Thus, it's density/cost ratio might be higher and the source more desirable. The greedy approach described in Algorithm 2 might waste resources by querying a source of which a subset has already been queried. However, this waste is unlikely, since during reassessment the density of the superset is reduced by that of its previously chosen subsets.

5 Conclusions

The seamless integration of several information systems provides new functionality for information management, many of which are well described in research literature. In this work, we have described possibilities to improve IIS towards cooperative query answering. We thus enlarge the targeted group of IIS users to a much broader audience of non-professional users.

To this end, we have introduced the density measure, which quantifies and qualifies the amount of information returned by an information source. Using this measure we relax the condition of returning correct and thus perfect answers to the condition of returning as good (dense) an answer as possible. For a monetary cost model we introduced two algorithms that find such answers.

Future research will include cases where arbitrary relationships between sources can be considered at the same time. Also, we currently only optimize toward dense results, but not towards large results. Adding a *coverage value* to the density scores will enhance our algorithms toward that end.

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

[CGMH+94] 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.

[CHS+95] M.J. Carey, L.M. Haas, P.M. Schwarz, M. Arya, W.F. Cody, R. Fagin, M. Flickner, A.W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J.H. Williams, and E.L. Wimmers. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proceedings of the International Workshop on Research Issues in Data Engineering*, pages 161–172, March 1995.

[DMDJ+98] Georgio De Michelis, Eric Dubois, Matthias Jarke, Florian Matthes, John Mylopoulos, Michael P. Papazoglou, Klaus Pohl, Joachim Schmidt, Carson Woo, and Eric Yu. Cooperative information systems: a manifesto. In Michael P. Papazoglou and Gunter Schlageter, editors, *Cooperaztive Information Systems*, pages 315–363. Academic Press, San Diego, 1998.

[FGL+98] Peter Fankhauser, Georges Gardarin, M. Lopez, J. Munoz, and Anthony Tomasic. Experiences in federated databases: From iro-db to miro-web. In *24th Conference on Very Large Database Systems*, pages 655–658, New York, 1998.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, New York, 1979.

[GKD97] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An information integration system. In *ACM SIGMOD Int. Conference on Management of Data 1997*, pages 539–542, 1997.

[GL94] Terry Gaasterland and Jorge Lobo. Qualified answers that reflect users needs and preferences. In *20th Conference on Very Large Databases*, pages 309–320, Santiago de Chile, Chile, 1994.

[JQJ98] M.A. Jeusfeld, C. Quix, and M. Jarke. Design and analysis of quality information for data warehouses. In *Proc. of the 17th Int. Conf. on Conceptual Modeling (ER98)*, Singapore, November 1998.

[Kar72] 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.

[Les98] Ulf Leser. Combining heterogeneous data sources through query correspondence assertions. In *1st Workshop on Web Information and Data Management, in conjunction with CIKM'98*, pages 29–32, Washington, D.C., 1998.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the 22nd VLDB Conference*, Bombay, India, 1996.

[LSK95] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5(2):121–143, 1995.

[Min98] Jack Minker. An overview of cooperative answering in databases. In *3rd Conf. on Flexible Query Answering Systems*, pages 282–285, Roskilde, Denmark, 1998.

[MR98] Amihai Motro and Igor Rakov. Estimating the quality of databases. In *Proc. of the 3rd Int. Conf. on Felxible Query Answering Systems*, Roskilde, Denmark, May 1998. Springer Verlag.

[NLF99] Felix Naumann, Ulf Leser, and Johann Christoph Freytag. Quality-driven integration of heterogenous information systems. In *Proc. of the Int. Conf. on Very Large Databases*, Edinburgh, 1999.

[SAD+95] Ming-Chien Shan, Rafi Ahmed, Jim Davis, Weimin Du, and William Kent. Pegasus: A heterogeneous information management system. In Won Kim, editor, *Modern Database Systems*, pages 664–682. ACM Press, Addison-Wesley, Ney York, 1995.

[Sah75] S. Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM*, 22:115–124, 1975.

[SL90] Amit Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Survey*, 22(3):183–236, 1990.

[Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), 1992.

[WS96] Richard Y. Wang and Diane M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal on Management of Information Systems*, 12, 4:5–34, 1996.

[YM98] C. Yu and W. Meng. *Principles of database query processing for advanced applications.* Morgan Kaufmann, 1998.

[YPAGM98] Ramana Yerneni, Yannis Papakonstantinou, Serge Abiteboul, and Hector Garcia-Molina. Fusion queries over internet databases. In *Proc. of the International Conference on Extending Database Technology (EDBT)*, Valencia, Spain, March 1998.