Selecting Sources for Query Approximation with Bounded Resources

Hongjie Guo, Jianzhong Li(✉), and Hong Gao

Harbin Institute of Technology, Harbin 150001, Heilongjiang, China
hjguo@stu.hit.edu.cn, {lijzh,honggao}@hit.edu.cn

Abstract. In big data era, the Web contains a big amount of data, which is extracted from various sources. Exact query answering on large amounts of data sources is challenging for two main reasons. First, querying on big data sources is costly and even impossible. Second, due to the uneven data quality and overlaps of data sources, querying low-quality sources may return unexpected errors. Thus, it is critical to study approximate query problems on big data by accessing a bounded amount of the data sources. In this paper, we present an efficient method to select sources on big data for approximate querying. Our approach proposes a gain model for source selection by considering sources overlaps and data quality. Under the proposed model, we formalize the source selection problem into two optimization problems and prove their hardness. Due to the NP-hardness of problems, we present two approximate algorithms to solve the problems and devise a bitwise operation strategy to improve efficiency, along with rigorous theoretical guarantees on their performance. Experimental results on both real-world and synthetic data show high efficiency and scalability of our algorithms.

Keywords: Big data · Data quality · Source selection · Query approximation

1 Introduction

Traditional query processing mainly focuses on efficient computation of exact answers $Q(D)$ to a query Q in a dataset D. Nowadays, with the dramatic growth of useful information, dataset can be collected from various sources, i.e., websites, data markets, and enterprises. In applications with huge amounts of heterogeneous and autonomous data sources, exact query answering on big data is not only infeasible but also unnecessary due to the following reasons.

(1) Query answering is *costly*, even simple queries that require a single scan over the entire dataset cannot be answered within an acceptable time bound. Indeed, a linear-time query processing algorithm may take days on a dataset D of PB size [1].

(2) *Overlaps* among data sources are significant. Due to the autonomy of data sources, data sources are likely to contain overlap information, querying redundant sources may bring some gain, but with a higher extra cost.

W. Wu and Z. Zhang (Eds.): COCOA 2020, LNCS 12577, pp. 61–75, 2020.
https://doi.org/10.1007/978-3-030-64843-5_5

(3) Data sources are often *low-quality*. Even for domains such as flight and stock, which people consider to be highly reliable, a large amount of inconsistency has been observed in data collected from multiple sources [2], querying low-quality data sources can even deteriorate the quality of query results.

We next use a real-world example to illustrate these.

We consider the dataset D of all Computer Science books from an online bookstore aggregator. There are 894 bookstore sources, together providing 1265 Computer Science books. Each of these sources can be viewed as a relation: *Book (ISBN, Name, Author)*, describing the ISBN, name and author of Book, and each source identifies a book by its ISBN. Consider the following.

(1) A query Q_1 is to find the total number of books in D:

SELECT * FROM Bookstores

It is costly to answer Q_1, for not only the number of data sources is large, but also a data source contains a large volume of data. If we process the sources in decreasing order of their provide books, and query the total number of books after adding each new source. We can find that the largest source already provides 1096 books (86%), and we can obtain all 1265 books after querying 537 sources. In other words, after querying the first 537 sources, the rest of the sources do not bring any new gain [3].

(2) As another example, consider a query Q_2 to find the author lists of the book with ISBN = 201361205, query Q_2 is written as:

SELECT *Name* FROM Bookstores

WHERE *ISBN* = 201361205

We observed that D returns 5 different answers for Q_2, it shows that sources may have quality problems and can provide quite different answers for a query. In fact, in this dataset, each book has 1 to 23 different provided author lists.

These examples show that it is inappropriate to accessing the whole dataset when querying data from a large number of sources, which motivates us to select proper data sources before querying. Due to the importance, data source selection draws attention recently. [3] selects a set of sources for data fusion, and maximizes the profit by optimizing the gain and cost of integrating sources. However, it does not consider data self-conflicting and incomplete, and could hardly scale to big data sources for high time complexity of algorithm. [12] proposes a probabilistic coverage model to evaluate the quality of data and consider overlaps information. However, this method requires some prior statistics truth probability of each value, furthermore, this paper adopts a simple greedy method to solve the problem which is not optimal.

Based on the above discussion, in this paper, given an upper bound on the amount of data that can be used, we propose efficient source selection methods for approximate query answering. We propose a gain model for source selection by measuring the sources according to their coverage, overlaps, quality and cost. Considering these measures, we formulate the data source selection problem as two optimization problems from different aspects, both of which are proven to be NP-hard. Due to the hardness, we present two approximate algorithms to solve the optimization problems. These algorithms are proved can obtain the best

possible approximate factor, furthermore, the running time of these methods are linear or polynomial in source number for the two propose problems, respectively. To improve the efficiency and scalability, we prestore the coverage information of each source offline and eliminate overlaps between sources online using bitwise operation, in this way, we can greatly accelerate the source selection algorithms.

In this paper, we make the following contributions.

First, we propose the gain model for data source selection from big autonomous data sources. This model take into consideration data quality and sources overlaps.

Second, under the proposed gain model, We formalize two optimization goals for source selection, called BNMG and BCMG, respectively. We show these two problems both are NP-hard.

Third, We present a greedy approximation algorithm for BNMG. We show that the greedy heuristic algorithm for BCMG has an unbounded approximation factor, and present a modified greedy algorithm to achieve a constant approximation factor for BCMG. A bitwise operation strategy for our algorithms is proposed to achieve better performance.

Finally, we conduct experiments on real-life and synthetic data to verify the effectiveness, efficiency and scalability of our proposed algorithms.

2 Problem Definition

This section first formally defines the basic notions used in the paper, then formulates the source selection problems, and then analyzes the complexity of these problems.

2.1 Basic Notions and Quality Metric

Definition 1 *(Data source). A dataset D is specified by a relational schema $\mathcal{R}$, which consists of a collection of relation schemas $(R_1, \cdots, R_m)$. Each relation schema R_i is defined over a set of attributes. A dataset consists of a set of data sources $\mathbb{S} = \{S_i | 1 \leq i \leq m\}$, where each source S_i includes a schema R_i. We assume that the schemas of sources have been mapped by existing schema mapping techniques.*

Definition 2 *(Data item). Consider a data item domain $\mathcal{DI}$. A data item $DI \in \mathcal{DI}$ represents a particular aspect of a real-world entity in a domain, such as the name of a book, and each data item should be identified by a key. For each data item $DI \in \mathcal{DI}$, a source $S_i \in \mathbb{S}$ can (but not necessarily) provide a value, and $\mathcal{DI}_i$ denotes that a set of data items provided by S_i.*

Definition 3 *(Functional dependency (FD)). An FD φ: $[A_1, \cdots, A_l] \rightarrow [B]$, where A_i and B are attributes in relational table. The semantic of φ is that any two tuples are equal on the left-hand side attribute of φ, they should also be equal on the right-hand side, otherwise, we say such tuples violate the FD [4].*

Definition 4 *(Claim). Let $\mathcal{S}$ be a set of sources and $\mathcal{C}$ a set of claims, each of which is triple $< S, k, v >$, meaning that the source, S, claims a value, v, on a data item with key, k. Sources often provide claims in the form of tuples.*

For example, in Table 1 there are two sources, s_1 and s_2, providing 3 claim-tuples. Note that a source can provide conflicting claims for a data item, for instance, consider data item: *ISDN = 02013.Name*, s_1 claims that the name of *ISDN = 02013* are *Java* and *C++*, respectively. However, only one of the conflicting values is true. s_2 also miss the value of the author of the book with *ISDN = 02014*.

Table 1. Claims tuple of books.

$Source_{id}$	$Tuple_{id}$	ISDN	Name	Author
s_1	t_1	02013	Java	Robert
s_1	t_2	02013	C++	Robert
s_2	t_3	02014	Analysis of Algorithms	

Next, we consider quality metrics. Selecting sources should consider three sides. First, we wish to select a source that has a high *coverage* and a low *overlap*: such a source would return more answers and contribute more new answers than another source with the same coverage. Second, the answers returned by a high-quality source are high *reliability*. Third, a source with low *cost* will yield better performance. We next formally define these measures considered in source selection.

Definition 5 *(Coverage). We define the* coverage *of source S as the number of its provided data items, denoted by $V(S)$. Formally,*

$$V(S_i) = |\mathcal{DI}_i|, 1 \leq i \leq m \tag{1}$$

For source set $\mathcal{S}$ (a subset of source $\mathbb{S}$), we have

$$V(\mathcal{S}) = \cup_{S_i \in \mathcal{S}} V(S_i) \tag{2}$$

coverage of the source S reflects the expected number of answers returned by S, and *coverage* of the source set $\mathcal{S}$ represents the total distinct data items provided by sources, which has already eliminated the *overlap* information.

A source may provide self-conflicting or incomplete data, which means that the source has low reliability, querying low reliability may lead to an unexpectedly bad result. Thus, it is non-trivial to select sources according to their reliability, there can be many different ways to measure source reliability, in this paper, we measure it as the maximum correct number of claims provided by source, called the *reliability* of the source. The *reliability* of source S is denoted by $R(S)$.

In Table 1, s_1 claims two different values for the name of book, and no more than one of these can be correct. The upper bound of correct claims number provided by source S over key k is

$$u_{s,k} = \max_{v}(N_{s,k,v}) \tag{3}$$

where $N_{s,k,v}$ is the number of claims provided by S for k with a value v.

Definition 6 *(Reliability). The reliability of source S according to the upper bounds as*

$$R(S) = \sum_{k} u_{s,k} \tag{4}$$

For source set $\mathcal{S}$, we have

$$R(\mathcal{S}) = \sum_{S_i \in \mathcal{S}} R(S_i) \tag{5}$$

Collecting sources for querying come with a *cost*. First, many data sources charge for their data. Second, collecting and integrating them requires resources and time.

Definition 7 *(Cost). We define the cost of source S as the total number of its provided claims, denoted by $C(S)$.*

$$C(S) = \sum_{k,v}(N_{s,k,v}) \tag{6}$$

Similarly, for source set $\mathcal{S}$, we have

$$C(\mathcal{S}) = \sum_{S_i \in \mathcal{S}} C(S_i) \tag{7}$$

Definition 8 *(Gain). Now we define the gain model of source selection.*

$$G(\mathcal{S}) = \alpha V(\mathcal{S}) + (1 - \alpha)R(\mathcal{S}) \tag{8}$$

where $\alpha \in [0, 1]$ is a parameter controlling how much coverage and reliability have to be taken into account.

2.2 Problems

It is impractical to maximize the *gain* while minimizing the *cost*. Thus, we define the following two constrained optimization problems to select sources.

In some scenarios, a query system gives an upper **B**ound on the **N**umber of data sources can be accessed, and the optimization goal is to **M**aximize the **G**ain, we call this problem as BNMG.

Definition 9 *(BNMG). Given a source set* $\mathbb{S}$ *and a positive integer* K*, the BNMG problem is to find a subset* $\mathcal{S}$ *of* $\mathbb{S}$*, such that* $|\mathcal{S}| \leq K$*, and* $G(\mathcal{S})$ *is maximized.*

In some scenarios, a query system gives an upper **B**ound on the **C**ost as the constraint, and wishes to obtain the **M**aximum **G**ain, we define this problem as BCMG.

Definition 10 *(BCMG). Given a source set* $\mathbb{S}$ *and* τ_c *be a budget on cost, the BCMG problem is to find a subset* $\mathcal{S}$ *of* $\mathbb{S}$*, such that maximizes* $G(\mathcal{S})$ *under* $C(\mathcal{S}) \leq \tau_c$*.*

2.3 Complexity Results

Theorem 1. *The gain function of formulation (8) is non-negative, monotone and submodular.*

Proof. ***non-negative****. Obviously.*
monotone*. For* $x \in \mathbb{S} - \mathcal{S}$*, if* $G(\mathcal{S} \cup x) \geq G(\mathcal{S})$*, the gain model is monotone.*

$$\alpha V(\mathcal{S} \cup x) \geq \alpha V(\mathcal{S}), obviously \tag{9}$$

$$(1-\alpha)R(\mathcal{S} \cup x) = (1-\alpha)R(\mathcal{S}) + (1-\alpha)R(x) \geq (1-\alpha)R(\mathcal{S}) \tag{10}$$

According to Eqs. (9) and (10), we get

$$G(\mathcal{S} \cup x) = \alpha V(\mathcal{S} \cup x) + (1-\alpha)R(\mathcal{S} \cup x) \geq \alpha V(\mathcal{S}) + (1-\alpha)R(\mathcal{S}) = G(\mathcal{S}) \tag{11}$$

Submodular*. For* $\mathcal{R} \subset \mathcal{S}$ *and* $x \in \mathbb{S} - \mathcal{S}$*, if* $G(\mathcal{S} \cup x) - G(\mathcal{S}) \leq G(\mathcal{R} \cup x) - G(\mathcal{R})$*, the gain model is submodular.*

$$\begin{aligned} V(\mathcal{S} \cup x) - V(\mathcal{S}) &= V(\mathcal{S}) + V(x) - V(\mathcal{S} \cap x) - V(\mathcal{S}) \\ &= V(x) - V(\mathcal{S} \cap x) \end{aligned} \tag{12}$$

Similarly,

$$V(\mathcal{R} \cup x) - V(\mathcal{R}) = V(x) - V(\mathcal{R} \cap x) \tag{13}$$

Since, $\mathcal{R} \subset \mathcal{S}$*, then* $V(\mathcal{S} \cap x) \geq V(\mathcal{R} \cap x)$*. Hence*

$$\alpha V(\mathcal{S} \cup x) - \alpha V(\mathcal{S}) \leq \alpha V(\mathcal{R} \cup x) - \alpha V(\mathcal{R}) \tag{14}$$

And

$$\begin{aligned} (1-\alpha)R(\mathcal{S} \cup x) - (1-\alpha)R(\mathcal{S}) &= (1-\alpha)R(\mathcal{S}) + (1-\alpha)R(x) - (1-\alpha)R(\mathcal{S}) \\ &= (1-\alpha)R(x) \end{aligned} \tag{15}$$

Similarly, $(1-\alpha)R(\mathcal{R} \cup x) - (1-\alpha)R(\mathcal{R}) = (1-\alpha)R(x)$*, we have*

$$(1-\alpha)R(\mathcal{S} \cup x) - (1-\alpha)R(\mathcal{S}) = (1-\alpha)R(\mathcal{R} \cup x) - (1-\alpha)R(\mathcal{R}) \tag{16}$$

Combining Eqs. (14) and (16), we get

$$G(\mathcal{S} \cup x) - G(\mathcal{S}) \leq G(\mathcal{R} \cup x) - G(\mathcal{R}) \tag{17}$$

Theorem 2. *Both BNMG and BCMG are NP-hard problems.*

Proof. For a submodular function f, if f only takes non-negative value, and is monotone. Finding a K-element set $\mathcal{S}$ for which $f(\mathcal{S})$ is maximized is an NP-hard optimization problem [5,6]. For BNMG problem, function f is the *gain* model, thus, BNMG is NP-hard.

The BCMG problem is an instance of the Budgeted Maximum Coverage Problem (BMC) that is proven to be NP-hard [7]. Given an instance of BMC: A collection of sets $\mathbb{S} = \{S_1, S_2, \cdots, S_m\}$ with associated costs $\{C_i\}_{i=1}^{m}$ is defined over a domain of elements $X = \{x_1, x_2, \cdots, x_n\}$ with associated equivalent-weights. The goal is to find a collection of sets $\mathcal{S} \subseteq \mathbb{S}$, such that the total cost of elements in $\mathcal{S}$ does not exceed a given budget L, and the total weight of elements covered by $\mathcal{S}$ is maximized. BCMG can be captured by the BMC problem in the following way: 1) the sets in BMC represent the sources in $\mathbb{S}$ of BCMG; 2) the elements in BMC represent the data items in BCMG; 3) the parameter α of the *gain* model in BCMG is equal to 1. Since the reduction could be accomplished in polynomial time, BCMG is an NP-hard problem.

3 Algorithm for Source Selection

Due to the NP-hardness of BNMG and BCMG, in this section, firstly, we devise a greedy approximation algorithm for BNMG and analyze the complexity and approximation ratio (Sect. 3.1). Then, we show that a greedy strategy is insufficient for solving the BCMG problem. Indeed, it can get arbitrary bad results. Thus, we generate a modified greedy algorithm using the enumeration technique for BCMG, and demonstrate that such algorithm has the best possible approximation factor (Sect. 3.2). We devise a bitwise operation strategy to accelerate the running of proposed algorithms (Sect. 3.3).

3.1 Algorithm for BNMG

For a submodular and nondecreasing function f, f satisfies a natural "diminishing returns" property: The marginal gain from adding a source to a set of sources $\mathcal{S}$ is at least as high as the marginal gain from adding the same source to a superset of $\mathcal{S}$. Here, the marginal gain ($G(\mathcal{S} \cup S_i) - G(\mathcal{S})$ in this algorithm) is the difference between the gain after and before selecting the new source. Such problem is well-solved by a simple greedy algorithm, denoted by Greedy (shown in Algorithm 1), that selects K sources by iteratively picking the source that provides the largest marginal gain (line 6).

Algorithm 1. Greedy

Input: $\mathbb{S}$, K
Output: a subset $\mathcal{S}$ of $\mathbb{S}$ with $|\mathcal{S}| \leq K$
1: Initialize $\mathcal{S} \leftarrow \emptyset$
2: **while** $|\mathcal{S}| < K$ **do**
3: **for all** $S_i \in \mathbb{S}$ **do**
4: $G(\mathcal{S} \cup S_i) \leftarrow \text{CompGain}(\mathcal{S} \cup S_i)$;
5: **end for**
6: $S_{opt} \leftarrow \arg\max_{S_i \in \mathbb{S}} G(\mathcal{S} \cup S_i) - G(\mathcal{S})$;
7: $\mathcal{S} \leftarrow \mathcal{S} \cup S_{opt}$;
8: $\mathbb{S} \leftarrow \mathbb{S} \backslash S_{opt}$;
9: **end while**

Time Complexity Analysis. The time complexity of Algorithm 1 is determined by the complexity to compute the *gain* of $(\mathcal{S} \cup S_i)$, this complexity is $O(n)$, n is the maximal number of data items in S_i. Clearly, the complexity of Algorithm 1 is $O(K * n * m)$, where K is the number of selected sources, and m is the number of sources in $\mathbb{S}$.

Theorem 3. *Algorithm 1 is a* $(1 - 1/e)$ *– approximation algorithm, where e is the base of the natural logarithm.*

Proof. The greedy algorithm has $(1-1/e)$ approximation ratio for a submodular and monotone function with a cardinality constraint [6].

3.2 Algorithm for BCMG

The greedy heuristic algorithm that picks at each step a source maximizing the ratio $\frac{G(\mathcal{S} \cup S_i) - G(\mathcal{S})}{C(S_i)}$ has an unbounded approximation factor. Namely, the worst case behavior of this algorithm might be very far from the optimal solution. In Table 2 for example, two sources S_1 and S_2 are subjected to an FD: $key \rightarrow value$. According to our problem definition, S_1 has $V(S_1) = 1$, $R(S_1) = 1$, $C(S_1) = 1$; S_2 has $V(S_2) = p$, $R(S_2) = p$, $C(S_2) = p + 1$. Let $\mathbb{S} = \{S_1, S_2\}$, $\alpha = 0.5$, and the budget of cost $\tau_c = p + 1$. The optimal solution contains the source S_2 and has *gain* p, while the solution picked by the greedy heuristic contains the source S_1 and has *gain* 1. The approximation factor of this instance is p, and is therefore unbounded (since p is not a constant).

Table 2. Two sources for an example

(a) Source S_1

id	key	value
t_1	1	1

(b) Source S_2

id	key	value
t_1	1	1
...	...	...
t_p	p	p
t_{p+1}	p	m

We modify the greedy heuristic using the enumeration technique, so as to achieve a constant approximation factor for the BCMG problem. The main idea is to apply the partial enumeration technique [8] before calling greedy algorithm, denoted by EnumGreedy (shown in Algorithm 2). Let l be a fixed integer. Firstly, we enumerate all subsets of $\mathbb{S}$ of cardinality less than l which have cost at most τ_c, and select the subset that has the maximal *gain* as the candidate solution (line 2). Then, we consider all subsets of $\mathbb{S}$ of cardinality l which have cost at most τ_c, and we complete each subset to a candidate solution using the greedy heuristic (line 3–17). The algorithm outputs the candidate solution having the greatest *gain* (line 18–22). *Time Complexity Analysis.* The running time of Algorithm 2 is

Algorithm 2. EnumGreedy

Input: $\mathbb{S}$, τ_c, l
Output: a subset $\mathcal{S}$ of $\mathbb{S}$ with $C(\mathcal{S}) \leq \tau_c$
1: Initialize $\mathcal{S} \leftarrow \emptyset$, $\overline{\mathcal{S}'} \leftarrow \emptyset$, $\overline{\mathcal{S}''} \leftarrow \emptyset$
2: $\overline{\mathcal{S}'} \leftarrow \arg\max_{\overline{\mathcal{S}'} \subseteq \mathbb{S}} \{G(\overline{\mathcal{S}'}) |\ C(\overline{\mathcal{S}'}) \leq \tau_c, |\overline{\mathcal{S}'}| < l\}$
3: **for all** $\mathcal{S}'' \subseteq \mathbb{S}$, $|\mathcal{S}''| = l$, $C(\mathcal{S}'') \leq \tau_c$ **do**
4: $\quad \mathbb{S} \leftarrow \mathbb{S} \backslash \mathcal{S}''$
5: $\quad$ **for all** $S_i \in \mathbb{S}$ **do**
6: $\quad\quad G(\mathcal{S}'' \cup S_i) \leftarrow \text{CompGain}(\mathcal{S}'' \cup S_i)$;
7: $\quad\quad C(S_i) \leftarrow \text{CompCost}(S_i)$;
8: $\quad$ **end for**
9: $\quad S_{opt} \leftarrow \arg\max_{S_i} \frac{G(\mathcal{S}'' \cup S_i) - G(\mathcal{S}'')}{C(S_i)}$;
10: $\quad$ **if** $C(\mathcal{S}'') + C(S_i) \leq \tau_c$ **then**
11: $\quad\quad \mathcal{S}'' \leftarrow \mathcal{S}'' \cup S_{opt}$;
12: $\quad\quad \mathbb{S} \leftarrow \mathbb{S} \backslash S_{opt}$;
13: $\quad$ **end if**
14: $\quad$ **if** $G(\mathcal{S}'') > G(\overline{\mathcal{S}''})$ **then**
15: $\quad\quad \overline{\mathcal{S}''} \leftarrow \mathcal{S}''$;
16: $\quad$ **end if**
17: **end for**
18: **if** $G(\overline{\mathcal{S}'}) > G(\overline{\mathcal{S}''})$ **then**
19: $\quad \mathcal{S} \leftarrow \overline{\mathcal{S}'}$;
20: **else**
21: $\quad \mathcal{S} \leftarrow \overline{\mathcal{S}''}$;
22: **end if**

$O((n \cdot m)^{(l-1)})$ executions of enumeration and $O((n \cdot m)^{2l})$ executions of greedy, where m is the number of sources, n is the maximal number of data items in S_i. Therefore, for every fixed l, the running time is polynomial in $n \cdot m$.

Discussion. When $l = 1$, Algorithm 2 is actually a simple greedy method which has an unbounded approximation factor as previously mentioned. When $l = 2$, Algorithm 2 finds a collection of sources according to the greedy heuristic as the first candidate for solution. The second candidate is a single set in $\mathbb{S}$ for which *gain* is maximized. The algorithm outputs the candidate solution having the maximum *gain*.

Theorem 4. *For $l = 2$, Algorithm 2 achieves an approximation factor of $\frac{1}{2}(1 - \frac{1}{e})$ for the BCMG problem. For $l \geq 3$, Algorithm 2 achieves a $(1 - \frac{1}{e})$ approximation ratio for the BCMG, and this approximation factor is the best possible.*

Proof. The proof is by generalized the proof of approximation factor for the BMC problem, presented in [7]. The detail is omitted due to space limitation.

3.3 Improvement for Algorithms

The time complexities of proposed algorithms are determined by the complexity to compute the *gain*. In fact, the time complexities are dominated by the computing of *coverage* since *reliability* and *cost* can be computed in constant time. To reduce the computation time, we transform the process of computing *coverage* into building bit vectors and conducting bitwise *or* operation between them so that we can compute *coverage* without accessing original data. We build a bit vector $\boldsymbol{B}(S_i)$ for each S_i offline and use them to compute the coverage of sources online.

Constructing a bit vector for each source in an offline phase. Given $\mathbb{S}$, we consider all data items of $\mathbb{S}$ as bit vector space. For a source S_i, we maintain a bit vector $\boldsymbol{B}(S_i) = \{(b_1, b_2, \cdots, b_M) | b_i \in \{0, 1\}\}$, where M is the total number of data items provided by $\mathbb{S}$, b_j equals 1 if S_i contains j-th data item of $\mathbb{S}$. The bit vector building algorithm for each source S_i is described in Algorithm 3.

Algorithm 3. Bit Vector Building

Input: $\mathbb{S}$, Σ (FDs set)
Output: $\{\boldsymbol{B}(S_i) | S_i \in \mathbb{S}, 0 \leq i \leq m\}$

```
Initialize B(S_i) ← ∅
for all S_i ∈ 𝕊 do
    for all φ ∈ Σ do
        for all Left-hand side A_j of φ do
            if S_i contains A_j then
                b_j = 1;
            else
                b_j = 0;
            end if
            Add b_j to B(S_i);
        end for
    end for
end for
```

Then, the *coverage* of S_i is the number of 1 in $\boldsymbol{B}(S_i)$, denoted as $\boldsymbol{B}(S_i).cardinality$.

$$V(S_i) = \boldsymbol{B}(S_i).cardinality = \sum_{1}^{M} b_j \tag{18}$$

Computing source *coverage* online. Given a source set $\mathcal{S}$, the *coverage* of $\mathcal{S}$ can be easily computed by bitwise *or* operation:

$$V(\mathcal{S}) = \cup_{S_i \in \mathcal{S}} V(S_i) = (\vee_{S_i \in \mathcal{S}} \boldsymbol{B}(S_i)).cardinality \tag{19}$$

where $\vee$ is the bitwise *or* operation.

Time Complexity Analysis. The expected time of Algorithm 3 is $O(m * |\Sigma| * n_{avg})$, where $n_{avg} = avg_{0 \leq i \leq m} |S_i|$. Although the complexity is quite high, such bit vectors are computed offline. Hence it will not affect the performance of the algorithm.

We denote the improvement greedy algorithm for BNMG problem which combines the bit vector operation as BitGreedy, and BitEnumGreedy for BCMG problem similarly.

4 Experimental Results

In this section, we study the proposed algorithms experimentally. The goals are to investigate (1) the comparison of performance between Greedy and BitGreedy for BNMG problem, as well as EnumGreedy and BitEnumGreedy for BCMG problem, and (2) how our algorithms perform in terms of efficiency and scalability.

4.1 Experiment Setup

We conducted our comparison experiments over a real-world dataset: *Book* [3]. In addition, to investigate the efficiency and scalability of our algorithm, we evaluated the performance of BitGreedy and BitEnumGreedy on synthetic datasets that yielded more sources and more tuples.

The *Book* dataset contains 894 data sources. Information on Computer Science books was collected from online bookstore aggregator AbeBooks.com, there are two FDs between the attributes: $ISDN \rightarrow Name$ and $ISDN \rightarrow Author$.

The *Synthetic Data* is synthetic data sets with various data source number and data size. We used 10 attributes $A1 - A10$ and 8 FDs: $A1 \rightarrow A8$, $A1 \rightarrow A9$, $A1 \rightarrow A10$, $A2 \rightarrow A6$, $A2 \rightarrow A7$, $A3 \rightarrow A6$, $A3 \rightarrow A7$, $[A4, A5] \rightarrow A8$. Each data source randomly chose an attribute with 20% probability, and each source contains at least one of the FDs, and the size of each data source is a random number in the range of [2000, 10000].

In practical situations, due to the enormous number and volume of sources, Algorithm 2 needs to consume considerable time even when $l = 3$ to guarantee the best approximation ratio. Thus, in this paper, we set $l = 2$.

In this paper, we focus more on selecting data sources with high coverage, thus, we set $\alpha = 0.9$. Users can set different values for α according to their preferences.

All experiments are implemented in Java and executed on a PC with Windows 7, a 16 GB of RAM and a 3.6 GHz Intel i7-7700U CPU.

4.2 Comparison

For BNMG problem, we firstly compare the effectiveness of Greedy and BitGreedy with #Bounded Source (K in problem definition) varying from 1 to 10 on *Book* dataset, shown in Fig. 1(a) and Fig. 1(b), then we vary #Bounded Source from 10 to 100 on *Synthetic Data* with #Source = 100, #Tuple = 2000, shown in Fig. 1(c) and Fig. 1(d).

For BCMG problem, we compare EnumGreedy and BitEnumGreedy with Bounded cost varying from 5k to 50k on *Book* dataset and Bounded cost varying from $5*10^4$ to $5*10^5$ on *Synthetic Data*, The results are shown in Fig. 1(e)–1(h).

We have the following observations. (1) For BNMG problem, both on real-world data set and Synthetic data. Greedy and BitGreedy achieve the same Gain, and with the increase of #Bounded sources, the runtime of Greedy is linear to #Bounded sources, while the runtime of BitGreedy grows much slowly with #Bounded and outperforms Greedy significantly. (2) BCMG problem get a similar result. Whether the data set is *Book* or *Synthetic Data*, compare to EunmGreedy, BitEumGreedy achieves hundreds of speed up while not sacrifice the Gain. (3) Due to the higher time complexity, the algorithm for BCMG problem requires much runtime than that of BNMG, it also signifies that algorithm EnumGreedy, which without improvement strategy, for BCMG problem can not apply to real-time query systems.

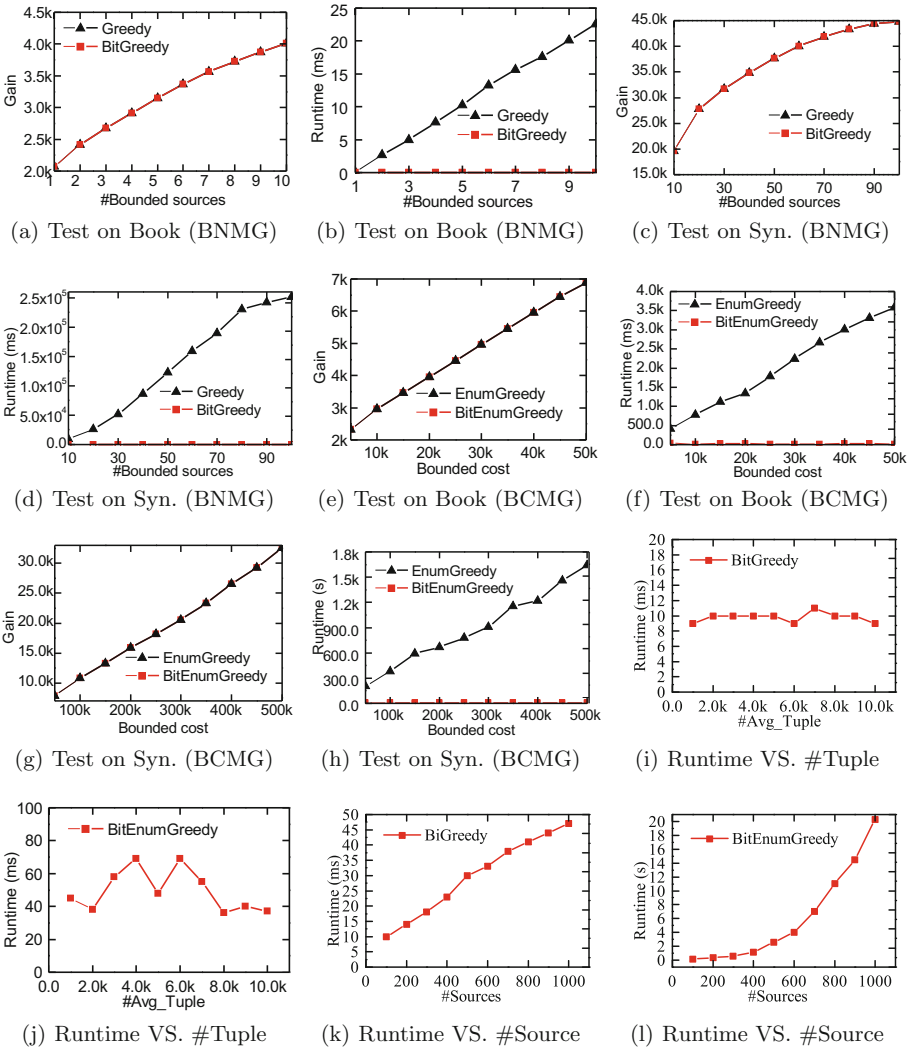
4.3 Efficiency and Scalability

To further test how the number of sources and source size affects efficiency and scalability, we conduct experiments on synthetic datasets. (1) Fig. 1(i) and Fig. 1(j) report the runtimes of both algorithms with varying the data size. We observe that the runtimes of BitGreedy and EnumBitGreedy are very stable, it shows that the high efficiency and stability of our method when the volume of data source grows. Figure 1(k) and Fig. 1(l) plot the running time of BitGreedy and EnumBitGreedy respectively, as we vary the number of data sources from 100 to 1000. These show that the runtimes of both BitGreedy and EnumBitGreedy increase nearly linearly. BitGreedy costs 47ms when the source number reaches 1000, and EnumBitGreedy finishes in 20316ms when the source number is 1000, showing the great scalability of our methods.

Summary. (1) The sources selected by BitGreedy and BitEnumGreedy are same as the selections of Greedy and EnumGreedy. (2) The algorithms using bitwise operation outperform original methods both on efficiency and scalability significantly. (3) The effectiveness of BitGreedy and BitEnumGreedy are insensitive to the source size. (4) Our algorithms scale well on both the data size and the number of sources.

5 Related Work

Source selection [3,9–13] has been recently studied. [3] selects a set of sources for data fusion, it efficiently estimates fusion accuracy, and maximizes the profit by



(a) Test on Book (BNMG) (b) Test on Book (BNMG) (c) Test on Syn. (BNMG)

(d) Test on Syn. (BNMG) (e) Test on Book (BCMG) (f) Test on Book (BCMG)

(g) Test on Syn. (BCMG) (h) Test on Syn. (BCMG) (i) Runtime VS. #Tuple

(j) Runtime VS. #Tuple (k) Runtime VS. #Source (l) Runtime VS. #Source

Fig. 1. Experimental results

optimizing the gain and cost of integrating sources. However, it does not consider data self-conflicting and incomplete, and could hardly scale to big data sources. [9] studies online ordering sources by estimating overlaps of sources, but this method requires some prior statistics and neglects data quality. [10] only takes freshness as quality metric without a comprehensive consideration for the quality of data sources such as functional dependency, completeness and the required resources. [11] focuses on finding sources relevant to a given query and does not take overlap into consideration. [12] proposes a probabilistic coverage model to evaluate the quality of data and consider overlaps information. However, this

method requires some prior statistics truth probability of each value, furthermore, this paper adopts a simple greedy method to solve the problem which is not optimal. [13] selects sources for data inconsistency detection and does not take quality and cost into consideration.

Fan et al. [1,14–18] proposed a series of work with respect to approximate query processing with bounded resources, which can answer a specific class of queries by accessing only a subset of the whole dataset with a bounded number of tuples. Unlike our study, these works access a portion of data from each data source instead of selecting sources by considering their coverage, overlap and quality.

6 Conclusion

This paper studies source selection problem for query approximation taking efficiency and effectiveness into consideration. We first propose a gain model to evaluate the coverage, reliability and cost of data source and we formulate source selection problem to two problems, which are both proven to be NP-hard. Then we develop a greedy algorithm for bounded source number problem and a modified greedy for bounded cost problem and show their approximations, both algorithms come with rigorous theoretical guarantees on their performance. Finally, we propose an efficient bitwise operation strategy to further improve efficiency. Experimental results on both the real-world and synthetic datasets show our methods can select sources efficiently and can scale to datasets with up to thousands of data sources.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under grants 61732003, 61832003, 61972110 and U1811461.

References

1. Fan, W., Geerts, F., Neven, F.: Making queries tractable on big data with preprocessing. PVLDB **6**(9), 685–696 (2013)
2. Li, X., Dong, X.L., Lyons, K., Meng, W., Srivastava, D.: Truth finding on the deep web: is the problem solved? VLDB **6**(2), 97–108 (2012)
3. Dong, X.L., Saha, B., Srivastava, D.: Less is more: selecting sources wisely for integration. In: VLDB, vol. 6, pp. 37–48. VLDB Endowment (2012)
4. Codd, E.F.: Relational completeness of data base sublanguages. In: Courant Computer Science Symposia, vol. 6, pp. 65–98. Data Base Systems (1972)
5. Cornuejols, G., Fisher, M.L., Nemhauser, G.L.: Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. Manag. Sci. **23**(8), 789–810 (1977)
6. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions-I. Math. Program. **14**(1), 265–294 (1978). https://doi.org/10.1007/BF01588971
7. Khuller, S., Moss, A., Naor, J.: The budgeted maximum coverage problem. Inf. Process. Lett. **70**(1), 39–45 (1999)

8. Shachnai, H., Tamir, T.: Polynomial time approximation schemes. In: Handbook of Approximation Algorithms and Metaheuristics, pp. 9.1–9.21. Chapman & Hall/CRC Computer and Information Science Series (2007)
9. Salloum, M., Dong, X.L., Srivastava, D., Tsotras, V.J.: Online ordering of overlapping data sources. VLDB **7**(3), 133–144 (2013)
10. Rekatsinas, T., Dong, X.L., Srivastava, D.: Characterizing and selecting fresh data sources. In: SIGMOD, pp. 919–930. ACM (2014)
11. Lin, Y., Wang, H., Zhang, S., Li, J., Gao, H.: Efficient quality-driven source selection from massive data sources. J. Syst. Softw. **118**(1), 221–233 (2016)
12. Lin, Y., Wang, H., Li, J., Gao, H.: Data source selection for information integration in big data era. Inf. Sci. **479**(1), 197–213 (2019)
13. Li, L., Feng, X., Shao, H., Li, J.: Source selection for inconsistency detection. In: Pei, J., Manolopoulos, Y., Sadiq, S., Li, J. (eds.) DASFAA 2018. LNCS, vol. 10828, pp. 370–385. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91458-9_22
14. Cao, Y., Fan, W., Wo, T., Yu, W.: Bounded conjunctive queries. PVLDB **7**(12), 1231–1242 (2014)
15. Fan, W., Geerts, F., Libkin, L.: On scale independence for querying big data. In: PODS, pp. 51–62. ACM (2014)
16. Fan, W., Geerts, F., Cao, Y., Deng, T., Lu, P.: Querying big data by accessing small data. In: PODS, pp 173–184. ACM (2015)
17. Cao, Y., Fan, W.: An effective syntax for bounded relational queries. In: SIGMOD, pp 599–614. ACM (2016)
18. Cao, Y., Fan, W.: Data driven approximation with bounded resources. PVLDB **10**(9), 973–984 (2017)